

**A STATE COMMUNICATION AND SOFTWARE
SWITCHING MODULE AND THIN MIDDLEWARE LAYER
FOR RECONFIGURATION MANAGEMENT IN
RECONFIGURABLE MANUFACTURING SYSTEMS**

MSc Eng (Mechanical) Dissertation

**In fulfilment of the MSc Eng (Mechanical), College of Agriculture, Engineering and
Science, University of KwaZulu-Natal**

Submitted on: 25 May 2015

By MSc Eng Candidate: Roscoe Roy Peter McLean

Student Number: 214584328

Identification Number: 910508 5093 08 4

Supervised by: Mr Jared Padayachee

Co-supervised by: Professor Glen Bright

1. Declaration of Submission and Originality

As the Candidate's Supervisor, I agree to the submission of this dissertation.

Signed: Date:
Mr Jared Padayachee

As the Candidate's Co-Supervisor, I agree to the submission of this dissertation.

Signed: Date:
Prof Glen Bright

I declare this dissertation has not previously been submitted for any degree in this, or any other university, and it is the student's (MSc Eng Candidate RRP McLean's) own original work.

Signed: Date:
Roscoe Roy Peter McLean

2. Plagiarism Declaration

I, **Roscoe Roy Peter McLean**, declare that:

1. The research reported in this dissertation, except where otherwise indicated, is my original research.
2. This dissertation has not been submitted for any degree or examination at this or any other university.
3. This dissertation does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
4. This dissertation does not contain other persons' writing, unless specifically acknowledged as being sourced from other researchers. Where other written sources have been quoted, then:
 - a. Their words have been re-written but the general information attributed to them has been referenced
 - b. Where their exact words have been used, then their writing has been placed in italics and inside quotation marks, and referenced.
5. This dissertation does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the dissertation and in the References section.

Signed: Date:
Roscoe Roy Peter McLean

3. Declaration of Publications

Details of contribution to peer-reviewed publications that include research presented in this dissertation. The undersigned agree that the following submissions were made and accepted as described and that the content therein is contained in this research.

3.1 Publication 1: SAJIE

McLean, R. Padayachee, J. & Bright, G., “A Thin, Hardware-Supported Middleware Management System for Reconfigurable Manufacturing Systems”, *South African Journal of Industrial Engineering (SAJIE)*. Not yet Published, 16 Pages, 2015.

The paper was accepted in March 2015 and is to be published in the next issue of SAJIE.

RRP McLean was the lead author for this paper and conducted all research and experimentation under the supervision of the other two.

3.2 Publication 2: WASET ICIESE 2015

McLean, R. Padayachee, J. & Bright, G., “A Middleware Management System with Supporting Holonic Modules for Reconfigurable Management System”, *17th International Conference on Industrial Electronics and Systems Engineering (ICIESE 2015)*. Not yet Published, 5 Pages, 2015.

The paper was accepted for oral presentation and publication in March 2015. The conference is to be held in Berlin in September 2015.

RRP McLean was the lead author for this paper and conducted all research and experimentation under the supervision of the other two.

Signed:
Mr Roscoe McLean

Date:

Signed:
Mr Jared Padayachee

Date:

Signed:
Prof Glen Bright

Date:

4. Abstract

Reconfigurable Manufacturing Systems are a new area of operations and manufacturing research. The global need for production systems which can react rapidly to dynamic markets has increased in the last decade and will continue to drive changes in the manufacturing industry. The further development of RMS technologies is therefore highly important for future industries. The Reconfiguration Management and Middleware System (RMMS) developed in this research aimed to form a hardware-supported middleware technology which allows for the fast and seamless ramp-up of heterogeneous machine controllers on a newly reconfigured factory floor. The goal was to allow for the autonomous assignment and switching of software routines on machine controllers after a physical reconfiguration, thereby speeding up the ramp-up of the system.

The technology was based on a recorded literature review and fits into the paradigm of RMS. The RMMS was developed not as a traditional software-heavy layer, but as a thin layer of software assisted by interactive mechatronic hardware, designed to remove heterogeneity in the control software. The system design was based on research into areas of engineering and operations management and followed the Mechatronic design approach. The literature led to a technology that takes the entire RMS paradigm into account and the development was conducted in conjunction with experiments to verify the individual functionality of each sub-system and ensure the overall system's success. The RMMS uses hardware to handle heterogeneity and uses a positioning system (developed by the author) along with an intelligent processing system (a clustering algorithm and artificial intelligence engine) to construct data into a factory floor model. The positioning system, when assisted by the intelligence, operates at an accuracy of over 90%, which is comparable to commercial positioning techniques which cost over ten times more. The RMMS used the developed model to, autonomously and wirelessly, assign new programs to machine controllers after a physical reconfiguration, to complete a factory reconfiguration.

The system was verified through practical scenarios constructed in the Mechatronics laboratory. Realistic reconfiguration operations were performed and the RMMS was required to detect changes in the factory floor and respond by assigning new, appropriate, software routines to each machine controller in the system. Experiments have proved that the system was capable of re-establishing operations in under half an hour, as opposed to a full day using manual techniques.

The system has accurately switched between control routines based on the physical state of the factory floor, which amounts to control reconfiguration. The reconfiguration of factory floor control was successful in four out of four factory layouts tested and therefore successfully does a job no commercially available system can do.

5. Acknowledgments

The author would like to thank, first and foremost, Jared Padayachee – the primary supervisor for the research. His guidance throughout was beyond reproach and without his insights and help in idea development this research would not have been possible. Thanks must also be made to Professor Glen bright, the co-supervisor for his advice and encouragement and to the other post-graduate students sharing the author’s workspace who made a great sounding board for ideas and whose discussions were valued greatly.

6. Turnitin Report

The report, contained on the attached compact disk was generated by the plagiarism detection service, Turnitin.

The report showed a similarity index of 2%, which is well within the valid range (see the figure below). Most of the similarities found were actually in standard sections such as the “Plagiarism Declaration” section above and in small standard sections of code, like the sorting algorithm in section 15.6. In addition, some of the declaration of terms were also flagged – which would be impossible to avoid.

Investigation of the Turnitin report shows that no significant similarities were found and thus the research was original.



7. Table of Contents

1. Declaration of Submission and Originality	i
2. Plagiarism Declaration	i
3. Declaration of Publications	ii
3.1 Publication 1: SAJIE	ii
3.2 Publication 2: WASET ICIESE 2015	ii
4. Abstract.....	iii
5. Acknowledgments	iv
6. Turnitin Report	iv
7. Table of Contents	v
8. Nomenclature	xi
9. List of Acronyms	xii
10. List of Tables.....	xiv
11. List of Figures	xv
12. List of Equations	xvii
PART A.....	1
1. Chapter 1: Introduction	2
1.1 Chapter Introduction	2
1.2 Background and Necessity.....	2
1.3 Existing Research.....	2
1.4 Research Gaps and Problem Identification.....	3
1.5 Research Aims and Objectives	4
1.6 Overview of the Dissertation	5
1.7 Chapter Summary	6
2. Chapter 2: Literature Review	7
2.1 Chapter Introduction	7
2.2 Introduction to the RMS Paradigm	7
2.3 Supervisory Control and Data Acquisition	10
2.4 Industrial Control and Planning	11
2.5 Reconfigurable Machine Tools.....	12
2.6 Factory Floor Reconfiguration in the RMS Paradigm	13
2.7 Enumerating RMS States	13
2.8 Research Gaps.....	15
2.9 Problem Identification in the Current State of the RMS.....	15

2.10	Decomposition of the Identified Problems	16
2.11	Chapter Summary	17
3.	Chapter 3: A Conceptual Solution	18
3.1	Chapter Introduction	18
3.2	System Requirements and Specifications	18
3.3	Aim and Objectives of the Concept	19
3.4	A Software middleware Solution	19
3.5	A Thin, Hardware-Supported Middleware Solution	19
3.6	Technology Review and Breakdown of RMMS	20
3.6.1	Communication in the Network	21
3.6.2	A Real-Time Location System	25
3.6.3	Further Research on RF-Based RTLS Strategies	26
3.6.4	Processing of RTLS Data	30
3.6.5	Homogeneous Machine Controller Communication	31
3.6.6	Data Processing and Program Assignment	32
3.6.7	Physical and Electrical Design	33
3.7	Chapter Summary	34
PART B.....		35
4.	Chapter 4: System-Level Design and Functionality	36
4.1	Chapter Introduction	36
4.2	Control Position and Operation of the RMMS	36
4.3	Structure of the RMMS	37
4.4	Chapter Summary	38
5.	Chapter 5: The State Communication Module	39
5.1	Chapter Introduction	39
5.2	Structure of the SCM	39
5.3	Selection of the Core Components	39
5.3.1	Radio Frequency Module	39
5.3.2	Microcontroller Selection	41
5.3.3	Interfacing of the Modules	43
5.4	Communication: Machine Controller to SCM	44
5.4.1	Electrical Conceptual Designs and Considerations (Step Down)	45
5.4.2	Comparison of Electrical Concepts (Step Down)	48
5.4.3	Electrical Design: Logic Level Step-Down	48

5.5	Communication: SCM to Machine Controller.....	49
5.5.1	Electrical Conceptual Designs and Considerations.....	51
5.5.2	Comparison of Electrical Concepts (Step Up).....	53
5.5.3	Electrical Design: Logic Level Step-Up.....	53
5.6	Peripheral Electrical Design	54
5.6.1	SCM Power Supply	54
5.6.2	Manufacture and PCB Design	55
5.6.3	SCM Battery Considerations	56
5.7	Chapter Summary	57
6.	Chapter 6: Real-Time Location System.....	58
6.1	Chapter Introduction	58
6.2	RSSI-Based Distance Estimation.....	58
6.2.1	Built-In RSSI Command	58
6.2.2	RSSI Using PWM Signal	58
6.2.3	Using RSSI To Measure Distance	59
6.3	Multilateration Localisation.....	60
6.4	Implementation of the RTLS	63
6.5	Chapter Summary	64
7.	Chapter 7: Central RMMS Control Software	65
7.1	Chapter Introduction	65
7.2	Structure of the CRC.....	65
7.3	Programming Methodology of the CRC.....	66
7.4	Complete Structure and Operations GUI.....	67
7.5	Chapter Summary	70
8.	Chapter 8: Data Processing and Program Assignment	71
8.1	Chapter Introduction	71
8.2	Clustering Technique for Processing Data.....	71
8.2.1	Workings of the DBSCAN Algorithm	72
8.2.2	Adapting DBSCAN for the RMMS.....	74
8.3	A Knowledge-Based Method for Program Assignment	75
8.3.1	Inference Engine of the KBS	75
8.3.2	The Knowledge-Base.....	76
8.3.3	Program Assignment Strategy	76
8.4	Chapter Summary	77

9. Chapter 9: Other Design Aspects of the RMMS	78
9.1 Chapter Introduction.....	78
9.2 Implementation of the Wireless Network.....	78
9.3 Physical Design of the SCM.....	80
9.4 SCADA in the RMMS.....	82
9.5 Chapter Summary	83
10. Chapter 10: Sub-Systems Experimentation.....	84
10.1 Chapter Introduction.....	84
10.2 Testing the Electronics of the SCM.....	84
10.2.1 Voltage Step-Down for Communication	84
10.2.2 Voltage Step-Up for Communication	85
10.2.3 Voltage Regulation for Power Supply	86
10.3 Literature Logarithmic Drop-off in RSSI with Distance.....	88
10.4 Linear Increase in PWM Pulse Length with Distance.....	91
10.5 Testing the RSSI Distance Estimation	93
10.6 A Theoretical Test of the Multilateration Algorithm	95
10.7 Testing the RTLS in the with one Roaming Node	96
10.8 Constants and Testing of the DBSCAN Algorithm	99
10.9 Testing the Operation of the KBS	104
10.10 Chapter Summary.....	106
PART C.....	107
11. Chapter 11: Experimentation on the Integrated RMMS	108
11.1 Chapter Introduction.....	108
11.2 Introduction to the Experiment:.....	108
11.3 Situation:.....	108
11.4 Aim	112
11.5 Experimental Methodology	112
11.6 Results	112
11.7 Experimental Discussion	114
11.8 Experimental Conclusion	115
11.9 Chapter Summary.....	116
PART D	117
12. Chapter 12: Discussion.....	118

12.1	Chapter Introduction	118
12.2	The RMMS in the Context of Reconfigurable Manufacturing.....	118
12.3	Experimentation Results	119
12.4	Determination of the Factory Floor State	120
12.4.1	Gathering Machine Configuration.....	120
12.4.2	The Real-Time Location System.....	120
12.4.3	The DBSCAN Refinement Algorithm.....	121
12.4.4	Performance of the State Determination System.....	121
12.5	The Intelligence System and Program Assignment	121
12.5.1	Hardware-Supported Middleware	121
12.5.2	The Knowledge-Based System.....	122
12.5.3	The Reconfiguration of Control	122
12.6	Post Reconfiguration Control Reconnection	122
12.7	Discussion on the RMMS	123
12.7.1	On the Cost of the System	123
12.7.2	Management System Performance	123
12.8	Research Contributions	124
12.9	Impact of the Research Contributions.....	124
12.10	Opportunity for Further Research and Improvements	125
12.11	Chapter Summary	126
13.	Chapter 13: Conclusion.....	127
13.1	Chapter Introduction	127
13.2	Aims, Objectives, Findings and Results	127
13.3	Conclusions.....	128
13.4	Avenues for Further Research.....	128
14.	References	130
15.	Appendices	135
15.1	Appendix A: P2N2222A Transistor Data Sheet	135
15.2	Appendix B: Teensy 3.1 Datasheet Excerpt	137
15.3	Appendix C: UA7812CKCT Datasheet Excerpt	138
15.4	Appendix D: L78S05CV Datasheet Excerpt	139
15.5	Appendix E: Data Sheet for XBee Series 2	140
15.6	Appendix F: C Code for Multilateration.....	141
15.7	Appendix G: Matlab Code for Multilateration	143

15.8	Appendix H: Structure of CRC Classes	146
15.9	Appendix I: C# Code for DBSCAN Experimentation	149
15.10	Appendix J: C# Code for CRC Inference Engine	151
15.11	Appendix K: Raw Data for 10.3	153
15.12	Appendix L: Calculations for 10.3	156
15.13	Appendix M: Raw Data for 10.4	157
15.14	Appendix N: Raw Data for 10.8.....	159
15.15	Appendix O: Knowledge Base for KBS Test.....	163
15.16	Appendix P: Knowledge Base for System Test	164
15.17	Appendix Q: Factory Floor States for 11.6	165

8. Nomenclature

Symbol	Name	Meaning	Units
c	Speed of light	The speed at which light travels in a vacuum	m/s
EPS	Neighbourhood	Area neighbourhood radius for DBSCAN	Depends on data
f	Frequency	Number of cycles or operations per second	Hz
I	Current	The amount of charge flow in a circuit	A
j	Reconfiguration number	Number of aspects of a RMT which can be reconfigured	ul
K	Layout states of an RMS	Number of states a RMS with variable layouts can adopt	ul
K_m	States of an RMT	Total number of states which a RMT can adopt	ul
m	Number of stages	Number of stages in an RMS into which machines can be positioned	ul
$MinPts$	Minimum points	Minimum number of points needed to form a cluster in DBSCAN	ul
N	Number of machines	Number of machines available to an RMS	ul
p	Aspect states	Number of states in which a reconfigurable aspect can be set	ul
Q	Battery capacity	Capacity of a battery to deliver its potential difference	mAh
q	Impossible states	Number of states or combinations which are redundant or impossible for an RMT	ul
R	Resistance	Resistance to current of a conductor	Ω
P	Electrical power	Quantity of electrical power	W
P_k	Received power	Power of an incoming wireless signal	dBm
P_k'	Received power	Power of an incoming wireless signal	mW
R^2	Goodness of fit	Excel value showing how well data fits onto some defined function	ul
s	Software states	Sum of the number of software states for a single hardware configuration	ul
S_{ff}	Factory floor states	Total number of states which a RMS can adopt	ul
t	Time	Time interval between two points	s
V	Potential difference	The difference in voltage potential between the positive and negative poles of a source or between any two nodes of a circuit	V
ρ	RSSI vector	A vector of RSSI values for each beacon for the Maximum Likelihood RTLS	dBm
σ	Standard deviation	Quantity of variation in data, around 68% of data points lie within this amount of the mean	Depends on data

9. List of Acronyms

ACO:	Ant Colony Optimisation
ANN:	Artificial Neural Network
API:	Application Program Interface
APIT:	Approximate Point in Triangle
ASCII:	American Standard Code for Information Interchange
BPM:	Beacon Positioning Module
CNC:	Computer Numerical Control
CRC:	Central RMMS Controller
CSV:	Comma Separated Value
DBSCAN:	Density Based Special Clustering of Applications with Noise
DE:	Differential Evolution
DML:	Dedicated Manufacturing Line
DMS:	Dedicated Manufacturing System
DMT:	Dedicated Machine Tool
EMI:	Electromagnetic Interference
ERP:	Enterprise Resource Planning
FMS:	Flexible Manufacturing System
GPIO:	General Purpose Input/Output
GPS:	Global Positioning System
GUI:	Graphical User Interface
HDMI:	High Definition Multimedia Interface
HMI:	Human Machine Interface
I2C:	Inter-Integrated Circuit
IDE:	Integrated Development Environment
IEEE:	Institute of Electrical and Electronics Engineers
ISA:	International Society of Automation
ISO:	International Organisation for Standardisation
KBS:	Knowledge-Based System
LAN:	Local Area Network
LED:	Light Emitting Diode
MAC:	Media Access Control
MCM:	Mass Customisation Manufacturing
MCU:	Microcontroller

MES:	Manufacturing Execution System
MOM:	Manufacturing Operations Management
MRP:	Material Requirements Planning
MRPII:	Manufacturing Resource Planning
MS:	Microsoft
NPN:	Negative-Positive-Negative
OPTICS:	Ordering Points to Identify the Clustering Structure
PC:	Personal Computer
PCB:	Printed Circuit Board
PIT:	Point in Triangle
PLC:	Programmable Logic Computer
PSO:	Particle Swarm Optimisation
PWM:	Pulse-Width Modulation
RAM:	Random Access Memory
RF:	Radio Frequency
RFI:	Radio Frequency Interference
RFID:	Radio Frequency Identification
RMMS:	Reconfiguration Management and Middleware System
RMS:	Reconfigurable Manufacturing System
RMT:	Reconfigurable Machine Tool
RTLS:	Real-Time Location System
RSSI:	Received Signal Strength Indication
RTOF:	Real Time of Flight
SCADA:	Supervisory Control and Data Acquisition
SCM:	State Communication Module
TDMA:	Time Division Multiple Access
TDOA:	Time Difference of Arrival
TOA:	Time of Arrival
UART:	Universal Asynchronous Receiver/Transmitter
UKZN:	University of KwaZulu-Natal
UWB:	Ultra Wide-Band

10. List of Tables

Table 1: System Specifications	18
Table 2: Comparison of Wired and Wireless Communication	21
Table 3: Common Unlicensed Bands	22
Table 4: Comparison of Frequency Characteristics	23
Table 5: A Summary of Industrial Wireless Possibilities	25
Table 6: Summary of Localisation Techniques	26
Table 7: Weighted Comparison of Wireless Techniques	40
Table 8: Comparison of Zigbee Modules	40
Table 9: Comparison of Arduinos	43
Table 10: Standard Message Table: SCM to Machine.....	45
Table 11: Comparison Table of Logic Level Converters	48
Table 12: Machine Controller Step-Down Ratios	48
Table 13: Resistor Values Chosen for Different Controller Voltages.....	49
Table 14: Standard Message Table: Machine to SCM	50
Table 15: Comparison of Logic Level Step-Up Methods.....	53
Table 16: Results of Step-Down Experiment	85
Table 17: Results of Step-Up Experiment	86
Table 18: Results of Voltage Regulation Experiment	87
Table 19: Summary of the Experimental Results.....	89
Table 20: Summary of the Experimental Results.....	91
Table 21: Results of the Distance Estimation Test	94
Table 22: Results of the Multilateration Experiment.....	96
Table 23: Tabularised Results of the Roaming Node Experiment	98
Table 24: Actual Position of Machines.....	100
Table 25: Results of the Variable EPS Experiment (Part 1).....	103
Table 26: Results of the DBSCAN Test (Part 2)	103
Table 27: Results of the KBS Experiment	105
Table 28: Machine-Part Matrices and Cell Formation	109
Table 29: Machines Available to the Factory	110
Table 30: Layout and Configuration Data for the Machines	111
Table 31: Results of Layout 1 Reconfiguration	113
Table 32: Results of Layout 2 Reconfiguration	113
Table 33: Results of Layout 3 Reconfiguration	113
Table 34: Results of Layout 4 Reconfiguration	114
Table 35: Summary of All Experimental Results	119

11. List of Figures

Figure 1: A Simple Cellular Reconfiguration.....	7
Figure 2: How the 5 Characteristics Link to the RMS Requirements	9
Figure 3: Breakdown of Key Reconfigurability Characteristics.....	9
Figure 4: Layers of Manufacturing Control Software	11
Figure 5: A Comparison of Literature MMS and the Proposed Concept	20
Figure 6: Localisation Error Change with Relation to Beacons	29
Figure 7: The MMS in the Control Structure of an RMS	36
Figure 8: MMS Software Data Flow for a Reconfiguration.....	37
Figure 9: Structure of the MMS	38
Figure 10: Function Blocks in the SCM	39
Figure 11: XBee Series 2 Whip Module and Sensitivity Pattern [97].....	41
Figure 12: The PJRC Teensy 3.1	43
Figure 13: Connecting the Teensy 3.1 to a XBee	44
Figure 14: Pseudo-Code for SCM (left) and Machine Controller(right)	45
Figure 15: Opto-Coupler to Convert Logic Levels.....	46
Figure 16: A Relay to Convert Logic Levels	46
Figure 17: Voltage Divider to Convert Logic level.....	47
Figure 18: A NPN Transistor to Convert Logic Level	47
Figure 19: A Retail Logic Level Converter	47
Figure 20: Circuit for the Logic Level Step-Down of All Channels.....	49
Figure 21: Pseudo-Code for SCM and Machine Controller	50
Figure 22: An Example of Responding to SCM Software Switching	51
Figure 23: Opto-Isolator to Convert Logic Levels	52
Figure 24: A Relay to Convert Logic Levels	52
Figure 25: A Transistor to Convert Logic Levels.....	52
Figure 26: A P2N2222A Transistor.....	53
Figure 27: Circuit Diagram for Step-Down Circuit.....	54
Figure 28: Cascade Regulator Circuit Diagram	54
Figure 29: Battery Switching Circuit	55
Figure 30: PCB Design	56
Figure 31: Assembled SCM PCB	56
Figure 32: PulseIn Code Snippet	59
Figure 33: Localisation with Two Beacons	60
Figure 34: Four Beacon Multilateration	61
Figure 35: Pseudo-Code for Multilateration	61
Figure 36: Labelled RTLS Diagram for Equation Generation.....	62
Figure 37: Flow Chart of RTLS Operation	64
Figure 38: Communications & Software Infrastructure	65
Figure 39: Software Structure of the CMC.....	67
Figure 40: GUI Start Window	68
Figure 41: Request State Window.....	68
Figure 42: Physical Factory Floor State Window	68
Figure 43: Run AI and Assign New Program Window	69
Figure 44: Windows for Successful (left) and Unsuccessful (right) KBS Matches	69
Figure 45: Window Shown During Assignment Process	69
Figure 46: Windows for Successful (left) and Unsuccessful (right) Assignment	69

Figure 47: DBSCAN Result Map Showing Noise.....	72
Figure 48: DBSCAN Linking Separate Clusters.....	72
Figure 49: Pseudo-Code for DBSCAN	73
Figure 50: Pseudo-Code for DBSCAN Method: ExpandCluster.....	74
Figure 51: KBS Pseudo-Code	76
Figure 52: Zigbee Mesh Network Topologies [96]	79
Figure 53: Example MMS Mesh Network Paths	79
Figure 54: CAD of Box	81
Figure 55: CAD of Lid (left) and Assembly (middle & right).....	81
Figure 56: CAD of Case Assemblies (left without PCB, right with).....	81
Figure 57: CAD of Closed Case Assembly	82
Figure 58: Photograph of the Module (Lid Removed)	82
Figure 59: Pseudo-Code for SCADA Operation.....	83
Figure 60: Distance vs. RSSI for Each Experiment.....	89
Figure 61: Distance vs. RSSI for the Average Reading	89
Figure 62: Distance vs. PulseIn Value for Each Experiment.....	92
Figure 63: Distance vs. PulseIn Value for the Average Reading.....	92
Figure 64: Positions Used in the Experiment	97
Figure 65: Photograph of the Factory Floor Test Space	97
Figure 66: Results Shown in their Position on the Factory Floor	98
Figure 67: Hypothetical Test Layouts	100
Figure 68: An Example Gaussian and Random Distribution	101
Figure 69: Gaussian Distribution of Machines in Layout 3.....	102
Figure 70: Random Distribution Cases of Layout 3	102
Figure 71: Stylised Gantt Chart for Production	109
Figure 72: Machine Layout in Each Period.....	110
Figure 73: Reconfiguration Operations.....	111
Figure 74: Results of RTLS Roaming Node Test	121

12. List of Equations

[Eq. 1]	13
[Eq. 2]	13
[Eq. 3]	14
[Eq. 4]	14
[Eq. 5]	14
[Eq. 6]	27
[Eq. 7]	27
[Eq. 8]	27
[Eq. 9]	28
[Eq. 10]	28
[Eq. 11]	49
[Eq. 12]	59
[Eq. 13]	59
[Eq. 14]	59
[Eq. 15]	62
[Eq. 16]	62
[Eq. 17]	62
[Eq. 18]	62

PART A

This part of the dissertation details the literature-based research conducted and the conceptualisation of a solution to problems identified in the RMS paradigm.

1. Chapter 1: Introduction

1.1 Chapter Introduction

This chapter serves to introduce the reader to the concept of the Reconfigurable Manufacturing System (RMS), introduce some of the significant findings of the literature review, introduce the aims and objectives of the research, and give a dissertation overview. This introduction covers the background and necessity of this research, introduces the existing research into RMS middleware and introduces applicable problems and issues found in the literature review. This chapter aims to familiarise the reader with the problem at hand, describe the research aims and to introduce a guide to the dissertation.

1.2 Background and Necessity

The concept of RMS was formulated due to the global necessity for production systems that are able to economically evolve according to changes in markets and products [1-7]. The frequency and unpredictability of these changes in modern industry require a manufacturing system capable of a rapid response to change [5, 6]. This market-driven necessity led to a shift in the focus of manufacturing research development towards the responsiveness of the manufacturing system to changes in requirements, regulations and competition [8].

There is a large volume of research into the RMS paradigm, the technology used in the machines of an RMS and other constituent aspects which make up the RMS. The sections that follow discuss and review the current state of reconfigurable manufacturing systems and identify problem areas and gaps in the research.

Research into Reconfigurable Manufacturing Systems has been underway for well over a decade, since it was introduced in the closing years of the last millennium in a paper by Koren et al [1] at the University of Michigan. This paper has been the basis for the RMS paradigm and its five essential characteristics: *Modularity*, *Integrability*, *Customisation*, *Convertibility*, and *Diagnosability*. A 6th characteristic, *Scalability*, is now outlined commonly as an essential characteristic of a RMS, in addition to the other five characteristics [2, 6, 9] or in place of customisation [7]. The RMS paradigm aims to create a system which can respond to new market needs rapidly, economically, efficiently, and effectively. This design for reconfigurability from the outset of a project is the central concept of RMS and makes RMS an economically attractive manufacturing system design for high variety batch manufacturing, volatile market sectors and Mass Customisation Manufacturing (MCM) [5].

There were some gaps in current knowledge and implementation of RMS which this research has aimed to address, specifically relating to factory floor ramp-up after a reconfiguration and communication with heterogeneous entities. The primary aim of this research was to propose a full-scale developmental platform and implement a laboratory-scale solution to problems associated with the reconfiguration process in the RMS paradigm. These problems are identified in sections 2.8 to 2.10 and are mentioned in this introduction in section 1.4.

1.3 Existing Research

A recent publication was presented by researchers from the Norwegian University of Science and Technology dealing with the development of a software framework for RMS [10]. The software, IceHMS, operates over an internet protocol and has many complex characteristics. This research produced a system which uses a relatively thin layer of middleware to communicate status

information, controls and provide communication with a broad variety of machines (both type and brand) and aims to simplify the heterogeneity of a RMS. The layer researched in that paper was thinner than traditional middleware, it is still complex from a computer programming point of view and this research (indicated by the research title) included the creation of a very thin middleware layer with hardware support as an objective.

There are many quoted cases of research into RMSs as a concept but little international research effort has been devoted to automatic control reconfiguration, planning and operations management functions of a RMS after a physical change has been made to the system. RMSs require a middleware management system to reconcile physical reconfigurations (i.e. changes to machines, cells and processes) with high level Materials Requirement Planning (MRP), Manufacturing Resource Planning (MRPII) and ERP software functions.

This research had to address the problem of heterogeneity on the factory floor, this is usually a job for a suitable form of middleware system. The middleware system had to play an important role in the system designed and developed in this research. Traditionally, middleware serves as a form of communication software between distributed systems and a central controller. The heterogeneity in manufacturing is represented by machines, cells, material handling systems and their associated controllers. The middleware management system that will be developed during this research will provide advanced information modelling, communication and control utilities required by a RMS. This will enable the automatic reconfiguration of manufacturing management and execution functions, in parallel to the physical reconfiguration of the system structure and hardware resources.

1.4 Research Gaps and Problem Identification

The research done using existing literature aimed to introduce the author to the Reconfigurable Manufacturing System paradigm and was documented to allow the reader to become familiar with the paradigm. Problems and gaps were looked for during the review, with particular focus on issues surrounding the reconfiguration process. The RMS paradigm has been intensely studied in the past fifteen years, but some areas, identified in the literature review, have not been adequately addressed. The targeted area of research in this paper, the re-establishment of control after a reconfiguration, has not been studied in sufficient detail for industrial adoption and this gap affects the performance and reconfigurability of a RMS. The literature review section (Chapter 2) found that a substantial issue during the reconfiguration of a system is the delay between completion of physical restructuring and the start of system operation in its new period. This process is known as system ramp-up and the speed of this system ramp-up directly affects the flexibility and effectiveness of a RMS [1]. The critical process involved in ramp-up is the re-establishment of control of heterogeneous entities on the factory floor. This process is currently old-fashioned, labour-intensive and error-prone and a more automated, modern solution, directly aimed at the RMS paradigm is proposed in this paper.

During the literature review some complications which arise from the modular and reconfigurable nature of the RMS were identified. These include the large number of possible combinations of factory floor state (which are investigated in detail in section 2.7) and the heterogeneous nature of a modern RMS which makes use of multiple control architectures and brands of machine controller. This heterogeneity causes issues in communication and had to be addressed in the research.

Sections 2.8 to 2.10 discuss and decompose the problems which were identified in the rest of the literature review (Chapter 2). The problems identified corresponded to areas in which research has not been adequately conducted (the gaps which were identified) and in particular, this research focussed on system ramp-up. Currently, system ramp-up is slower than it could be because of the use of older, traditional methods for the re-establishment and reconfiguration of control. A clear opportunity for improvement was identified – automation of the process of program re-assignment. This in itself is plagued by the problem of heterogeneity on the factory floor, which makes the reconfiguration of control difficult and thus this problem had to be addressed in this research.

1.5 Research Aims and Objectives

The aim of this research was to increase the ramp-up speed of a RMS through the streamlining of the software reconfiguration process. The research had the following objectives to achieve it:

- ◆ Conduct a literature review to gain knowledge on the RMS paradigm.
- ◆ Identify and decompose research gaps and problems in the RMS paradigm.
- ◆ Conceptualize a solution to identified problems.
- ◆ Develop the concept into a laboratory-scale system capable of decreasing the ramp-up time of a RMS, while dealing with control heterogeneity.
- ◆ Test the various sub-systems developed as part of the system.
- ◆ Test the system using a laboratory-scale production system.
- ◆ Discuss and make conclusions on the outcomes of the research system development.

The Reconfiguration Management and Middleware System (RMMS) which was developed in this research had the primary aim of developing a system capable of speeding up the re-establishment of control (including control reconfiguration) of a RMS after a reconfiguration operation. This had to be achieved by the RMMS while reducing human error and labour time. In order to achieve this, the core objectives of the RMMS were to:

- ◆ handle the heterogeneity in factory floor level machine controllers;
- ◆ automate the reconfiguration and re-establishment of control with the following strategy:
 - determine the new state of the factory floor;
 - develop the state into a factory floor model;
 - match that model up to some known control strategy;
 - send the updated control instructions to each machine controller;
- ◆ be cost-effective and scalable;
- ◆ reduce human interaction, interference and labour time;
- ◆ create a suitable software framework for operation and management of the system;

The purpose of reducing the ramp-up time of a RMS, via the RMMS, was to increase its flexibility and reconfigurability, thereby making it a more feasible manufacturing paradigm. These objectives had to be handled through the development of suitable technologies and strategies and this research aimed to develop these to a level of a prototype laboratory system with a framework capable of being scaled to an industrial level. All processes defined as part of the automated reconfiguration and re-establishment of control had to be handled by the proposed RMMS, while it also dealt with the complex nature of the RMS state models and the heterogeneity of the machine control architectures.

Throughout the research, the Mechatronic design philosophy was followed, this involves problem solving using a combination of multiple aspects of engineering. The amalgamation and integration of the different aspects lead to an elegant and optimal solution to the problem. The design approach used in mechatronics and this research involved in-depth research into the problem at hand and careful planning of the solution before it was implemented. The research was applied and empirical in nature, although the research in this project did cover some conceptual ideas. The outcomes of the research will be performance tested.

1.6 Overview of the Dissertation

The following list gives a summary of what is covered in each of the chapters of this dissertation:

- ◆ Chapter 1;
 - Introduces and supports the necessity of the research.
- ◆ Chapter 2;
 - Literature review and problem identification.
- ◆ Chapter 3
 - A conceptual solution is introduced and research into technologies needed to make the concept a reality are investigated.
- ◆ Chapter 4
 - A discussion on and description of the final concept for the RMMS, including its structure and aspects which required design work.
- ◆ Chapter 5
 - Design of the State Communication Module (SCM), which houses the hardware support for the RMMS, the communication strategy and on-board processing and connects to the machine control.
- ◆ Chapter 6;
 - Development of the RTLS which gathers essential information about the layout of the factory floor which can be processed into a factory floor state.
- ◆ Chapter 7;
 - Development of the Central RMMS Controller, which contains the middleware layer, the graphical user interface and is the structure – in which the intelligent system discussed in the next chapter was built.
- ◆ Chapter 8;
 - Development of the intelligent system which converts raw factory floor state data gathered by the middleware and RTLS into a usable factory floor state and then uses that state to reconfigure the factory floor controllers.
- ◆ Chapter 9;
 - Covers the other design aspects of the RMMS which do not warrant their own chapters. These are; the wireless communication strategy, physical design of the SCM and inclusion of the SCADA framework for the RMMS.
- ◆ Chapter 10;
 - Documents each of the experiments conducted during the design process, which either helped to guide the process or were used to verify the functionality of each essential sub-system before integration.
- ◆ Chapter 11;

- Documents the experiment conducted on the integrated RMMS, where the real-world operation of the system is simulated at a laboratory scale.
- ◆ Chapter 12;
 - A discussion on the performance of the system and outcomes of the research.
- ◆ Chapter 13;
 - A conclusion of the research.

1.7 Chapter Summary

This chapter gave a brief introduction into the RMS paradigm and explained the necessity for a system to handle heterogeneity and streamline system ramp-up. A short overview of the existing research into similar technologies was given and problems with both current research and existing solutions were identified (these are expanded in sections 2.8-2.10). The core aim and objectives of the research were presented and the solution system objectives necessary to achieve them were identified and commented on in this chapter. Finally, a chapter overview of the dissertation was given, with a brief explanation of their contents.

2. Chapter 2: Literature Review

2.1 Chapter Introduction

This chapter covers background to the concept of Reconfigurable Manufacturing Systems and goes into detail in research on the necessary technologies involved in RMS implementation. The shortfalls of some of these technologies and in the implementation of RMSs in general are also outlined and elaborated on in this chapter. The RMS paradigm is explained in greater detail than in the previous chapter and its key characteristics are explained. The supporting technologies and strategies of the RMS paradigm, such as reconfigurable machine tools and RMS modelling using enumeration of states, are introduced and discussed. At the end of the chapter applicable research gaps and problems within the RMS paradigm are identified and decomposed. This chapter serves to familiarise the reader with this paradigm, so the text is made up of summarised content of the current body of work on the RMS paradigm and its supporting technologies and some conclusions and adaptations of the research.

2.2 Introduction to the RMS Paradigm

This literature review aimed to investigate and report on the state of the art of RMS technology, current RMS-related research and specifically on the state of management and communication within the RMS paradigm. The core attributes of a RMS which are addressed directly by the research are Convertibility and Modularity, though all aspects were to be considered and would influence the design and development process. In this research, the particular focus is on cellular type reconfigurable manufacturing systems which can reconfigure themselves in their layout. A very simple cellular reconfiguration, for the purpose of visual clarity is shown in Figure 1 – a machine is swapped between cells and another machine is removed and replaced with one from outside the system.

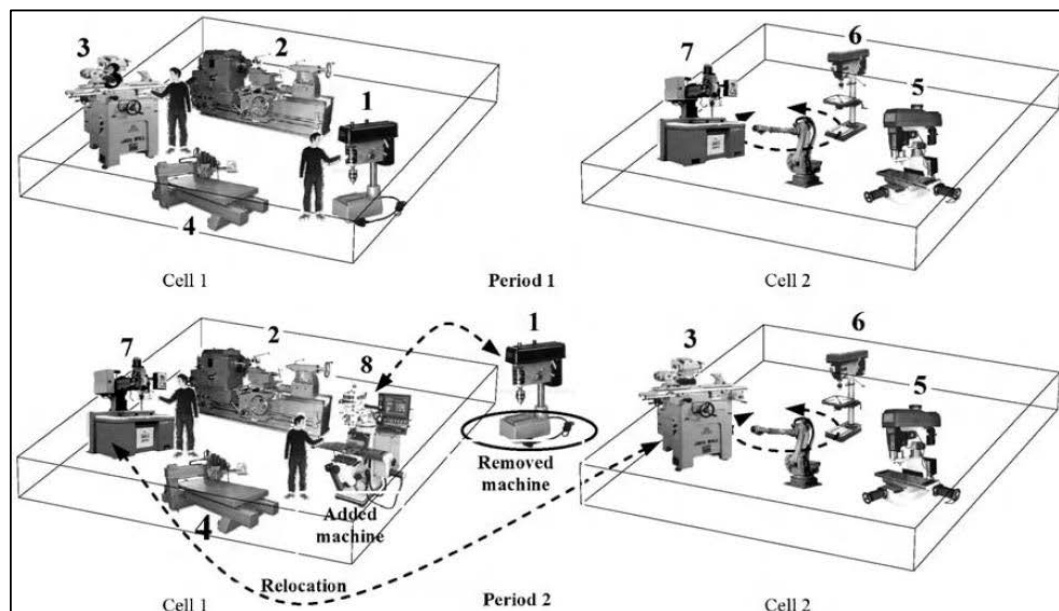


Figure 1: A Simple Cellular Reconfiguration

For the purpose of this research the combination of a machine's position on the factory floor and its configuration (both in hardware and software) will be referred to as its state. In a similar manner, a combination of all the machine states on the level of factory floor will be referred to as the factory state or RMS state. Thus the factory floor state encompasses a complete description

of the physical and software structure of the system including the combination of unique manufacturing cells made up of particularly configured machines.

Reconfigurability in an industrial manufacturing environment is defined by Gumasta et al [11] as being “*the ability to repeatedly change and rearrange the components of a system in a cost-effective way*”. A RMS is one designed to conduct a fast change of structure or layout of machine hardware and software, at a machine and factory level [2]. In other words, a RMS is a manufacturing system designed specifically to be capable of reconfiguration as defined above. It should combine a measure of the flexibility of a Flexible Manufacturing System (FMS) with some of the high productivity of a Dedicated Manufacturing Line (DML) or Dedicated Manufacturing System (DMS) and should be able to react rapidly and cost-effectively to changes in requirements [1]. A FMS may aim for large amounts of flexibility and the ability to produce widely different products, while a RMS aims to be able to produce (at a lower cost and higher volume than a FMS) a particular customised set of parts or products which are part of a family which require similar processes. This customisation is a limitation on the flexibility of the system, but allows for higher throughput and a lower unit (and system) cost. In comparison to DMSs, the RMS aims to be able to reconfigure (in order to adapt to new market requirements) at a much lower cost and with a drastically reduced ramp-up time.

The literature contains the following explanations of the six key characteristics of the RMS paradigm for a high level of reconfigurability as the characteristics were proposed, expanded and commented on [1, 6, 7, 12, 13].

- ♦ To have *Modularity* in a RMS it is required that all important components in the system are modular in design and can be changed and replaced seamlessly within the machines of the RMS. This allows for components within RMTs and software to be changed in a fast and efficient manner, with minimal compatibility issues and debugging.
- ♦ *Integrability* is important to allow the RMS to adopt new technology and integrate the new technology with older hardware and software in the RMS, aiding the system’s ability to be updated and added to without major interruption. This also allows the system to be built at a lower cost.
- ♦ Detecting problems in quality of the output product and in the reliability of the RMS output is critical for quick ramp-up times and a feasible system. The ability to detect and identify this is known as *Diagnosability*.
- ♦ *Convertibility* is required for a rapid changeover between products already being manufactured and timely and efficient adoption of new products into the RMS. It directly applies to the actual reconfigurability of the system.
- ♦ *Customisation* in a RMS requires that flexibility in the RMS (in both hardware and software) should only extend to a particular product family or families. This limits the flexibility and thus prevents the RMS from being over-capable therefore reducing the overall system cost.
- ♦ *Scalability*, the newest addition to the key characteristics of a reconfigurable system stems from the requirement of an RMS to deal with market uncertainty and fluctuations in demand while maintaining a low cost.

Figure 2, shown below, (adapted from [7]) shows how each of the 5 characteristics a good RMS (as chosen in the cited paper) link to the requirements of the RMS paradigm:

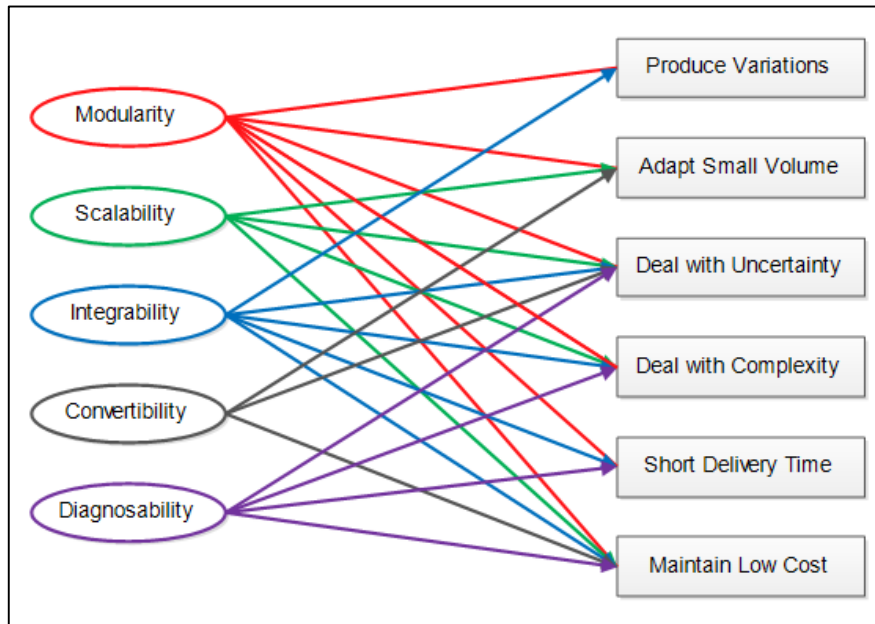


Figure 2: How the 5 Characteristics Link to the RMS Requirements

A RMS is required to deal with changes and demands in the marketplace including, but not limited to, examples outlined in the above diagram. The need to deal with these changes and demands is the driving force behind reconfiguration. In order to be considered truly reconfigurable, the system must have all the characteristics listed above.

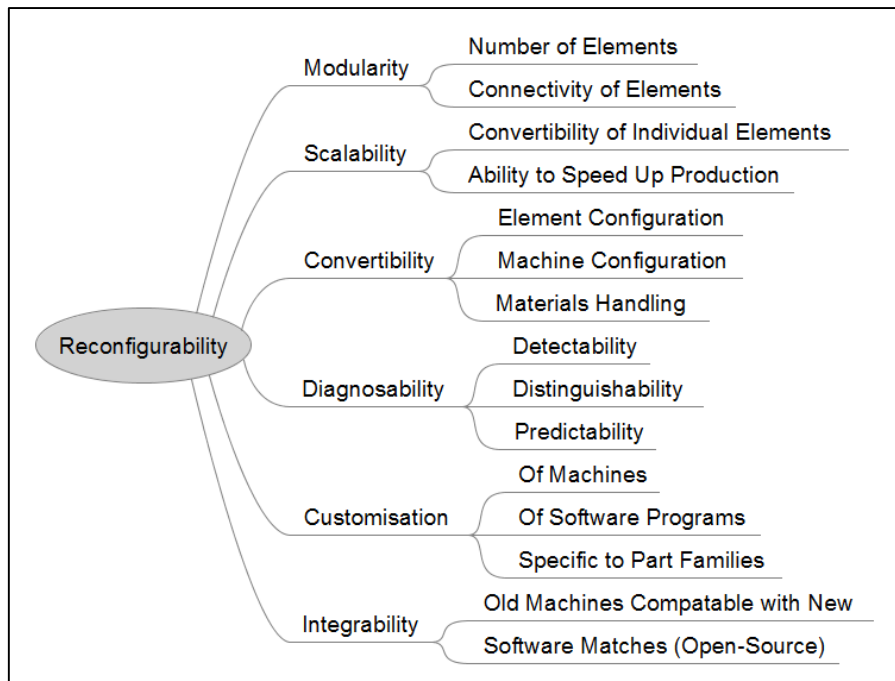


Figure 3: Breakdown of Key Reconfigurability Characteristics

RMSs should be able to produce a large varied set of customised products, often small in quantity, but taking advantage of a lucrative gap in the marketplace [11]. The quantities may be small but the speed at which the products must be produced should be high to ensure a gap is taken and to

give the system scalability. A RMS should have only the exact requirements and functionality needed for a current production plan (Customisation) [1]. Without considering this, the cost of the system would likely be too high to make the system a feasible development. Figure 3 shows how the characteristics of reconfigurability branch out into specific requirements [11].

The RMS paradigm provides the ability of a factory to evolve with changing market requirements, an extremely powerful tool for many industries. The use and development of RMS makes some new concepts such as MCM more feasible and allows for faster production of products traditionally limited to small scale production [5]. The following sections (2.3 – 2.7) discuss the background to technologies used by RMS and how these technologies handle the demands of the RMS paradigm.

2.3 Supervisory Control and Data Acquisition

Supervisory Control and Data Acquisition (SCADA) software serves as an indirect industrial control system centring on a computer system monitoring and process supervision [14]. SCADA differs from automation in that it provides an operator with data via a Human-Machine Interface (HMI) rather than providing autonomous real-time feedback control to an input device. SCADA is used to log data and to monitor the ongoing processes via the HMI. Many companies ranging from ABB to Siemens provide SCADA and HMI systems including hardware and software. SCADA systems usually store data in a complex distributed database and gather this data from the factory network. Depending on the source, SCADA has been through three or four stages, with a number of sources not adding the newer fourth generation [15, 16]. In summary, SCADA has developed in the following four broad stages of development:

1. Monolithic;
2. Distributed;
3. Networked;
4. Internet of Things/Web-based;

Many companies, especially those involved in manufacturing, still operate in the second generation, with multiple access points connected via a LAN [16]. Because these systems are still proprietary, there are significant security risks from hackers and competition. Companies which have moved on to the third generation, which is a small developmental step from the second, have less of a security threat through the use of a secure open communication system. The third generation also has the ability to access the web and can share information, but does not operate primarily as a web-based application. This web access allows the SCADA to survive network crashes and improves reliability. Lately there has been research into web-based SCADA systems and into combining automation and SCADA into one technology. Siemens has introduced the newest generation of Simatic Win-CC which includes automation, SCADA and HMI software in one web-based package. The web ability of this type of technology allows for remote monitoring of processes on a factory floor, although the security risks and physical limitations of control imply that only traditional SCADA functionality should be implemented on the web based system.

SCADA is especially useful in an RMS environment, the ability to monitor the conditions in the production environment and record performance and production data greatly increases the feasibility of a RMS. A SCADA system for RMSs would need the ability to reconfigure in itself

to meet any possible new requirements [17]. SCADA in the RMS paradigm must be rapidly flexible and capable of handling a broad array of data gathering and supervisory techniques.

Because this research will be experimentally implemented using the Festo MPS system at UKZN, further research was conducted into Festo's SCADA technology. Festo provides a SCADA system named VipWin, which uses a Microsoft (MS) .COM technology for server communication [18]. The VipWin software forms part of Festo's OPC EasyServer software. This software has become out of date as the newer technology has adopted MS .NET and Indigo communications architectures [19]. The non-adoption of this technology means that the system does not possess integrability and thus is not well suited to RMS. It appeared at this stage that a simplified SCADA system will have to be developed, or another SCADA software chosen implemented in the experimentation in order to function seamlessly with the laboratory-scale RMS.

2.4 Industrial Control and Planning

Enterprise Resource Planning (ERP) and Manufacturing Operations Management (MOM) software make decisions regarding the future and current operations and management of a manufacturing system. MOM is a developmental step from a Manufacturing Execution System (MES), though the differences are unimportant for this research and for the purpose of this paper the term MES will be used. The following illustration (Figure 4) shows the layers of a manufacturing control system [20]:

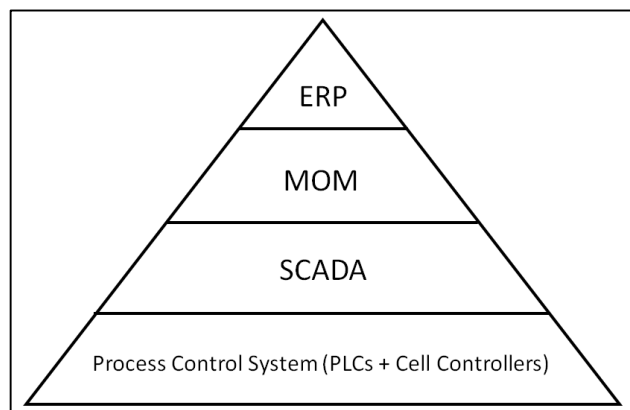


Figure 4: Layers of Manufacturing Control Software

True ERP was developed in the 1990's as computer technology began to become more powerful and thus more useful in the manufacturing industry [20]. It is the highest level of manufacturing software and plans for all the lower level layers of software. ERP is made up of multiple modules each with a specific planning role. Most applicable to the RMS paradigm and the concept of a RMMS are the modules covering Manufacturing Resource Planning (MRPII), workflow planning and, to a certain extent, quality management. ERP has been extended to include more modules and control the planning of entire businesses, from manufacturing and purchasing to payroll. In the last few years, ERP has been developed into a cloud based software, aiding resilience to crashes. Decisions made by these high level software elements are reflected in the associated manufacturing systems and the relevant feeder systems. ERP is widely commercially available and can, in small scale cases and when using limited modules of ERP, be written into code quickly. The development of ERP is beyond the scope of this research.

MES software lies below the ERP layer but above the SCADA and control system of a factory. MES is more involved in the direct control of a manufacturing system and receives the

manufacturing plan from the ERP software or from direct instructions from a designer. The MES can be seen as a factory control system which does not deal directly with the sensors and actuators on the shop floor [21, 22]. The MES is usually driven by MRP control, which thus guides the factory control. There are many brands of MES software solutions, many of which are offered by the same companies that deal with industrial ERP systems. The development of a demonstration MES and MRP are beyond the scope of this research.

2.5 Reconfigurable Machine Tools

The Reconfigurable Machine Tool (RMT) is an essential piece of any RMS, without RMTs a RMS will not be able to economically reconfigure while retaining the ability to handle sufficient part numbers [4, 23]. The RMT is a machine designed to handle MCM and its requirements and thus the needs of the RMS paradigm. It has a combination of the characteristics of a Dedicated Machine Tool (DMT) and of a Computer Numerical Controlled (CNC) machine [23]. There has been an effort in RMT research to develop a modular-type method for reconfigurability [8]. A multi-axis CNC machining tool can provide massive amounts of flexibility, but CNC machines are generally too expensive and slow for most production purposes, but DMTs are not designed from the outset to be reconfigurable, which means that these machines cannot adjust to new product requirements cost-effectively. The RMT fills this gap by allowing a number of aspects of the machine to be altered to handle a finite sub-set of parts and/or operations [3, 9, 23, 24]. This gives the RMT the ability to perform a limited number of operations to a specific part family or families. A true RMT must therefore be able to alter in multiple ways. This research will include a strategy for analysing these possible alterations to accurately sense the configuration of the machine. This strategy will include a method to communicate this configuration back to the main controller running MES and ERP software.

The RMT must be able to change its physical (and software-level) configuration in order to handle part changes. It should be able to perform this reconfiguration in a rapid and cost-effective manner to speed up the ramp-up time of and make feasible a factory reconfiguration [23, 25]. The number of possible physical changes to an RMS is dictated by the amount of parameters within the RMT which can be altered and by the ability of the factory floor layout to be changed. Some of these alterations may be purely software based, such as the drilling depth or number of holes drilled by a RMT (though this borders on CNC territory). Certain RMTs may require modular control software changes to provide a new manufacturing service [8]. Also, many will be based on which of a range of physical modules can be placed and replaced in the machine for the operation [8] and/or what position an element of the RMT can be set to [23-25].

Research has been conducted into modular control of RMTs so that control of additional modules or changed modules is added in a 'plug-and-play' fashion in order to significantly reduce reconfiguration complexity and down time [8]. The advantage, for the purpose of this research, of this method of control is that a machine can know its physical configuration based on the control modules attached to it.

CNC machines are not practical for use in a RMS in large numbers because there is often excessive cost due to unused built-in functionality [1, 8, 25] and closed architecture [3]. They are often slow to work because it is not their purpose to produce parts at a rapid rate. They provide a very accurate method of producing complex geometries and are thus sometimes used to provide a large degree of flexibility.

2.6 Factory Floor Reconfiguration in the RMS Paradigm

There are many reasons for a RMS to change its configuration; changes are mostly market-driven with new competition and new products forcing production changes at a high frequency (that is, compared to traditional manufacturing where changes happen on the scale of many months). A well designed RMS should be able to handle the following changes with a rapid and cost-effective reconfiguration [1]:

- ◆ stochastic high frequency introduction of new products;
- ◆ existing products with parts changes;
- ◆ substantial changes in product demand;
- ◆ changes in product mix demand;
- ◆ government regulations and requirements changes;
- ◆ available process technology changes;

The RMS can change in a number of ways in order to adapt to changing market requirements in any combination of the following ways:

- ◆ machines can change position;
- ◆ machines can be added to or removed from the system;
- ◆ materials handling programs can change;
- ◆ physical configuration of machines can be changed;
- ◆ software on machines can be changed;

The combination of these changes can lead to a massive number of different possible configurations. Below is a proposed method for the enumeration of these possible changes.

2.7 Enumerating RMS States

Before the RMS paradigm was proposed there was already a large amount of research into factory configurations and how layouts affect the performance of a manufacturing system. Shortly before RMS was defined the literature grew, especially in the case of any form of agile or flexible manufacturing system [26, 27]. According to Koren and Shpitalni [12], a RMS factory floor can change configuration by adjusting the layout of the machines. This paper stated that the number of possible RMS configurations, given by K , was governed by the number of machines, N , and the number of stages in which those machines could be arranged, m . The equation depends on whether m is defined as being an *exact* number ([Eq. 1]) of stations or whether it can be anything *up to* m stations ([Eq. 2]).

$$K = \binom{N-1}{(N-m)(m-1)} \quad [\text{Eq. 1}]$$

$$K = \sum_{m=1}^N \binom{N-1}{m-1} = 2^{N-1} \quad [\text{Eq. 2}]$$

This idea of configuration based only on the set-up of machines on the factory floor is also investigated in other research [28]. This method of calculating, however, only describes the number of layout position states of the machines within RMS. Each RMT or other machine also had its own finite number of configurations which also changed the effective configuration of the RMS as a whole. The formula for the total enumeration of possible factory floor states will have

to include the number of possible machine configurations on the factory floor and of the number of configurations of those machines themselves. This product is a fairly simple calculation, but is worth mentioning and investigating when enumerating the number of states possible for an RMS. CNC machines are capable of huge amounts of software configurations and are more suitable for use in true FMSs and are thus not included in these enumerations.

For the physical configuration of a modular or position-changeable RMT there are two parameters to consider, the number of reconfigurable aspects, j , and the number of possible configurations of each of these aspects, p , of the RMT (i.e. number of different modules that could be placed or the number of positions a component could be in). The number of configurations which are redundant or impossible, q , should also be taken into account and subtracted. Note that this does not take into account the number of additional software changes in the RMT that can occur without a hardware change. It is clear that with a hardware change there must be a software reconfiguration, but it is possible that there may be multiple software options for one physical configuration, s . This has to be determined empirically and added, giving the new formula ([Eq. 3]):

$$K_m = \prod_{i=1}^j (p_i + s_i) - q \quad [\text{Eq. 3}]$$

For instance, an example RMT has three reconfigurable aspects, two are modular – one with four modules and the other with three – and the other is a movable tool with three discrete positions. The machine has two impossible combinations because of physical limitations. This implies the multiplication of 4 times 3 times 3, for each of the three reconfigurable aspects two is then subtracted from the product for the impossible combinations. This gives the number of possible physical machine configurations as 34. In this case, software configurations were ignored, this would increase the number of reconfigurations, further showing the flexibility of a RMT.

The number of configurations, K_m , that a Dedicated Machine Tool (DMT) or traditional machine can adopt should be taken as 1, unless some software changes can be made, in which case the product of the sequence is equal to 0 and the number of impossible configurations must also be 0, thus K_m is the number of possible software configurations:

$$K_m = s \quad [\text{Eq. 4}]$$

A CNC machine can be configured using software (and usually a number of hardware changes) and may have a practically infinite number of configurations. This is not a practical number to use and true CNC machines are not common in RMTs. For this reason the number of configurations for a CNC machine should be taken as 1 or as the number of substantial hardware changes that it is capable of (in which case it may be considered an RMT and should use [Eq. 4]).

Thus the number of possible configurations for an RMS should be a combination of the number of configurations for each machine, K_m , and the number of factory floor layout configurations, K . This gives the contracted form of the factory floor state, S_{ff} :

$$S_{ff} = K \prod_{i=1}^N K_{m_i} \quad [\text{Eq. 5}]$$

This formula indicates that the number of possible configurations becomes truly massive – using the simple example in the research on factory floor configuration [12], the number of configurations for 7 machines is taken as 15. If each of these seven machines has a number of

configurations equal to: 34, 1 (a DMT), 4 (a software changeable DMT), 8, 6 and 18 then, according to the equations here, the number of possible factory floor states, S_{ff} , is 616890. This is well beyond what will ever be used, but proves the huge potential of the RMS paradigm and the need for a form of technological way of mapping these different states without the need for human information gathering.

2.8 Research Gaps

It is a goal of RMS to have a rapid reconfiguration and low ramp-up time [1, 29] but this is hindered by the lack of research into the autonomous reconfiguration of software, or control of machine controllers on the factory floor. It is currently done using manual methods involving human labour and time. Thus, there is a problem with the RMS paradigm – ramp-up is slow and tedious. This research is necessary in order to speed this process up and improve and insure the overall feasibility of the RMS paradigm. There is currently a gap in the research covering the re-establishment of control after a reconfiguration.

There has been research into the intelligent reconfiguration of an RMS [30, 31], but this research did not address how the information about the current factory floor state is gathered and provided to the decision making software or person or how new physical configuration information is gathered to assign new programs to machines. There was a clear gap in the research covering how to detect a change on the factory floor, both in positions of the machines (relative to each other and to the factory floor and materials handling system) and in configuration of the machines in the system (both in hardware and software). In other words, there was a gap in the research covering the gathering of factory floor *state*. The decisions made by the intelligent software must be based on the current state of the RMS and once reconfiguration has been achieved, the new state of the factory floor must be known by the factory controller in order for a new Manufacturing Execution System (MES) iteration to be implemented. Manual gathering of this information is possible, but it is a very tedious, labour intensive and time-consuming process which may be prone to human error.

There has been research into middleware for holonic control and communication with heterogeneous machine controllers, but it has not been widely studied. A gap also exists for the implementation of a more simple method, light on software which can be made more universal and be adopted by any RMS. The reconfiguration of control programs on the factory floor is implied by this research into middleware, but its implementation is complex and not discussed in detail in the cited papers. It is a secondary goal of this research to provide a simpler and more universal means of ‘middleware’ to deal with heterogeneous communication.

It proved to be a reasonably simple calculation but even so, there existed no published literature into the enumeration of states possible for a RMS.

2.9 Problem Identification in the Current State of the RMS

This section discusses insufficiently addressed problems which are part of RMSs, which this research aimed to solve. According to Koren et al [1] the RMS paradigm is based on five fundamental principles (see section 2.2). These characteristics strongly draw on factory layout reconfiguration, modularity and the need for RMTs (among other things), all of which have their own inherent complications to ramp-up speed. The factory reconfiguration problem will be discussed first, followed by problems raised by the need for modularity and by the use of RMTs in the RMS.

Reconfiguration in the paradigm of the UKZN RMS project must involve physical reconfiguration of machine positions on the factory floor. Thus not only a change in the machines themselves and their materials handling systems, as in some previous research [32]. This concept of moving the machines in a system to aid reconfigurability is outlined in even the earliest research on RMSs [1]. This physical reconfiguration of the machine position on the factory raises an issue in production planning and implementation. Building of a factory floor model, including the configuration should be based on physical data and not simulation. The future of RMS may involve more automated systems which need to base decisions on actual layouts. High ramp-up speed is a fundamental concept of RMS [1, 3, 12, 33] and thus the time taken to physically build a new configuration and to reconfigure the control elements should be reduced as far as is possible. Another issue with a dynamic factory floor is the reconnection of machines to the control network. This process can cause delays as some communication may be difficult to re-establish as access points are changed with the physical position change.

The need for modularity and the fact that most factories are not built as all-new entities and do not use only one brand and series of machine and machine controller brings up an issue of heterogeneity on the factory floor. The various brands, ages, types and complexities of the machine controllers are unlikely to be compatible with one form of software run by the main controller [10, 34]. This makes it difficult to communicate the new configuration to the machines and to read the current configuration of the machines.

The use of RMTs is fundamental to the RMS paradigm and is a major source of complication in RMS modelling and planning [35]. The number of possible factory floor configurations grows massively with each RMT added to the system (and even each non-reconfigurable machine and CNC machine). If the new set-up of each machine has to be manually entered into the MES and the new factory floor map has to be programmed in to the MES it dramatically increases the RMS ramp-up time. This is the current method employed and is a problem which needed to be addressed in this research. A form of automated factory information gathering system would solve this problem and help to make the ramp-up after a reconfiguration more smooth and rapid.

RMTs require modular control to implement the control changes which arise with a change in the physical makeup (configuration) of the machine [8, 25]. This can be implemented by reprogramming each machine with the new suitable control program each time a reconfiguration takes place. Another method is for all the relevant programs can be stored on the machine controller and it can be given a signal to change between these programs when a reconfiguration is undertaken. The second alternative should make for a faster ramp-up time, though the initial controller set up will be longer and more complex and it will only be truly applicable to cyclic-type or known customisation.. Both these methods require communication with the machines, with the second alternative requiring less data-intensive information communication.

2.10 Decomposition of the Identified Problems

Moving machines around the factory floor allows the reconfiguration options to broaden significantly over fixed-position RMSs. A dynamic factory floor layout also allows for new machines to be added to the system or for machines to be replaced without 'breaking' a fixed layout. The dynamic factory floor concept allows for efficient materials handling programs to be implemented, whereas a fixed factory floor layout requires much more complex materials handling to achieve the same level of reconfigurability. The fixed factory floor model is, however

more simple from a control perspective and more customised [32] and can base its reconfiguration on a simple map. A dynamic factory floor type RMS may have a complex layout designed by the intelligent system or engineers dictating the new configuration. In a large factory with many machines, creating a computerised map for each of the machines using traditional human measurement methods and ensuring its accuracy would be time consuming and slow the ramp-up process.

A reconfiguration process is very likely to also require the changing of physical make up as well as the control program on the RMTs in the factory. Dedicated, CNC and flexible machines in the RMS will also need changes to their programming. After the process of gathering the factory floor state, the RMT configuration or other machine must still be entered into the MES so that the new program can be chosen.

Modularity is another of the key characteristics which gives a high level of reconfigurability [1]. This characteristic, however, brings with it certain difficulties and complexities. Modularity can be considered on a factory level and on a machine level:

- ♦ On a machine level the modularity of a machine means that the machine can be changed to meet the requirements of a new factory configuration requirements [8]. RMTs may be modular in a variety of ways and be capable of dozens of different distinct configurations. This presents the same problem as described above – recording and programming of the configuration of each machine can be a time-consuming and an error-prone process.
- ♦ On a factory level, each machine can be considered a module, capable of being changed, moved, removed, replaced and re-introduced to the factory. Thus the factory floor map is a description of the modular state of the factory. If the state (position and configuration) of each machine on the factory floor is taken and combined into a single RMS model, this will describe the RMS *state*. The RMS state is used to make intelligent decisions about the course of action and software requirements of the RMS.

2.11 Chapter Summary

This chapter's literature review discussed the RMS manufacturing strategy and the management, control and technological strategies used to implement it. This chapter also expanded on the current knowledge of enumerating factory floor states to include a method suited to RMSs containing adaptable machines. Research gaps and problems associated with the implementation of RMSs were identified and discussed in this chapter. These problems and research gaps were primarily surrounding the heterogeneity in control and the ramp-up of the system after a physical reconfiguration. It was these problems that the rest of this research would focus on.

3. Chapter 3: A Conceptual Solution

3.1 Chapter Introduction

This chapter covers the conceptualisation of a solution to problems identified in the previous chapter. First, system requirements are outlined, then a general strategy is developed. The general strategy is made up of many constituent aspects, each of which required further research. The research into each of the design facets are documented and will influence the final design decisions. The mechatronic design approach is one which integrates different disciplines of engineering from the conceptual stage. Specifically, Mechatronics aims to combine electronics, mechanics and information technology design aspects into one holistic view, which aims for the seamless integration of different disciplines. This amalgamated approach to design was used throughout this research, developing the solution from concept to reality with all stages being guided by the continuous assimilation of subsystems. The mechatronic design approach is ideally suited for design use in the RMS paradigm when compared to traditional approaches because of the fundamental multi-disciplinary nature of the manufacturing technique.

3.2 System Requirements and Specifications

The specifications shown below (Table 1) dictate the needs of a customer that would use the RTLS and the specifications which arise from those requirements, as well as the needs required by certain legal and industry standards. The derived and performance specifications in the table contain foresight into specific information found later in this work and should be considered accordingly.

Table 1: System Specifications

Definition of System Specifications			
Specification Type	Attribute	Requirement	
Customer Specifications	Factory floor size coverage	50m x 50m and Scalable	
	Machine quantity	Totally Scalable	
	System success rate	Minimum 80%	
	Operation time	<1h for 8 Machines	
	Power compatibility	Must be able to accept standard machine controller voltage	
	Communication compatibility	Able to communicate universally with machine controllers	
Required Performance Specifications	Battery capacity	Must be able to operate on battery >72h	
	Wireless range	>50m Point to Point	
	RTLS accuracy	±600mm	
	Clustering accuracy	Able to cluster data varying by ±600mm. Non-Spherical clusters	
	Clustering robustness	>90% Accuracy up to ±750mm	
	Program assignment accuracy	100% Success rate for match finding	
Derived Specifications	Logic step-up and step-down	From and to 5V, 12V, 18V and 24V	
	Distance estimation accuracy	(-250mm;+1250mm)	
	RTLS consistency	Must fall within ±600mm 80% of the time	
Prescribed Specifications	Attribute	Requirement	Standard
	Wireless band	Must be unlicensed	ITU-R 5.138; 5.150; 5.280
	Wireless physical and MAC layer	Must be an established standard	IEEE 802.15.4/a IEEE 802.15.1 IEEE 802.11 ISO 18000-7
	Environmental concerns	Parts to meet international standards	RoHS 1

3.3 Aim and Objectives of the Concept

The solution developed in this research has the primary aim of making the ramp-up process of a RMS after a reconfiguration faster, less prone to human errors and less labour intensive. Several objectives were needed to make this aim a reality and these are listed below. Methods had to be developed to:

- ◆ determine the factory floor state;
- ◆ develop the state into a model;
- ◆ determine the appropriate new program structure;
- ◆ complete the reconfiguration by setting the new program on each machine controller;
- ◆ communicate software information to heterogeneous controllers;

The conceptual system aims to reduce the ramp-up time of a RMS after a physical reconfiguration has been completed and to do this with minimal human guidance. Thus the system to be developed had to be one which manages and automates software reconfiguration using gathered physical information. Two options for the basic solution idea were considered; a traditional software middleware layer concept and a hardware-supported, thin software layer concept (a modular, mechatronic solution).

3.4 A Software middleware Solution

The traditional middleware concept for communication has been investigated in recent research [10, 36] and works on gathering information from the machines by detecting machine configuration from machine controller and communicating this information back along the primary communication line. The core concept of these research papers is the use of existing, wired communications and using software (traditional middleware) to overcome heterogeneity issues. This primary line of communication is the same that would run the SCADA and MES control network. This communication does bring about some problems, because raw data from heterogeneous machines must be communicated via a middleware system so that sense can be made of the information – in order for that heterogeneous data to be interpreted homogeneously. This is an example of dealing with the multi-agent heterogeneity using software methods. Middleware can also be used for the active control of a RMS using modern Holonic methods [10]. For this technique of RMMS the problem of locating the machines on the factory floor to establish state information would still require a form of Real-Time Location System (RTLS) module. Depending on the type of RTLS technology implemented, the module would have to communicate the position of the machine via its controller and middleware to the main controller or the location software would be housed with the central RMMS controller and communicate locally with higher-level control software. The positioning techniques that are available are discussed in detail in a later section (3.6.2).

3.5 A Thin, Hardware-Supported Middleware Solution

The modular concept was designed to remove the need for a complex middleware layer in the network, replacing it with a mechatronic hardware solution to the middleware problem. The concept, named the Reconfiguration Management and Middleware System (RMMS) was for a microcontroller-based module with network communication ability, a hard-line communication link with the machine and position detection capabilities. The module was envisioned to attach to the machine and provide a solution to the problem of gathering of information about the factory floor state and communicating with heterogeneous entities. The state is required after a

reconfiguration operation to provide information needed in the software reconfiguration process. Positioning technology could also be included in the module; it could communicate its position via the module if it is of that type, otherwise the module can just house the tag for the technique using a RTLS hub, as discussed briefly above.

Communication could have been achieved by using wired communication such as Ethernet or Profibus or using a wireless protocol such as WirelessHART, Zigbee or DASH7. The use of these modules is a basic form of multi-agent system where information is shared between the modules and the controller and machines, allowing intelligent operation [37]. Each of the modules is its own simple agent, gathering and sharing information about machine state and passing on reconfiguration instructions from the controller to the machine. Because of the novel nature of this RMMS concept, the decision was made early to make use of this hardware solution in this research. For this reason, the following sections further review the literature, based on the use of a modular hardware reconfiguration management middleware system.

3.6 Technology Review and Breakdown of RMMS

Figure 5 displays the fundamental differences in the two approaches to middleware management, the software-based and the thin, hardware-supported method of middleware management.

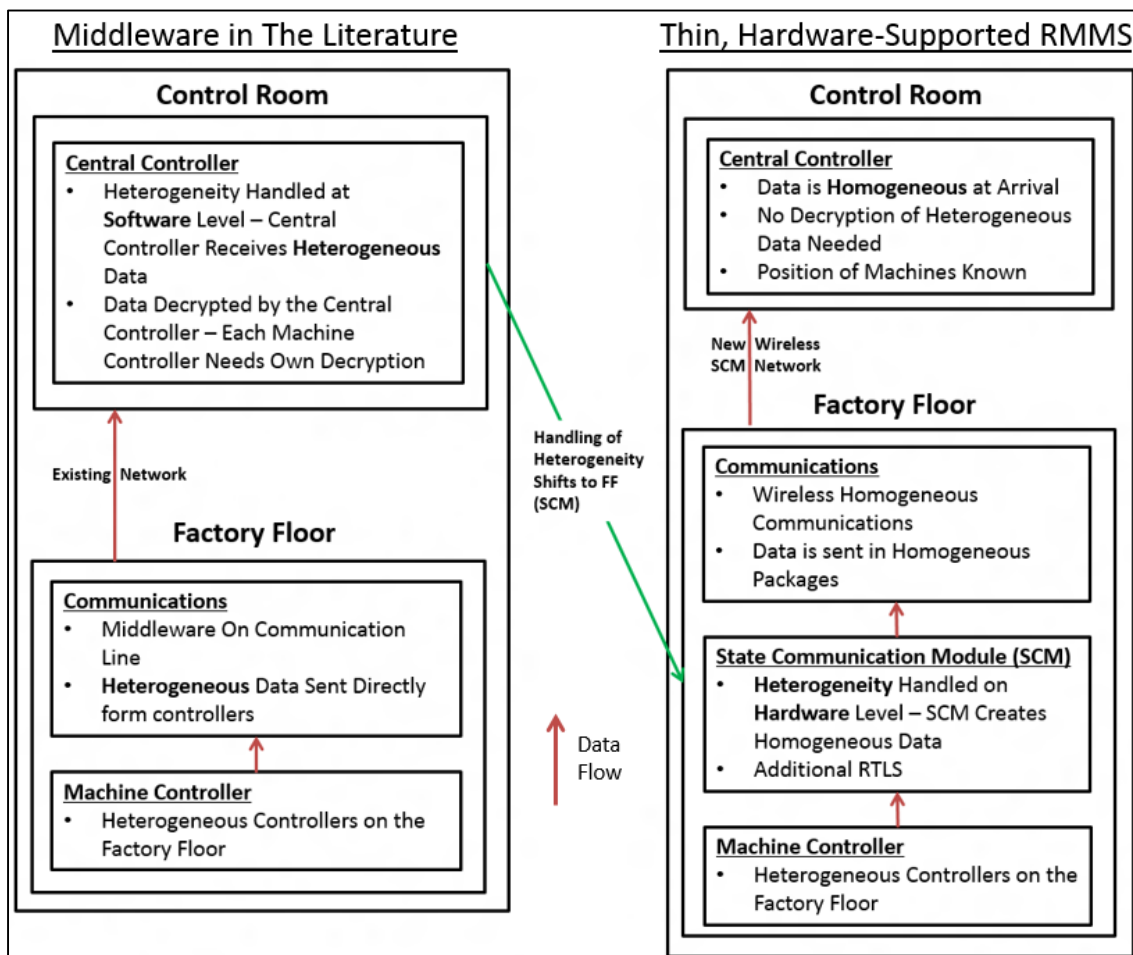


Figure 5: A Comparison of Literature MMS and the Proposed Concept

The proposed RMMS module, the State Communication Module (SCM), will need to contain a microcontroller (MCU) for processing and a technique for communications with the central controller and with its machine controller; and it must house a type of position sensing device.

There are a few options which were considered for further development with multiple concepts in each of the four main aspects of the module's design:

- ◆ controller communication techniques;
- ◆ positioning technique;
- ◆ machine communication technique;
- ◆ physical design;

The research into the state of the technology for the techniques considered in each of these aspects of the design is outlined in the following text (sections 3.6.1 to 3.6.7).

The following characteristics are required by the RMMS from the hardware module:

- ◆ Provide rapid reconnection after reconfiguration.
- ◆ Detect the configuration of a RMT/machine after a change.
- ◆ Detect the position of the machine.
- ◆ Communicate via secure, reliable protocol – wired or wireless:
 - communicate the configuration of each machine to the central controller;
 - communicate the position of each machine to the central controller;
 - receive software reconfiguration instructions from the central controller;
- ◆ Provide software reconfiguration information to the machine.

The central controller, which was designed to interact with the abovementioned module, had to be capable of the following:

- ◆ instruct the modules to gather factory floor data;
- ◆ receive factory floor data from the modules;
- ◆ use the data to produce a model (using a type of intelligence system);
- ◆ use the data and model to assign appropriate programs to machines;
- ◆ send the software reconfiguration information to machine controllers;
- ◆ interact with a human user;

3.6.1 Communication in the Network

The literature contains a large amount of research into multiple different communication techniques ranging from simple wired protocols to Internet Protocol (IP) based wireless methods. Each has its pros and cons and initially a comparison was made between wireless and wired communication. Table 2 summarises findings in the literature [38-42] and is specific to an industrial environment:

Table 2: Comparison of Wired and Wireless Communication

Wired vs Wireless Communication		
Attribute	Wired	Wireless
Reconnection Speed	Low	High
Data Transfer Speed	High	Low-High
Security	Very Good	Good
Reliability	Very High	Improving, Fast becoming as High
Compatability with Current Installation	Low	Independent of Other installations
Cost	Low - Very High	Medium

The *reconnection speed* of a wired network may be slow after a reconfiguration because wires will have to be physically plugged in before a connection can be made to the module. If made wireless the module can be powered by a battery and thus effectively never lose communication [43, 44]. Even if power is lost the connection time for most wireless protocols is very low [41]. The *data transfer speed* of a wireless network is totally dependent on the protocol and hardware in use, though in general a wired network will be faster, the speed of most wireless protocols is far beyond what is required in a factory environment [42, 45, 46], especially for requirements of the situation at hand. For this RMMS specifically, practically any wireless network will have sufficient data transfer speed. *Security* for a wired network is better than for a wireless network in general [47], but newer wireless protocols are sufficiently secure for industrial applications [43, 44, 48, 49]. *Reliability* is crucial in an industrial setting and because of the mature nature of wired communication it has very good reliability. Wireless networks have an improving reliability, with newer industrial-specific protocols recording very high reliability (> 99% data success) [44, 49]. Compatibility with current installations can be a problem for wired networks, if the network is not of the same type as the currently installed protocol then an entirely new network; including wiring will need to be implemented. This will also affect the *cost*, if the protocols are compatible then the cost will be low, if not it will be very high. A wireless network will not have to be compatible with the current communications strategy of the factory floor as it will use its own protocol no matter what. The cost is fixed (per unit installed) for a wireless network because it is not governed by compatibility and this cost is less than for an entirely new wired network. It was for these reasons that the research and review was continued only for wireless protocols and not for the wired options. Because of the extra complication involved in the design or modification of an appropriate industrial wireless protocol was deemed well beyond the scope of this project (it would make a research project on its own), it was decided that an off-the-shelf protocol could be implemented.

Because the proposed RMMS is envisioned to be implemented internationally and to be used by many unlicensed engineers, it must be designed in one of the bands which are unlicensed in most countries. These common unlicensed band ranges are summarised in Table 3 [50-52] :

Table 3: Common Unlicensed Bands

ISM Band	Range (MHz)
433 MHz	433 - 434
900 MHz	902 - 928
2.4 GHz	2400 - 2483.5
5.8 GHz	5725 - 5850

Electromagnetic Interference (EMI) and Radio Frequency Interference (RFI) cause problems with wireless communications in a manufacturing or industrial setting. In high power applications there is a lot of interference between 10 KHz and 50 MHz due to power and signal lines [53]. Variable frequency drives in an industrial setting emit strong interference below 1 MHz, but as frequency increases, the level of interference decreases [54]. Unprotected microwave devices, which may be present in a factory, can also cause major interference in the 2.4 GHz band [55]. The 433 MHz band is not commonly used outside of amateur radio control, remote door openers, home alarms, and RFID, so interference from other radio frequency users is unlikely to be an issue in an industrial setting [56]. Use of cell phones creates interference at 1.8 GHz and on some networks 900 MHz [57, 58]. Television broadcasts create significant interference in the low 600

MHz range, but these bands are licenced and thus could not realistically be used in the proposed system.

In general, the characteristics of a signal depend on broadcast power and frequency; the following table of characteristics (Table 4) corresponds to signals of the same broadcast power:

Table 4: Comparison of Frequency Characteristics

Frequency	Indoor Penetration	LoS Range	Data Rate	ISM Frequency
Low	High	High	Low	433 MHz
Mid	Mid	Mid	Mid	900 MHz
High	Low	Low	High	2.4 GHz

The following text reviews the available wireless communication methods available for use in an industrial environment. The table that follows the descriptions (Table 5) shows a summary of the basic characteristics of common and potentially applicable wireless communication protocols [40, 42, 48, 59]:

Wireless Fidelity, commonly known as *Wi-Fi*, is the standard of wireless communication used in lap top computers and other IT related applications [60]. It has not been a success in industry, predominantly due to the technology's popularity and popularity of the 2.4 GHz frequency band as a whole. The popularity of Wi-Fi leads to interference from other users in the factory – IT wireless networks, security cameras and other sources. Wi-Fi is capable of high-speed communications but sacrifices power consumption to allow for this speed and its need to be constantly on, which is not desired in an industrial application. The price of Wi-Fi technology hardware is low in comparison to other protocols because of the worldwide popularity.

WirelessHART (Wireless Highway Addressable Remote Transducer) is an extension of the HART Communication Foundation's HART protocol [48, 59, 61]. The HART protocol is a popular communication protocol designed for use with standard 4-20 mA remote sensors. The added wireless capability has been included to allow for remote wireless sensors in a network. WirelessHART has been designed from the outset to be used in industry and thus includes a very reliable channel hopping and channel blacklisting scheme as well as a fixed 10ms Time Division Multiple Access (TDMA) technology to ensure that communications are robust. Even though WirelessHART operates on the unlicensed 2.4 GHz band, it has been designed to be agile and never drop communications. Also, the power that WirelessHART communicates with is low and thus it does not cause much interference with other wireless networks. It is designed to use a very small amount of power and has a comprehensive security protocol to ensure the privacy of the network [48, 61]. WirelessHART has a built in check to ensure that communications are delivered and will resend if it detects that a message was not received. The built in mesh capabilities of WirelessHART allow for a large factory floor area to be covered by the network without concerns about transmission loss. WirelessHART installations number in the high thousands of networks already [43]. WirelessHART does need an existing HART network to be very smoothly integrated and is thus not always compatible with current installations [49, 62]. Even though WirelessHART is an open protocol, the software stacks used for the implementation are complex and not readily available. The WirelessHART protocol requires a complex gateway unit to control communications, these are expensive and not easily designed and programmed in a research project – and thus would have to be purchased. Thus the only feasible option for

implementing this protocol is to purchase preconfigured evaluation kits from a manufacturer such as *Aiwatech* or *Linear Technology*. These pieces of equipment are very expensive and over specified for the application of the RMMS.

Bluetooth is a popular 2.4 GHz protocol used in mobile phones and input devices for computers. It was originally designed as an easy to use connectivity protocol for consumer devices [38]. It is also capable of being used in a mesh network to communicate data on an industrial level. Bluetooth technology generally only has an indoor range of $\pm 10\text{m}$ [63]. The mesh network nature of Bluetooth does allow it to cover the area of a factory, but the low range means that there have to be more nodes in the network and the heavy load on the mesh architecture can cause long latency times [55]. Bluetooth technology is established and has been used commonly in open source projects and thus the software stacks are readily available. Furthermore, there exist affordable preconfigured modules readily available in the marketplace.

Zigbee is a specification aimed at scalability and self-organisation which builds on the medium-access control (MAC) layers of the IEEE 802.15.4 specification. New versions are now capable of both mesh networking and peer-to-peer communication [46, 59]. Security in a Zigbee network is not top priority, but the security techniques are available for use in the MAC layer. Industrially, Zigbee has yet to be proven, but popularity and viability are growing with new version introductions. The lack of channel hopping is of concern to an industrial user of Zigbee, especially as the network becomes large [59]. Zigbee is not an ideal solution in terms of performance and may require a complicated mesh network, but the advantages are that it is very inexpensive when compared to other communications protocols and there is a large amount of literature and support on using Zigbee and modules can readily be bought preconfigured. An additional advantage is that, as will be briefly discussed below, Zigbee can also be used as a positioning technique. The ease of use, cost and capability for use as a positioning technique make it a feasible candidate.

The *ISA 100.11a* specification is one of the newest developments in industrial wireless technology. It runs using the same IEEE standard as Zigbee and WirelessHART, namely 803.15.4. Many comparisons to WirelessHART have been conducted because of the shared target market of industrial sensor networks. ISA 100.11a can be configured to work with Internet Protocol version 6 (IPv6), making the network setup process fast and reliable [64]. There has been a delay in development because of the high flexibility of the protocol and lack of a well-defined application [43]. ISA 100.11a supports high data transfer rates which are not necessarily needed for this SCM-based RMMS application, though this does not affect the power consumption, which is very low. The protocol is capable of a complex mesh network with multi-path ability, making large networks covering a big factory floor possible. The mesh nature also allows for low transmission power and which aids resistance to interference and aids low power consumption. The security built into the ISA 100.11a protocol is advanced and very secure and the communication robustness has been designed from the outset to be very high [62, 65]. The robustness is achieved through TDMA, a complex channel-hopping technique, channel black-listing and deterministic nature of the protocol. ISA 100.11a also needs a complex gateway unit like the WirelessHART protocol. This protocol is also meant to operate using an existing HART installation, though it can be configured to communicate using any message configuration [66]. The protocol stacks are open-source, but few pre-installed modules are currently in the

marketplace and available products are specialised and expensive, making implementation for the purpose of research experimentation implausible.

DASH7 is an advanced and emerging 433 MHz wireless communication protocol developed by the Dash7 Alliance using the RFID tag specification ISO 18000-7 (18000DASH7). The ISO standard was originally intended for military use but has been repurposed [67]. The distinct advantage of using DASH7 as a communication technology is that it was designed from the outset to be used for both localisation (as will be discussed later) and communication. The low frequency nature of the DASH7 technology allows it to have very high penetration through solid objects and therefore a better indoor range than the other techniques discussed here [40, 67]. DASH7 has one of the lowest claimed power consumption of the protocols listed in this section [44, 67] because of its ability to sleep when not in use. DASH7 is secure, with encryption and the ability to be configured with IPv6 and other features [67]. An issue, however, is that DASH7 modules are not readily available and the speed of prototyping would be adversely affected.

Table 5: A Summary of Industrial Wireless Possibilities

Wireless Protocol Comparison									
Protocol	Frequency	Standard	Data Rate	Indoor Range	Mesh Enabled	Co-existence, Security & Robustness	Power Use	Simplicity	Cost
Wi-Fi	2.4GHz/ 5.8GHz	802.11	High	Low	Yes	Low	High	High	Med
WirelessHART	2.4GHz	802.15.4	Low	Med	Yes	High	Low	Medium	High
Bluetooth	2.4GHz	802.15.1	Med	Low	Yes	Low	Med	Medium	Low
Zigbee	2.4GHz/ 900MHz	802.15.4	Med-Low	Low/Med	No/Yes	Low	Med	Medium	Med
ISA 100.11a	2.4GHz	802.15.4	Low	Med	Yes	High	Low	Medium	Med
DASH7	433MHz	18000-7	Low	High	No	High	Low	High	Med

3.6.2 A Real-Time Location System

There has been much research into various positioning or localisation techniques, these pieces of modern research summarised in a dissertation by Mautz in 2012 [68] (a version this work was also published as a journal article [69]). From this research, the following summary (Table 6) was compiled.

Two options are realistic in the RTLS technique based on the requirement constraints for the RMS paradigm application of RTLS. Either the technology used to detect the position of the modules must be very accurate ($\pm 1\text{cm}$) or, a less accurate ($\pm 1\text{m}$) technique could be used along with an intelligent logic algorithm. This algorithm should be able to produce a realistic and accurate model based on less accurate data. It is clear that the RTLS technique needs to have a range capable of covering a factory floor ($>50\text{m}$).

The more accurate method is more desirable because it is much more reliably accurate and less prone to errors. Also, this system would be easier to implement. The caveat to this is the greatly increased cost of a system capable of accuracy to the order of cm and with a range large enough to cover a factory floor.

Table 6: Summary of Localisation Techniques

Localisation Technology Comparison			
Technique	Accuracy	Range	Cost
Cameras	um-mm	m-10m	Med
Infrared	cm-dm	m	Med
Tactile and Combined Polar Systems	um-mm	m-km	Very High
WLAN/WiFi	1-10m	10-100m	Low
Radio Frequency Identification	dm-m	m-100m	Low
Ultra-Wideband	cm-m	m-100m	High
Pseudilites	cm-dm	10m-10km	Very High
Other RF Technology	dm-100m	m-100m	Low-Med
Inertial Navigation Systems	10m-km	>10km	High
Magnetic Localization	mm-cm	m-10m	Low

Tactile and Combined Polar Systems (TCPSs) work in a similar manner to an outdoor Global positioning System (GPS), but use stationary beacons installed in the factory. They are extremely accurate and can cover huge areas [70], but the system cost would make the proposed module system practically inaccessible and much less attractive to RMS designers. Ultra-Wideband (UWB) systems are very accurate and far more suitable for the application [71]. UWB systems are much more financially viable than TCPSs but both these methods are still too expensive for the application at hand.

Using a less accurate positioning technique along with an intelligent technique to process the raw data into a usable factory floor model does involve more work than the more accurate techniques listed earlier, but it has the advantage of being far more cost-effective. Using the concept of heuristics and raw location data allows for the use of Pseudilites, Radio Frequency Identification (RFID) and other Radio Frequency (RF) Techniques for gathering location information. These technologies are more financially viable and should provide a more attractive product to industry. Further investigation has been done into new RF technologies which include Received Signal Strength Indication (RSSI) or Time of Arrival (TOA) based techniques to gather location information.

Zigbee and DASH7 technologies have both been developed into reasonably accurate and very cost-effective positioning technologies previously [45, 72]. This dual-duty of the wireless technology lends additional credit to the use of these technologies as the communication technique for the SCM-based RMMS. By using the same hardware for the purpose of communication and localisation the cost of the module is decreased and the feasibility is increased. Both these methods incorporate RSSI from multiple stationary transmitters and a computer algorithm to determine the location of a module. Because of the short processing time periods at hand, the measurements can be taken multiple times to ensure accuracy.

3.6.3 Further Research on RF-Based RTLS Strategies

Using the received signal strength from a RF communication source to locate a node is a well-documented method of judging distance [72-74], though the effectiveness of the system varies. The general strategy is to use the strength of a received signal to estimate the distance to a series of fixed 'beacons' and to estimate a position based on the combination of strengths, the majority of the cases considered are variations of the familiar triangulation technique used by the GPS. All methods of RSSI RTLSs work on this basic premise, but the techniques differ in the method used

to estimate distances and calculate the position based on those estimated distances. This subsection reviews some of the more popular methods and comments on their relative feasibility.

Other methods are available for the judgment of distance between two units (a node and a beacon), including time of arrival (TOA), time difference of arrival (TDOA) and real time of flight (RTOF) [75]. This means that any of the range-based algorithms discussed below can be used with RSSI (as is used in the descriptions given) or with another method of distance estimation. These other three methods all work on the premise of synchronised time between two modules for which the gap distance is needed or for there to be a rebound calculation including processing time for the return signal at the distant module [75]. They work on the basic formula:

$$distance = speed \times time \quad [Eq. 6]$$

Where the *speed* is the speed of propagation of an electromagnetic wave in air, which is taken as c , the speed of light, and *time* is the time measured for the signal to travel between boards. This clearly requires highly exact synchronisation of the involved modules' clocks (in the case of TOA and RTOF) and highly accurate measurement of time – note that a flight of 10m will take approximately 33.3ns, which is less than one clock cycle on a 16 MHz processor (typical of an MCU). With the use of high-specification MCU (84 MHz), with a clock cycle of 11.9ns, a measure of accuracy could be obtained. The method which found accurate (<16cm) readings based on TDOA used a very fast processor with highly consistent wireless modules (250 MHz), these are available, but are highly priced. The use of high-specification MCUs can help the accuracy of the system. This method was deemed impractical for the research, however, and was not investigated further. The feasibility that will be gained in the use of TOA type distance measurement may be a welcome addition to this research, but will only replace one small part. Due to the high specifications required for the MCU and restrictive costs associated the research further (from the next paragraph) in this paper follows the premise of the RSSI based RTLS.

Consider a case with i beacons of set known position (x_j, y_j) for $j=1, 2, \dots, i$ and a remote wireless module of unknown position. When the position is required, the module sends a request to each of the beacons. In turn, each beacon responds with a signal containing its position with a signal of regulated intensity. The module receives this signal at a certain intensity and the level of the signal's intensity allows the module to estimate its distance from each beacon [76, 77].

When the distance to a source (r_j) is known, along with the distance at which the reference power was measured (r_0) the most commonly accepted model to predict received power (p_j) is shown below. The signal is received from a distant source, with p_0 as the reference power and n as path loss (typically two to four, though variances can occur) [77].

$$p_j = p_0 + 10(n) \log\left(\frac{r_j}{r_0}\right) \quad [Eq. 7]$$

It is known, from basic communications, that the received power (in dBm) (P_k) can be calculated by considering the received power (in mW) at that point (P'_k) and at a reference point – or the output power at the source – (in mW) (P'_k) [77].

$$p_k = 10 \log\left(\frac{P_k}{P'_k}\right) \quad [Eq. 8]$$

Now the received power at a distance may vary for many reasons including channel fading, temperature and environmental effects and other forms of interference thus there will be a different received power (\hat{p}_j) which will come from the distance that the module ‘reads’ the beacon to be away from it [77]:

$$\hat{p}_j = p_0 + 10(n) \log\left(\frac{\hat{r}_j}{r_0}\right) \quad [\text{Eq. 9}]$$

Now, to combine the following equations is to find the ‘read’ distance which the module is predicted to calculate [77]:

$$\hat{r}_i = r_i 10^{\frac{x}{10n}} \quad [\text{Eq. 10}]$$

The understanding of the effects of these algorithms is essential in the consideration of the algorithms and which to choose on a theoretical level.

Research has been conducted by Zanca et al. [78] and others (cited in text) on different RSSI-based localisation algorithms, the following is a summary of their work and a discussion on the suitability of each of the algorithms for the RMMS for RMS. The subsequent paragraphs briefly discuss many of the available algorithms described in the literature.

Min-Max is popular localisation algorithm, used for RSSI and other localisation data types, predominantly because of the simplicity of its implementation. Modules of unknown position gather an estimation distance from each of the beacons. The module then uses this information to ‘draw’ a pair of vertical and horizontal lines around each beacon so that the least distance between each beacon and the line represents the estimated position of the module. When each set of lines is ‘drawn’ a rectangle will be formed around the probable position of the module – the smaller the rectangle the better. In an ideal system, the module should be in the geometric centre of the rectangle.

The *Multilateration* method is a geometry-based decentralised localisation algorithm. In a similar manner to the above technique, the algorithm detects an estimated distance from each beacon. It then ‘draws’ a circle with a radius equal to the distance and centred on each beacon. In a perfect situation the circles will intersect on a single point, but in reality there will be an area of higher-density intersections and the module can be estimated to be in the centre of that area. This method works in a similar way to Min-Max and is theoretically more accurate, but more complex. This method is very popular in other literature [79, 80] and in the form of other slightly modified algorithms. Oguejiofor et al [80] suggested an accuracy of 0.56m for a 4 beacon Multilateration method – this is very promising for the SCM-based RMMS development.

ROCRSSI stands for Ring Overlapping based on Comparison of Received Signal Strength Indication; it is considered a range-free algorithm and relies on the decreasing power function of signal strength with distance (it is not fully range-free, but negates environmental fluctuations in range estimations). Every beacon must periodically broadcast a RSSI vector including each of the other beacons’ signal strength. The module receives and stores these vectors and when it has multiple readings it will start calculating its position. For each beacon the module will compare its RSSI from the reference beacon to the RSSIs between the reference beacon and other beacons in the network. It then determines, by comparison of the known position signals (from beacons), how far away the module is from the reference beacon. It does this for every beacon, ‘drawing’ rings around each beacon and thus determines an estimated position. This could be used as an

adaption to the multilateration method, where there is no direct range calculation, but rather radius estimation based on the relative strength of the module signal in comparison to signals from beacons which are a known distance away from the reference beacon. In certain cases, this method may prove a useful alternative to multilateration.

Research from Blumenthal et al [72] introduced the *Weighted Centroid Localization (WCL)* where firstly each beacon sends the module their position, and secondly the module calculates its position by a centroid determination from which a derived function is applied to convert the data into a weighted value, giving a reportedly better localisation. The centroid determination works by averaging estimated co-ordinates in a similar manner to multilateration.

Nguyen and Rattentbury [81] proposed a method called the *Regularised Gradient-Based Least Square (RGLS)* method, which makes use of the phenomenon of RSSI overestimating small distances and underestimating longer distances. This method aims to use a better model for distance estimates – the paper recommends that this model is adapted, using certain tests, for the particular type of hardware used. In many of the investigated cases the function approximates a straight line. This method is practically the same as the multilateration method (and can be adapted to other abovementioned methods), but uses an empirical method for distance estimation.

The multilateration results in the paper by Zanca [78] do not give promising estimations because the overall resolution is very low when a realistic number of beacons (four to six) is supplied. An average accuracy or resolution of under a meter is desired for the application of the RMMS in this research. To support the use of multilateration algorithm, the following illustration (Figure 6) [78] shows how the localisation error is significantly lower (1m) when a module is in a position with beacons surrounding it rather than being outside of a ring of beacons. It is clear that it will be beneficial to have beacons located around the perimeter of a factory (and possibly somewhere on the floor in addition). This is convenient because this is also the logical and most realistic placement of anchor nodes and will likely give much improved accuracy in comparison to those quoted in the literature.

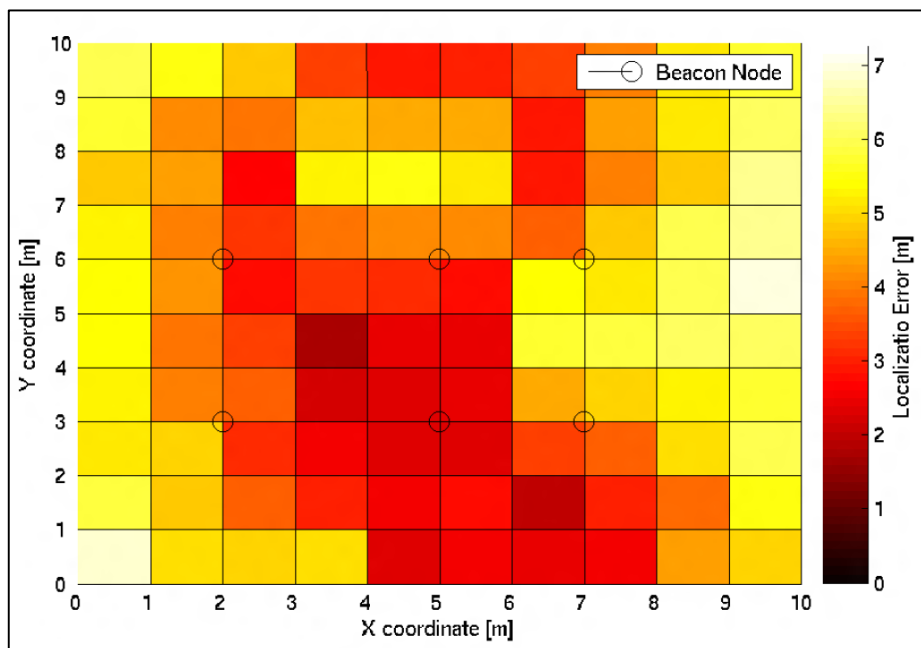


Figure 6: Localisation Error Change with Relation to Beacons

All the methods listed above are variations of ‘triangulation’ and all require a distance estimation based on signal strength indication (or possibly some other method) between fixed beacons and modules [74]. The following methods are based on other forms of RF-based localisation.

MDV-Hop and DV-Hop algorithms are range-free algorithms exclusively for use in mesh networks of approximately uniform distribution. They work by counting the number of hops in a path from a beacon to the node and using the average distance of one hop to estimate its position. It then can use any of the triangulation algorithms to estimate position. This method is an unlikely choice in a dynamic factory floor setting, where there is no guarantee of a meaningful average hop length [74].

Maximum Likelihood (ML) is an algorithm based on classical statistical interference theory, it is, like most of the others discussed before, a range-based algorithm. A vector of RSSI values from each of the beacons, \vec{p} , of coordinates \vec{x}_b & \vec{y}_b . The algorithm calculates the *a priori* probability of receiving \vec{p} for each position of the module. The position with the best probability is selected as the estimated position [74]. This method is complex to implement computationally (which is not ideal in a low-power microprocessor-based RTLS) and accurate when the number of beacons is high – which is not always possible in an industrial environment – and with a low number of beacons the algorithm is not very suitable.

Point in Triangle (PIT) algorithms, such as APIT [82], are range-free and are suitable for high-density networks. APIT algorithm detects whether a node lies within a particular triangle of other nodes of known position. When the smallest possible triangle of encompassing nodes is found, the approximate position of the lost node is found. This algorithm is not suitable for the application at hand, and is better used for tracking of a single node or finding a lost node.

The chosen method is identified in Chapter 5.3.1, where the choice is justified and the method is discussed in greater detail. The specific programming of the method is also discussed there.

3.6.4 Processing of RTLS Data

Data found by a suitable real-time location system needs to be processed in order to be useful for the RMMS software. This processing focuses on the ability to identify manufacturing cells based on the position of a module (attached to a machine) on the factory floor. Research was conducted on the suitable options present for this processing and their attributes. Hierarchical clustering, partitioning, grid-based clustering, and artificial intelligence methods are the common broad techniques that clustering algorithms are based on – the methods are discussed in the following text.

Hierarchical Clustering works on the concept of either allowing an initial state of singleton clusters to merge together or an initial single cluster of all points to split until a pre-determined stopping criteria is met [83]. The strategies used to generate mergers or separations vary greatly from algorithm to algorithm, but the premise of splitting or merging remains constant in all hierarchical methods [84]. The structure of a hierarchical clustering algorithm is often represented as a tree diagram or dendrogram where each end leaf or line-end is a point and joined points are part of the same cluster [83]. The stopping criteria will determine at which point the algorithm stops along the tree/dendrogram, and the highest remaining joins will be clusters.

Partitioning Methods for clustering include K-means methods, relocation algorithms, and density-based methods [83, 85], although K-means methods and density-based methods are

sometimes separated into their own categories. Due to the broad nature of these techniques, each of them will be discussed in the following text.

Relocation Clustering Algorithms start with an initial partition of the data based on an input containing a number of clusters into which the data should form. The algorithm then moves the clusters around and changes their size, depending on what rules are in place, iteratively until a best case is found. In general this method needs a small amount of user input, although more intelligent options are available which estimate the number of clusters and can change the estimate in case of an unacceptable output being found [85].

K-means Clustering is an iterative method based on Euclidian distance measure of cluster members about a K centroids made up of the mean of its points [84]. It is probably the most widely used clustering algorithm and produces high quality clusters in the right circumstances. The general K-means algorithm estimates spherical clusters about the centroid of its points and is thus useful for small uniform clustering [84]. It has many varied off-shoots and adaptations not documented here which perform better in each of their specific tasks [83].

Density-Based Clustering works by locating areas with a high density of points and assigns all points in that dense area to a cluster, which absorbs points near enough to its borders to be clustered [84]. This makes density based methods useful for detecting clusters of irregular shapes. These methods rely heavily on estimations of the scale of the cluster range to estimate density and must be configured for different environments [83]. A distinct advantage of density based methods is the ability to find clusters which are not linearly separable or spherical in nature.

Grid-Based Clustering Methods use both partitioning and hierarchical methods to cluster data, taking into account the density of the points to find relationships. The search space is given a finite grid pattern which is used to simplify the density calculations. It differs from other clustering techniques by concerning itself with the space around points rather than the position of the points themselves [83].

Artificial Intelligence (AI) can be used to improve other algorithms described earlier and to provide its own evolutionary methods for clustering. Evolutionary methods can be employed, for instance, to find the correct number of clusters for K-means or relocation clustering methods. Evolutionary methods tend to have a high computing cost and are thus not very popular. It is also an issue in some circles that these methods are not deterministic in nature and thus may give varying results given the same set of data, which is not the case with the other methods [83].

The chosen method is identified in Chapter 8.3, where the choice is justified and the method is discussed in greater detail. The specific programming of the method is also discussed there.

3.6.5 Homogeneous Machine Controller Communication

Two lines of communication are necessary between the proposed module and machine – from the module to machine controller and from the machine controller to module. Machine controllers can know the physical configuration of their machine and can thus provide a communication of that configuration (see section 2.5). The module must gather the state of the machine by receiving data from a machine controller which is aware of its physical configuration [8] and it must be able to communicate software reconfiguration instructions to the machine controller. It should be noted at this stage that although a Modular Mechatronic Control enabled RMT may know its hardware configuration, this is not sufficient information to know what program should be run in

the new production period – this depends on what product is being produced and thus on the overall state of the factory floor.

Considering a modular RMMS concept as discussed in the preceding sections, communication between the machine controller and the proposed module could be done in a bit-wise fashion or over a simple communications protocol such as RS232, UART or I2C. Most industrial controllers are Programmable Logic Controllers (PLCs) and are thus capable of reading High/Low signals on input pins [86-88], but in some applications this may require the purchase of an additional I/O module. Not all machine controllers are compatible with a standard simple communication protocol without the addition of a potentially expensive add-on module. Non-PLC industrial controllers are generally also capable of High/Low reading, making the option of bit-wise communication universal and cost effective in the vast majority of cases. Although many commonly available microcontrollers are capable of UART, RS232 and I2C, practically every marketed MCU is capable of digital bit-wise High/Low signalling and the MCU's that are capable of the other communication protocols are more expensive than those that are capable of only bit-wise communications [89]. Thus bit-wise signalling messages were designed and are discussed in sections 5.5 and 5.6.

3.6.6 Data Processing and Program Assignment

Once the RMMS software is aware of the physical state of the factory floor (as found by the RTLS and machine configuration gathering), the state processing needed to be made to turn it into a usable model. An amount software had to be included within the RMMS layer in order to produce usable functionality from gathered data. Two specific goals were required from the development of a processing system within the RMMS:

- ♦ The RMMS must use the physical data to produce software changing instructions and send them to the appropriate machines to complete the reconfiguration process. The software information is based on a database produced at an earlier stage.
- ♦ The RMMS must have the software framework needed to pass appropriate data to higher level software entities in order to update control and management modules with the new factory floor requirements and outputs.

It is this process of switching the machine operation programs is what makes the RMMS a management system. The switching of software routines without the need for constant human interaction improves the management ability of a traditional middleware layer in a factory management system. The software which has to make program assignment decisions based on the factory floor model needed to be built around some sort of Artificial Intelligence, the AI must be able to make quick, accurate decisions in a reasonably deterministic fashion. AI is a very popular modern branch of computing research and many powerful methods are available and have been developed into systems capable of handling different situations. *Genetic Algorithms (GAs)*, *Knowledge-Based Systems (KBSs)*, *Artificial Neural Networks (ANNs)*, and *Swarm Intelligence (SI)* methods are investigated in the following text.

Genetic Algorithms are a method of evolutionary computing which used a biology-inspired method of evolution. GAs represent the overall concept of evolutionary intelligence, even though other methods do exist [90]. A population of possible solution variables is generated in a random or pseudo-random manner in the form of a genetic string or string of genes which represents a solution to each of the variables in the problem. Mutation and selection processes are applied to

the population of genes to simulate evolution. The selection requires a pre-determined fitness function to judge how well the solution will perform. The majority of solution fitnesses can be judged by simulated performance or by some known rhetoric. Good choices of selection and mutation will cause the population to converge to a solution which is near ideal. This method is extremely powerful when correctly implemented, but is difficult to get right.

A *Knowledge-Based System* is a simple and well-proven form of artificial intelligence which is more simple to implement and more deterministic than some other methods of artificial intelligence. The system uses reasoning (an inference engine) and a base of known facts (the knowledge base) to solve a problem [91]. When coded for a specific problem set the KBS can be simply set up and changed, making it very suitable for the task at hand, with the only real disadvantage being the large amount of data needed in the knowledge base in order to get meaningful results [92].

An *Artificial Neural-Network* is a method of artificial intelligence which aims to emulate the structure of a biological brain by coding artificial neurons (also known as processing elements or units). The neurons use inputs and weights on those inputs in order to trigger a function. The weights on the inputs of the neurons need to be optimised by a known function and often another form of AI (such as a genetic algorithm) is used for this operation [93]. Neural networks are capable of ‘learning’ to handle problems of great complexity, especially when combined with powerful optimisation such as GAs or DE [90].

Swarm Intelligence is another method of intelligence, usually used in optimisation problem solving, which is based on nature. There are many forms of artificial swarm intelligence, including Particle Swarm Optimisation (PSO), Ant Colony Optimisation (ACO), Differential Evolution (DE) and many others [90]. These algorithms are highly effective at solving multi-dimensional optimisation problems [93], but are not necessarily applicable for the task of assigning programs to machines.

3.6.7 Physical and Electrical Design

The physical design of the module used in a RMS factory environment is influenced by the factors outlined above but is not defined in its entirety at this stage. The design cannot be properly defined without the proper attributes, which are defined in the section covering the design process (9.3). At this stage the conceptual physical design considered some feasibility constraints defined by the preceding research. If a module attached to an industrial machine is considered, which is capable of unpowered operation and wireless communication and possesses processing capability for a RTLS, the module should:

- ◆ be small and light so as to not interfere with the machine;
- ◆ contain its own method for voltage regulation;
- ◆ house a battery for power when the machine controller is offline;
- ◆ contain a microprocessor capable of serial or bit-wise communication;
- ◆ contain the components for a method of localisation;
- ◆ be resistant to dust and light impacts;
- ◆ be capable of voltage boosting and bucking for I/Os;

These constraints come from some basic needs for an industrial environment. The proposed module had to be capable of battery operation because power will be shut off for a reconfiguration

and modules must not lose communications during this reconfiguration. For this reason the module should contain a suitable battery.

Different installations will provide different power sources, most run at either 24V, 12V, 18V or, less commonly, 5V [86-88]. The module must be designed to be universally applicable and thus capable of operating on any of these voltage sources.

The module is designed for an industrial manufacturing setting and will thus likely be subject to dust settlement, vibrations and light impacts during reconfiguration and operation [94]. For these reasons the module should be contained in a sturdy housing and should not be vulnerable to dust. It is not necessarily required that the module housing should be waterproof, except in some particular cases and for this reason a waterproof option should be considered.

3.7 Chapter Summary

This chapter dictated the requirements of the system and introduced the possible solution concepts, from which a conceptual design was drawn. The conceptual design was made up of multiple aspects from different areas of engineering, each of which required further study. A review of the current state of knowledge in each of the constituent aspects of the conceptual system is documented in the chapter, without making decisions on the final design. This chapter showed the different routes available for the development of the working system and also identified which areas would need the most work.

PART B

This part of the dissertation covers the concept development described above into a working laboratory-scale system. The design procedure of the RMMS and each of its constituent aspects is described and experiments used during the design process are documented. This discusses the design procedure of the system and each of its sub-systems, down to the component level in the appropriate cases. Design decisions are motivated and other options and considerations are covered. Experiments which affected the design decisions were included in the next chapter and should be consulted along with the system design discussion.

There are five essential sub-systems in the RMMS:

- ◆ communication from the machine controller to SCM;
- ◆ communication from the SCM to machine controller;
- ◆ wireless communication between the RMMS control software and SCM network;
- ◆ wireless-based RTLS;
- ◆ central RMMS control software including a GUI, cell-formation intelligence, a database and SCADA considerations;

Each of these systems and their design processes are discussed in detail in this chapter. During the design processes discussed in this section several experiments were performed to validate hypotheses and optimise algorithms at a simulation level. These experiments are documented in section 10, which should be consulted during the reading of Chapters 4 to 9.

4. Chapter 4: System-Level Design and Functionality

4.1 Chapter Introduction

Before each of the aspects of the system and their designs are discussed, there should be a detailed description of the system and how each of these aspects interoperate in order to help the reader better visualise the requirements of the aspects. This chapter describes the Reconfiguration Management and Middleware System at the level of the system itself, stating the requirements of each of the aspects without divulging information specific to the design of the aspects themselves. This section covers the same principles discussed in section 3.5, but in higher detail and with further development in many of the aspects. Some design decisions are implied in this section but are not fully justified in the text – these are elaborated upon later in the sections that follow. The RMMS contains the following entities which must interact seamlessly:

- ♦ Central RMMS Control Software (CRC) (with GUI);
- ♦ higher Level Entities (ERP, MES, Interacting Engineers);
- ♦ State Communication Modules (SCMs);
- ♦ machine controllers;
- ♦ Beacon Positioning Modules (BPMs);

4.2 Control Position and Operation of the RMMS

The RMMS has software which runs as an application over the operating system of the central control computer and is thus not middleware in its traditional definition. It, however, functions in the same way as middleware, with the additional management functionality – the system could even be described as a device driver for the RMS. The system as a whole, including the hardware, communication links, and control software, slots into the control structure in the same position as traditional middleware. The following image (Figure 7) shows how the middleware, including its SCADA functionality fits into the software control structure of a factory:

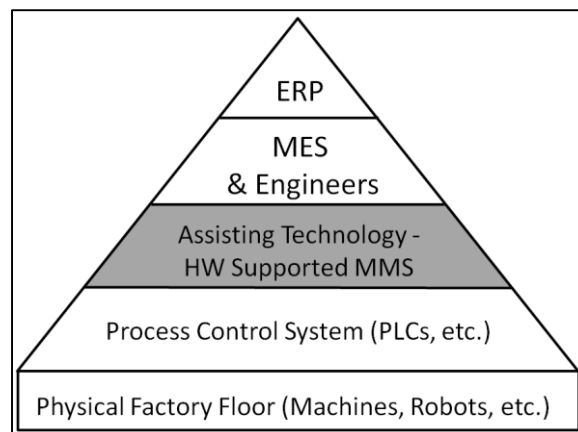


Figure 7: The MMS in the Control Structure of an RMS

Figure 8 shows the workings of the RMMS in the most critical of tasks – the handling of a reconfiguration. The RMMS must gather the factory floor state, associate it with a set of programs and instruct the machines in the RMS to switch to the correct programs for that configuration. This should all happen as one process, with no additional human interaction. A description is added after the figure to better explain the process.

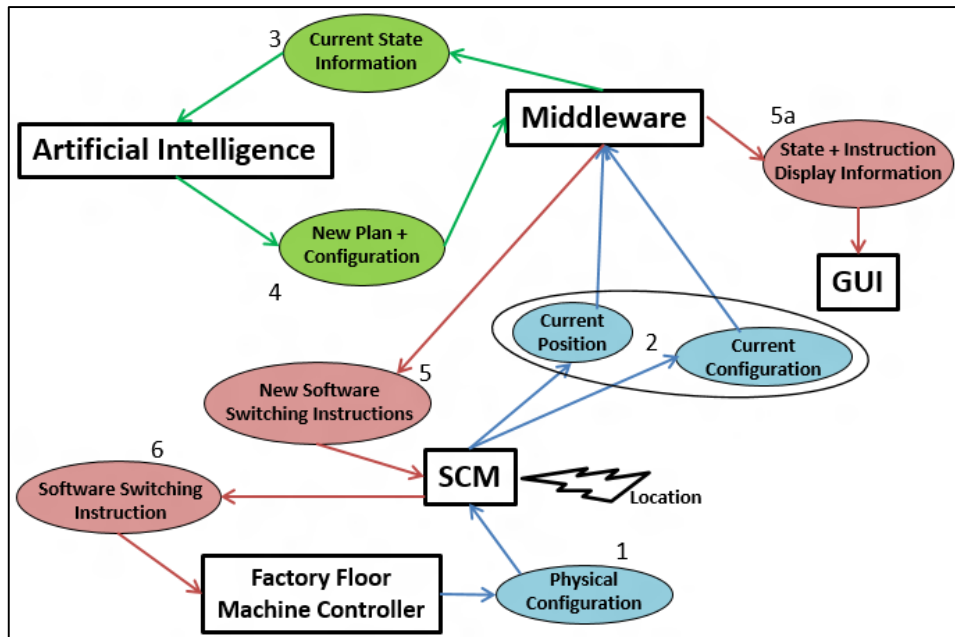


Figure 8: MMS Software Data Flow for a Reconfiguration

The following refers to Figure 8 and starts when the physical reconfiguration has been completed:

1. *Physical Configuration* from machine controller is read by the SCM and converted into a configuration message for the CRC.
2. The *SCM* determines its *Location* and sends it, along with the machine configuration information in a message, to the CRC.
3. The *Middleware* on the CRC sends the information to the *Artificial Intelligence* engine.
4. The *Artificial Intelligence* creates a model including the *New Plan and Configuration*.
5. The *Middleware* sends the *New Software Switching Instructions* to the *SCM*.
5. (a) The CRC *GUI* displays the configuration, with machine positions.
6. Once the physical reconstructions have taken place (according to GUI input) the *Software Switching Instructions* are sent to the *SCM*.

The illustration in Figure 8 shows the operation of the RMMS in its most critical operation and was constructed to guide the design process for each of the sub-systems. This figure should serve as a reference point to the reader. It does not, however, cover the additional functionality conceptually designed into the system – SCADA. It only illustrates the data flow operation of the RMMS in a reconfiguration. The SCADA operation only happens after a reconfiguration has been fully achieved. Once the RMS is in its new state and is in operation, the wireless network of the RMS can provide some basic SCADA operations (along with additional software which is beyond the scope of this research), allowing the SCM to receive progress and operation signals (part count, machine status, operation time, etc.) from the machine controller and convert it into a standard message to be sent to the CRC.

4.3 Structure of the RMMS

Figure 9 shows the structure of the system and how the communications discussed above are carried out. A single SCM/machine controller link is shown here, but this should be visualised with multitudes of SCMs communicating with the wireless module connected to the CRC. This figure serves to further illustrate the workings of the RMMS at a system level. This figure is expanded upon later on in this text, using a similar structure, but showing how the additional

structure fits into the greater system infrastructure. The illustration also shows interaction with higher-level entities like ERP, MES and interacting engineers. This allows new plans to be added and gives functionality to autonomously update inventory requirements and other management information, but its implementation was beyond the scope of the research.

Figure 9 also shows the general duties of the CRC in a reconfiguration operation. After the physical change has taken place, the CRC instructs the SCMs to gather the factory floor info (the physical configuration of the machines and their position). The CRC uses this information to cluster machines into cells, based on their position and uses the configuration of the machines in each cell and combination of cells on the floor to make a decision as to what program should be on each machine. Once this decision is reached by the CRC it sends program switching instructions to each of the SCMs, completing the reconfiguration process.

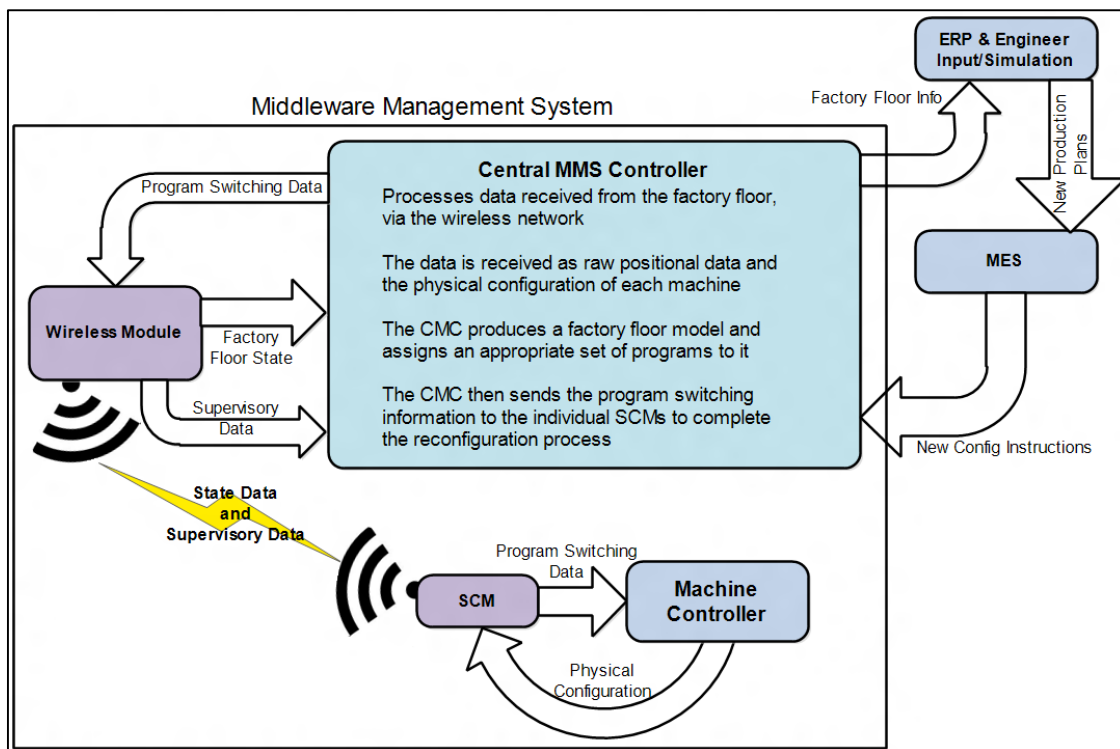


Figure 9: Structure of the MMS

4.4 Chapter Summary

At this stage the reader should be familiar with the goals of the RMMS and how the RMMS was designed to work. The text which follows greatly elaborates on each of the sub-structures outlined in this chapter and an understanding of the roles of each of the attributes is needed in order to fully understand the design decisions made.

This chapter described the RMMS at a system level, showing how it was positioned in the overall control structure of a RMMS and introduced the structure of the system itself. This chapter clarified and gave details of the concept introduced in the previous chapter (section 3.5). The operation description given in section 4.2 served as a guide for the entire design process and should have given the reader a clearer idea of how the system will work at the end of the design process. The structure explained in section 4.3 was built on throughout the design process and this template can be seen in later diagrams on the extended structure of the RMMS (Figure 38 and Figure 39).

5. Chapter 5: The State Communication Module

5.1 Chapter Introduction

The SCM had to play a few crucial roles in the RMMS, it had to provide the hardware support for the RMMS, act as a part of the RTLS, and house wireless communication. This chapter covers the design process of the SCM, including selection of the appropriate core components that make it up, design of the electrical sub-systems and design of the complete SCM circuit. This chapter introduces the structure of the SCM and uses matrix-based selection to choose between the different concepts for each of the facets of the module.

5.2 Structure of the SCM

The State Communication Module, as discussed in section 3.5, had to have a microcontroller and wireless communication hardware capable of using a wireless protocol. This section covers the choice of hardware for the two core components of the SCM and the functions that each of the modules must be capable of. Figure 10, below, shows the function block structure of the SCM as was implemented with the chosen modules.

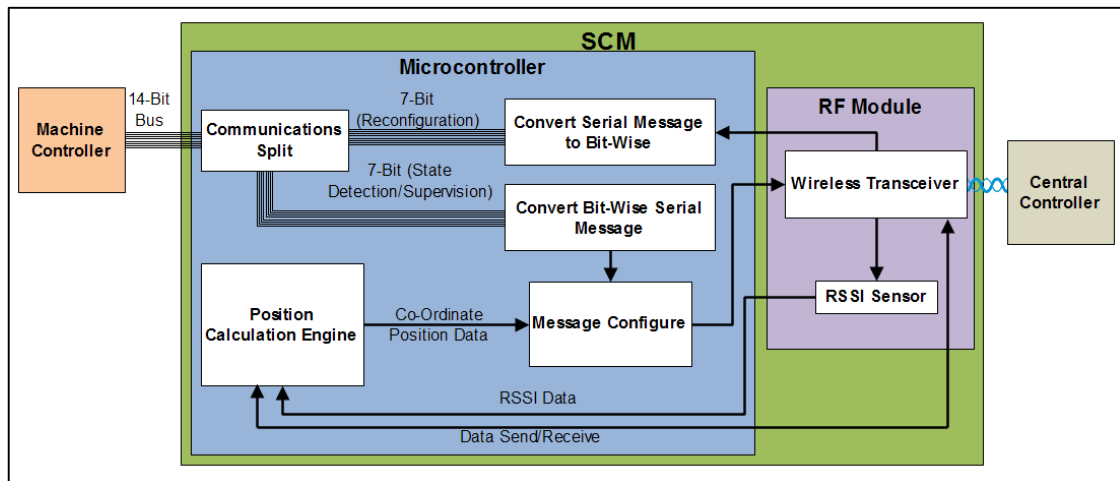


Figure 10: Function Blocks in the SCM

The SCM core components are a microcontroller and a radio-frequency communication module. The next section discusses how the Zigbee wireless protocol was chosen for use as the wireless communication channel. This choice narrowed down the choice for communication modules to ones which are capable of handling the Zigbee protocol.

5.3 Selection of the Core Components

The SCM was designed with core components, a microcontroller for data processing and a wireless module for communications and for the real-time location system. This sub-section discusses the selection of these components.

5.3.1 Radio Frequency Module

The protocol to be used had to be chosen before the radio frequency communications module could be chosen. The various wireless techniques which were considered are discussed in high detail in section 3.6.1. In that section, Table 5 shows a basic comparison of the techniques which were researched. Table 7, below, shows the comparison matrix in the format used in this section.

Each technique has its strong points and weak points, with the Zigbee protocol showing the most consistent high scores and averaging out as the best choice. The DASH7 protocol was not far behind in score, but the lack of readily available modules and difficulty of implementation made it an obviously less suitable option for the SCM module.

Table 7: Weighted Comparison of Wireless Techniques

	Attributes and Weight						Total
	Cost	Range + Mesh Ability	Power Use	Stability / Security / Data Rate	Availability / Support / Ease of Use	Positioning Ability	
Concept	22	23	8	15	17	15	100
Wi-Fi	5	9	2	2	10	3	57.8
WirelessHART	2	9	9	10	7	5	66.7
Bluetooth	10	5	6	3	8	6	65.4
Zigbee	7	10	7	7	9	8	81.8
ISA 100.11a	6	8	8	8	2	6	62.4
DASH7	8	8	10	9	2	10	75.9

The radio frequency module had to be capable of communicating using the Zigbee protocol and be capable of giving a Received Signal Strength Indication (RSSI), as discussed in section 3.6.3. It was beneficial that a Zigbee module is readily commercially available and that many of the commercially available options contain a means of RSSI. Only modules readily available in South Africa were considered. The following options were considered as shortlisted possibilities, although these are only a few of the modules available internationally:

- ◆ Renesas ConnectZED
- ◆ Digi International XBee
- ◆ Telegesis ETRX3 Zigbee
- ◆ Atmel Zigbit

The table below (Table 8) shows how the modules were compared. The method was to grade (with 10 being the best choice and the rest gauged out of 10 in comparison) each applicable attribute of the modules with a weighting as to how important each was to the design decision. In the table, the column labelled ‘Complexity’ covers ease of setup for mesh networking and ‘Support’ refers to the amount of readily available documentation and support for module use. The ‘RSSI Pin’ column is a 10 or 0 column referring to the presence of a physical output RSSI pin on the module. The ‘Processing’ column refers to the level of microcontroller processing power accessible for calculations on the module. The other two columns should have evident meaning, with the cost being estimated based on prices from the *RS Components South Africa* online store [95]. A module with good processing capabilities could, in theory, negate the need for a microcontroller in the SCM, hence the score for Processing. The main issue with this, however, is that the modules have few GPIOs and thus can’t fulfil the role in its entirety. This is the reason that, as can be seen in the table, a module with very little on-board processing capability was chosen as the Zigbee module.

Table 8: Comparison of Zigbee Modules

	Attributes and Weight						Total
	Cost	Complexity	Support	Efficiency	Processing	RSSI Pin	
Concept	35	25	10	10	10	10	100
Renesas ConnectZED	5	7	5	6	7	0	46.1
Digi Xbee	8	10	10	5	2	10	78.0
Telegesis ETRX3	10	3	3	10	10	0	55.6
Atmel Zigbit	9	3	5	7	9	0	51.1

With XBee® having been chosen as the platform for the Zigbee protocol, the choice of which model within XBee® range had to be made. It is evident, as was mentioned in section 3.6.1, that the modules had to be capable of mesh networking, so Series 2 XBee® modules had to be chosen. Once that was established, the decision had to be made between the stock and ‘Pro’ XBee® modules; the ‘Pro’ modules are more expensive modules which have higher strength transmitters, capable of higher signal power. It was known that the ranges of hops in the mesh network lie well within the obstructed indoor range of the standard modules (40m [96]) and the distance to any four positioning beacons would also be in this range. Therefore the extra cost of the ‘Pro’ modules was not justified and the standard module was chosen. Next there was the choice of antenna built into the modules, the available options were *Chip*, *PCB*, *Whip* and *External Connector* antennas, whip antenna modules had the best cost to performance ratio, and were thus chosen. For this reason the *Whip Antenna XBee® Series 2* modules were chosen, the picture below (Figure 11) shows the module, which has an unusual (but well supported) pin pattern using 2mm spacing rather than the standard 2.45mm.

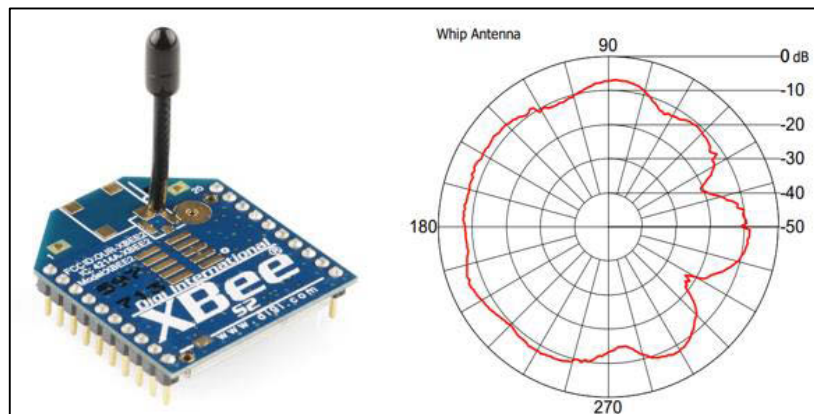


Figure 11: XBee Series 2 Whip Module and Sensitivity Pattern [97]

The RTLS requires consistent readings sensitivity in order to be accurate and the whip antenna gives a reasonably ‘circular’ sensitivity [97], a diagram of the sensitivity can also be seen in Figure 11.

XBee Series 2 modules can operate in both API mode, where the user is able to more directly modify and control the communications behaviour, and AT mode (also known as transparent mode), where the XBees act as a serial link. It was decided at this point that the benefit of implementing API mode would not justify the amount of extra work needed, so the default AT mode was used for the rest of the development.

5.3.2 Microcontroller Selection

The microcontroller is responsible for receiving the serial information from the XBee® module (via a serial port), understanding the message and performing a certain task, defined by the message. The tasks are:

- ◆ Converting a received message into a bit-wise program swapping message for the machine controller (as discussed in detail in section 3.6.1, 3.6.5 and 5.5).
- ◆ Receiving a bit-wise configuration description from the machine controller and configuring it as a serial message (as discussed in detail in section 5.4).

- ♦ Calculating its own position based on a RSSI signal from the XBee® module (as discussed in sections 3.6.2 and 6.2).
- ♦ Constructing a message containing the machine configuration and its position on the factory floor and sending it, via the serial port and XBee module, to the central controller (discussed in sections 3.6.2, 3.6.5, 5.4, 8.3).

Receiving basic SCADA information via I/O ports and feeding this information to the CRC along the serial port and XBee® (discussed in section 9.4)

Most of the tasks are simple, with very little computational power required. The exception is the position calculation, which involves complex calculations and large (in terms of microcontrollers) amounts of Random Access Memory (RAM). The positioning system sub-routine alone required over 1 KB of RAM to operate, which limited the choice of microcontroller somewhat. It was also beneficial for the microcontroller to be as fast as possible, with as much RAM as is available at a reasonable cost. Over-specifying the microcontroller leaves room for more accurate calculations and further expansion, so that the RMMS would not be limited by hardware restrictions from the MCU.

It was decided very early on that a microcontroller integrated board would be used for the prototyping of the BPM and SCM. The main justification was that these boards are easy to set up and use with easily accessible I/Os and other features such as on board voltage regulation and connectors. Using a pre-assembled board greatly speeded up the development process and is justified in the simplicity of use.

The choice, therefore, lay in which brand or model of development board to use. There are many competitive choices falling into three main categories:

- ♦ Arduino and Arduino Clones,
- ♦ Raspberry Pi,
- ♦ BeagleBone.

Arduinos and *Licensed Arduino Clones* are popular for prototyping, hobbyists and development because of their low cost, simplicity and multitude of online support. Arduino has its own Integrated Development Environment (IDE) which uses basic C code and can also be programmed using any IDE which can compile C code. Arduinos generally house 8-bit processors from Atmel, although some models and clones have 32-bit, 84 MHz processors. These boards are available with high numbers of General Purpose Input/Output Pins (GPIOs). Arduinos operate at a very low power, which is advantageous in the industrial environment. The biggest advantages of the Arduino boards are support and cost.

Raspberry Pi boards are closer to full personal computers in their power, with a 700 MHz processor and 512MB of RAM. They are programmed in Linux but are highly customisable, with 17 available GPIOs. This device has many additional features which increase the cost, including HDMI output and networking ability which is not required for the SCM. The cost of a Raspberry Pi is generally about double that of an Arduino board, but with far more performance available. This price increase is not justified. The power usage is much higher than the Arduinos or Android device, to go with the better performance.

BeagleBone boards are very similar in concept to the Raspberry Pi, but with lower power consumption and far more GPIOs (65). They, like the Pi, were considered to be over specified for the application and their high cost (four times higher than a reference Arduino) could not be justified.

The best choice for the microcontroller board in the SCM was an Arduino development board, but many different Arduinos and Arduino Clones available in the marketplace. The comparison below (Table 9) shows how the model of Arduino platform board was chosen. The table follows the same format as before and most of the columns are self-explanatory. The ‘Extra’ column refers to a score for having extra features when compared to other boards, such as a built in XBee socket for the module mounting.

Table 9: Comparison of Arduinos

	Attributes and Weight						Total
	Cost	Performance	I/Os	Efficiency	Size	Extra	
Concept	30	25	12	13	10	10	100
Arduino Micro	9	5	3	10	9	5	65.2
Arduino Mega	3	7	10	4	2	4	45.7
Arduino Due	2	10	10	4	2	7	50.3
TeensyDuino 3.1	8	9	8	6	10	7	74.0
Seeeduino Stalker	6	6	5	10	5	10	57.1
Redboard	10	5	4	3	5	1	56.2
Fio v3	6	2	4	5	4	10	38.4

The table shows that the strongest choice for the Arduino board is the Arduino clone, TeensyDuino 3.1, also known as the Teensy 3.1. This model has an 84 MHz, 32-Bit ARM Cortex microchip, essentially the same as on the Arduino Due. It has 64KB of RAM and is priced very competitively. The board has an outstanding price to performance ratio. The Teensy is technically an Arduino Clone, but will also be referred to as ‘the Arduino’ or ‘the microcontroller’ in the rest of this text. A picture of the Teensy 3.1 is shown in Figure 12.

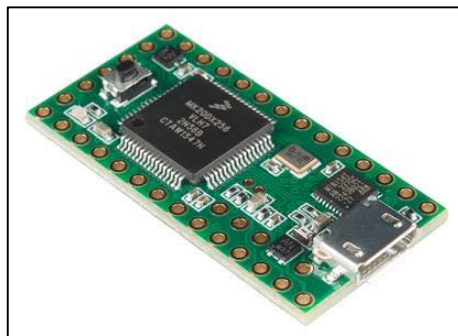


Figure 12: The PJRC Teensy 3.1

5.3.3 Interfacing of the Modules

Interfacing of the chosen Arduino with the XBee module is very simple, the XBee only needed to be linked to power pins (Ground and 3.3V) and to the pins corresponding to the serial port of the Arduino. In the case of the Teensy 3.1 there are 3 serial ports so the choices are:

- ◆ pin0 and pin1,
- ◆ pin9 and pin10,
- ◆ pin7 and pin8.

An extra link was also needed for the purpose of the RTLS. The XBee RSSI pin (pin6) had to be linked to an analogue pin on the Arduino (pin14) was chosen. The following circuit diagram (Figure 13) shows how the modules were connected.

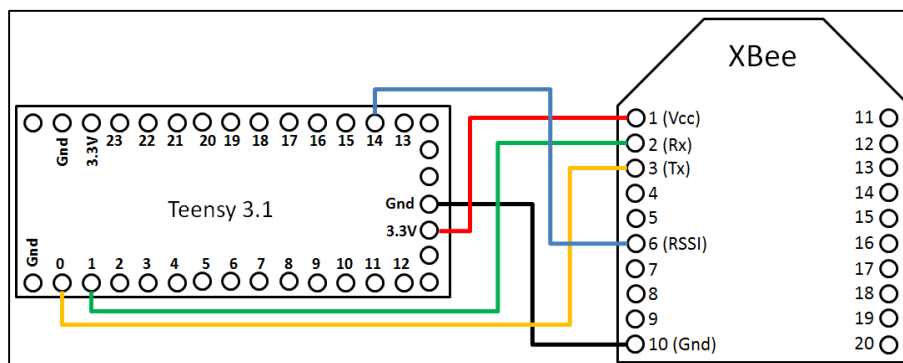


Figure 13: Connecting the Teensy 3.1 to a XBee

Figure 13 shows that many of the pins of the XBee module were not used in the interfacing. Some of the pins on the module can be used as logic output pins and for other functions which are not made use of in the application of the RMMS. The on-board processing on the XBee was too weak to be made use of, so the pins were not used. The open pins on the Teensy are reserved for use for communication with the machine controller, as is discussed in the next two sections.

5.4 Communication: Machine Controller to SCM

Section 3.6.5 discusses the strategy used for machine controller communications – communication between the large variety of different, heterogeneous machine controllers and SCM will be handled with set bit-wise high/low signals. This method maximises compatibility and thus allows for practically every brand and control architecture which is used in machine control to communicate simply and efficiently with the SCM. There is an issue which may not be obvious when thinking conceptually about this type of communication – the difference in the I/O voltages of the machine controller and SCM, the solutions to this are discussed later.

The machine controller has to be programmed at the initial setup phase of the RMS to store all the relevant programs (to be switched between on command) and to include the necessary understanding of the bit-wise messages. The table used as the basis for bit-wise message configuration is shown in Table 10.

Table 10 shows that some messages are reserved for certain functions, with the others left for program switching states. The set combinations of bits are shown in the table for both messages from the SCM to machine controller, and for messages in the other direction. Pseudo-code is shown in the listed figures, for the SCM requesting a configuration message from the machine controller (Figure 14), and for the machine controller responding with a physical configuration message upon request. In the pseudo-codes the pin0-pin6 on the SCM are output pins and are connected to the same named pins on the machine controller, which are inputs. The reverse is true for pin7-pin13. The two pseudo-codes should be read together, with the reader referring to each, as the routines are interactive. The other primary function of communication between the SCM and a machine controller (switching between programs) is discussed in section 5.5.

Table 10: Standard Message Table: SCM to Machine

Messages: SCM -> Machine	
Bit Pattern	Instruction
0 0 0 0 0 0 1	Emergency Stop
0 0 0 0 0 1 0	Soft Stop
0 0 0 0 0 1 1	Reconfiguration Request
0 0 0 0 1 0 0	Program Set Request
0 0 0 0 1 0 1	Configuration Received
0 0 0 0 1 1 1	Choose Program
1 1 1 1 1 1 1	

<ol style="list-style-type: none"> 1. if (serial port reads "State") do 2. wait 2s 3. set pin0-pin6 to: [0 0 0 0 0 1 1] 4. wait 5s, keeping pins set 5. if (pin7-pin13 reads [0 0 1 0 0 0 1]) do 6. wait 5s 7. read pin7-pin13 and store as array of type in 8. set pin0-pin6 to: [0 0 0 0 1 0 1] 9. end if 10. else wait 5s and jump to 5 11. end if 	<ol style="list-style-type: none"> 1. if (pin0-pin6 reads [0 0 0 0 0 1 1]) do 2. stop any operations 3. run method to gather current configuration 4. convert configuration into bit-wise 5. store configuration as an array 6. if (pin0-pin6 does not read [0 0 0 0 1 0 1]) do 7. set pin7 = array[0], pin8 = array[1], etc. 8. wait 10s, keeping bits set 9. end if 10. else wait 5s and jump to 6 11. end if
--	--

Figure 14: Pseudo-Code for SCM (left) and Machine Controller(right)

The machine controller must be able to communicate SCADA data to the SCM during the actual operation of the RMS as well as using these message tables. This is covered in section 9.4.

The message table shown above (Table 10) allows for 108 unique physical configurations to be communicated to the SCM and allows for 120 unique programs to be switched to on the machine controller, no practical setting will need more. It could even be argued that an excessive amount of I/Os is used by the SCM. It could easily be changed, especially by reducing the number of communication channels from the machine controller to the SCM (because the number of possible configurations is lower than, or at worst equal to, the number of possible programs). This is left for later research.

For both the SCADA applications discussed later and for the message signalling as reviewed above, the difference in high/low signal voltages on the I/O ports of the machine controller and SCM – in particular the microcontroller in the SCM. The SCM uses a microcontroller board with I/Os operating at a logic level of 3.3V, which is different to the prevailing 24V, 18V, 12V and 5V on PLCs and other machine controllers and breakout boards (see section 5.3.2). For this reason the voltage needs to be scaled down so as to not overload a typical 3.3V microcontroller's pins, which are only rated to 3.8V [98]. There are several ways to step down the voltage of the output from the machine controller to the input pins of the SCM's MCU, these methods reviewed are discussed in the next sections (5.4.1, 5.4.2 & 5.4.3).

5.4.1 Electrical Conceptual Designs and Considerations (Step Down)

Five basic methods of stepping down a logic level are described and deliberated below; an optoisolator, a relay, a voltage divider circuit, a retail logic-level converter and a transistor. The cost of each of these methods is based on prices from *SparkFun Electronics* [99]. A circuit of each of the concepts is given with each description.

An *opto-isolator* works by using a voltage (in this case, the higher voltage from the machine controller) to trigger one side of the optic system to provide an optic signal to a receiver which completes the circuit on the other side. The completed circuit is shown in Figure 15 and is connected to the V_{cc} of the MCU and to the input pin, so when the circuit is completed it sends a high to the MCU. In this case the logic levels of the machine controller will be reversed, but this is simple to solve in the software. In this case, the circuit would need to be repeated for each signal channel. Single integrated circuits (ICs) can be purchased which contain all of the circuitry shown below and only need to be connected to the input pin, output pin, V_{cc} and ground.

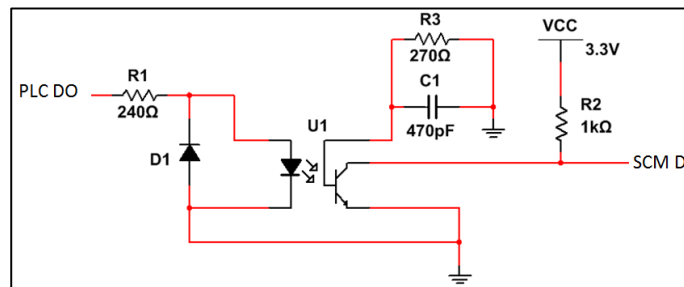


Figure 15: Opto-Coupler to Convert Logic Levels

A *relay* works by using the high from the machine controller's signal voltage to trigger a relay to complete the circuit on the MCU side of the relay. The completed circuit is shown in Figure 16 and connects the MCU's V_{cc} to the input pin, thus signalling a high. Many types of relays are possibilities, although relays are typically used in the reverse – to switch a high voltage line using a small signal voltage. In this case, with low voltages and a small required size, a solid-state relay would be most suitable. Multiple channel relays are commercially available, as are single channel units. Each signal channel will need to have a relay attached and these used would need to match the switching voltage of the relay to the signal voltage of the machine controller.

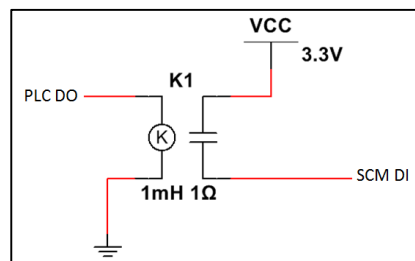


Figure 16: A Relay to Convert Logic Levels

A *voltage divider* circuit is a simple way of dropping a voltage signal; it works by using a series circuit of two selected resistors connected to ground with the input pin of the MCU connected between the two resistors. The ratio of the second resistor value and total resistance must match the ratio of voltages; so that a drop in voltage from 24V-3.3V implied the second resistor must be approximately 13% of the value of the sum of the first and second resistors. This method could be slightly energy inefficient, because energy is lost in the form of heat through the resistors. The strength is that this method is very inexpensive and very small, needing only two resistors (see Figure 17). In this case, the resistor-based voltage divider circuit will need to be duplicated for each of the I/O channels. Because of its size, though, the circuit can be printed on a very compact PCB and can thus be made very small.

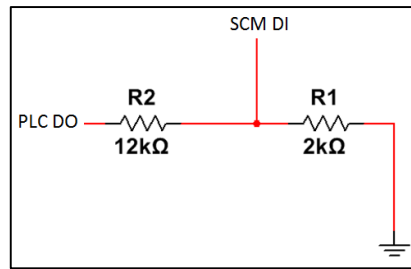


Figure 17: Voltage Divider to Convert Logic level

A *transistor* could be used where the base of the Negative-Positive-Negative (NPN) transistor is connected to the output pin of the machine controller, via a current limiting resistor, and used to trigger the transistor into forward bias mode (as shown in Figure 18). The forward bias mode of the transistor then allows current to flow from V_{cc} to ground of the MCU. A resistor limits the current and the input pin is connected after the resistor, this gives a high to the input of the MCU. In this case a low will trigger the transistor and send a high to the MCU, so the logic would have to be inverted in the machine controller software.

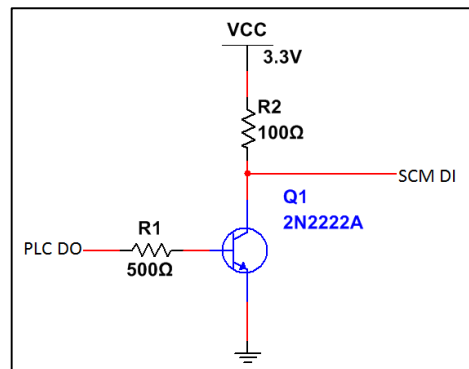


Figure 18: A NPN Transistor to Convert Logic Level

Retail Logic Level Converters are available from electronics shops and use one of the methods described above printed to a premade PCB and sold as a set package with different numbers of channels and set voltage conversions. An example of a 4-bit, 5V to 3.3V is shown in Figure 19. This circuit assembly part is readily available from *SparkFun Electronics* [99], but all the required voltage conversions are not as readily available.

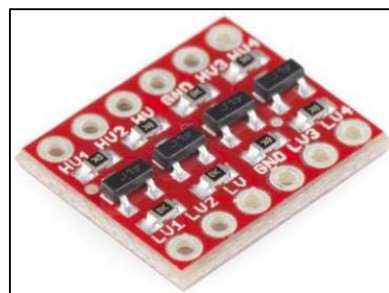


Figure 19: A Retail Logic Level Converter

5.4.2 Comparison of Electrical Concepts (Step Down)

The following comparison (Table 11) follows the same format as the tables discussed before. Each of the column meanings should be self-explanatory.

Table 11: Comparison Table of Logic Level Converters

Concept	Attributes and Weight					Total
	Cost	Complexity	Size	Efficiency	Compatability	
Concept	35	25	20	10	10	100
Opto-Isolator	3	5	6	9	10	54
Voltage Devider Circuit	10	8	10	6	2	83
Physical Relay	4	7	2	8	10	53,5
Transistor	7	6	8	10	10	75,5
Retail Converter	2	10	6	5	2	51

Table 11 shows that the voltage divider circuit was judged to be the best choice for step-down conversion of the logic channels from a machine controller to the SCM. The following section shows the circuit designed for the conversion of logic levels of various signal voltages supplied by machine controllers.

5.4.3 Electrical Design: Logic Level Step-Down

The voltage divider circuit was chosen for development in the RMMS (discussed in section 5.4.2); the development of the final design is discussed and described below.

Different resistor combinations are needed for each of the different possibilities of signal or operational voltage from the machine controller. In the case of the SCMs, the signal or operational voltage will always be 3.3V. This is the only real disadvantage of the concept, because each different voltage output needs a different combination of resistors and thus each SCM will only be compatible with a set controller voltage and thus each particular SCM can only be used on a limited subset of machines, a small disadvantage.

Ohm's law states that for a set voltage drop (equal to the signal voltage of the machine controller output) the circuit resistance and current are inversely proportional. The power law of circuits states that the power consumption is the product of voltage (which is constant in this case) and current. The goal was to use as little power as possible, which implies that the circuit was designed to draw little current.

The ratio of first resistor and sum of resistance had to match the ratios of voltages of the machine controller and SCM. Thus, the ratios for controllers of various voltages were calculated and are shown in Table 12.

Table 12: Machine Controller Step-Down Ratios

Machine Controller Operational Voltage	Ratio Resistor 1 : Sum R
5V	1 : 0.6600
12V	1 : 0.2750
18V	1 : 0.1833
24V	1 : 0.1375

These ratios showed that it was clear that there are hundreds of different combinations to suit the ratio requirement, but that further considerations need to be made as to the resistance of the circuit.

Because the voltage supplied is set, the higher the resistance, the better, because with high resistance comes low current and thus a low dissipated power. This is desired for power saving and for lower heat expulsion from the resistors. An example calculation is shown in [Eq. 11] for the conversion from a 24V controller to the 3.3V SCM – which will give the ‘worst-case’ values, where the sum of resistance is lowest. For these calculations it was desired to have a designed power consumption below 0.05W.

$$P=VI \Rightarrow I=\frac{P}{V} \quad \& \quad R=\frac{V}{I} \therefore R=\frac{V^2}{P} = \frac{24^2}{0.05} = 11520\Omega \quad [\text{Eq. 11}]$$

The calculation above shows that that a combined resistance of over 11520Ω is required. It is known that a higher resistance will give yet less power used so a set value for the first resistor, of R₁=15kΩ, with the second resistor value varying for each different controller voltage. For the reasons explained in the previous paragraph and by the calculations shown above the following decisions, shown in Table 13, were made with respect to resistor values.

Table 13: Resistor Values Chosen for Different Controller Voltages

Machine Controller Operational Voltage (V)	Resistor 1 (Ω)	Resistor 2 (Ω)	Output (V)	Power Use (W)
5	15000	10000	3.33	0.0010
12	15000	4000	3.20	0.0076
18	15000	2700	3.24	0.0183
24	15000	2000	3.20	0.0339

Table 13 shows that the required output can be achieved well within the sensitivity of the SCM’s MCU with common resistor values and low power consumption values. The following circuit (Figure 20) shows the layout of the circuit for logic level conversion for a 24V PLC.

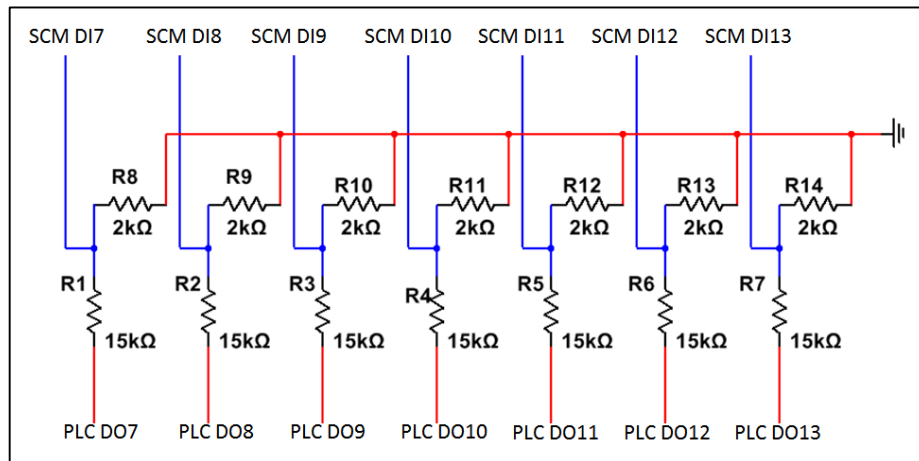


Figure 20: Circuit for the Logic Level Step-Down of All Channels

The design shown above was tested to work efficiently and proved to be a good method for stepping the voltage down in this application. An experiment was performed, documented in section 10.2.1, which proved the capability of the design for various voltages.

5.5 Communication: SCM to Machine Controller

Communication in the reverse direction to the one discussed in the section before was also critical for the desired RMMS functionality. Communication in this direction allows the SCM to pass program switching (software reconfiguration) instructions to the machine controller to complete

the reconfiguration process of the RMS. In a similar strategy to the communication from heterogeneous machine controllers to their attached SCM, the communication in the reverse direction was designed to be a bit-wise multi-channel high/low message using 7 channels.

The ability of the system developed in this research to update control instructions in the form of switching between programs on a machine controller is an important attribute of the management of reconfiguration. The automation of this process was a primary goal of this research with the aim to develop a system which was capable of removing the need for management by a human operator and replacing this labour intensive and time consuming activity with an efficient, computer controlled process.

The role of this communication is for the SCM to pass on program switching instructions from the CRC to the machine controller. The information is sent from the CRC to the SCM as a serial message along the wireless network and the SCM reads this message and interprets it as an instruction to start the program switching sequence. The SCM requests for the machine controller to accept a program switching instruction and then sets its 7-bit communication channel to the combination corresponding to the program required for the new RMS state. Each of the possible programs to be run on a machine must be contained on the memory of the controller, as a prerequisite for the use of the SCM RMMS. The 7-bit message sent by the SCM to the machine controller is an instruction to switch to a new program corresponding to the message as shown in Table 14. The following figure (Figure 21) is a set of pseudo-codes for what was programmed to the machine controller and SCM.

Table 14: Standard Message Table: Machine to SCM

Messages: Machine -> SCM	
Bit Pattern	Bit Read
0 0 0 0 0 0 0 0 0 0 1 1 1 1	Error 1 - 15
0 0 1 0 0 0 0	Resend Configuration
0 0 1 0 0 0 1	Request Received
0 0 1 0 0 1 0 1 1 1 1 1 1 0	Configuration Send
1 1 1 1 1 1 1	Switch Successful

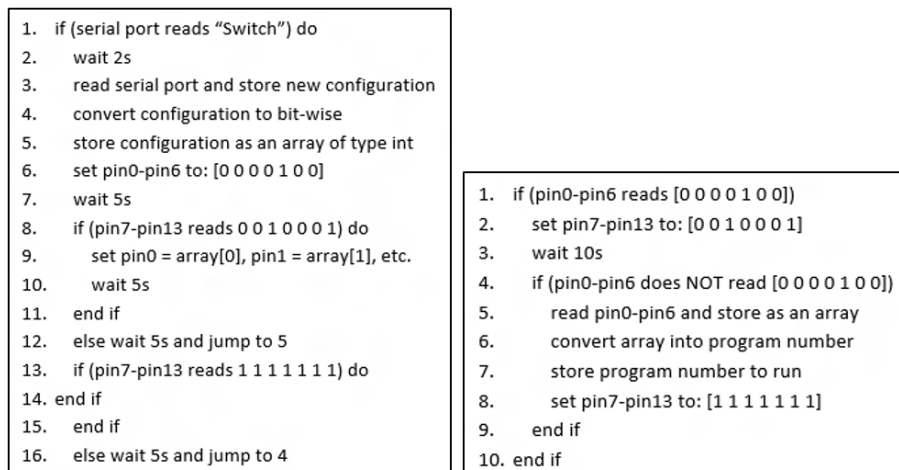


Figure 21: Pseudo-Code for SCM and Machine Controller

After this communication has taken place, the machine controller uses the stored program number to choose the program to run. The exact implementation of the method is beyond the scope of the research, but this can be achieved in many ways, depending on the machine controller and preferred method of the programmer:

- ◆ switch/jump statements at the start of each loop;
- ◆ an interrupt to run a subroutine;
- ◆ a main method which jumps to a sub-routine;

A simplified flowchart shown in Figure 22 provides a possible example structure of the routine to be implemented in a machine controller in order to respond to the SCM's program switching instructions. The flowchart shows that if the inputs on the digital IOs of the machine controller are changed then the program section which is implemented differs. The example shown would be implemented on a form of gantry machining platform such as the prototype designed at UKZN.

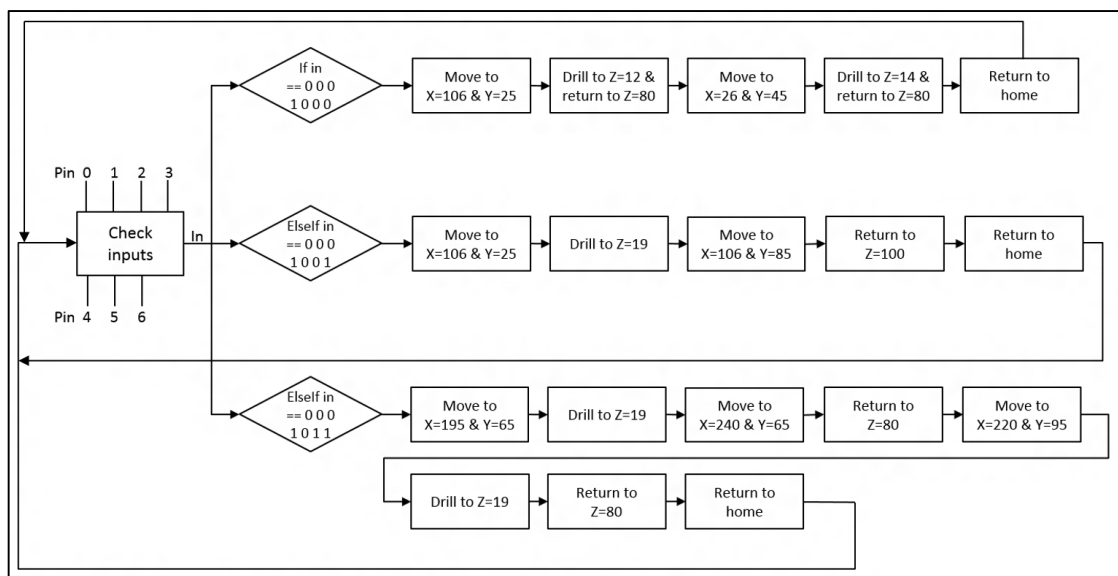


Figure 22: An Example of Responding to SCM Software Switching

The communication from the SCM to the machine controller encountered a similar (but reversed) problem to the communication channels in the opposite direction. In this case the voltage needs to be stepped up from the 3.3V of the SCM's microcontroller to the voltage of the machine controller (5V-24V). It is clear that the same method chosen in section 5.4.3 cannot be used for the reversed voltages. For this reason, new concepts were investigated and are discussed, compared and finalised in the next sections (5.5.1 to 5.5.3).

5.5.1 Electrical Conceptual Designs and Considerations

The application is the reverse of what was discussed in section 5.4.1 but many of the concepts can be made to work in the opposite direction. The options investigated for the step up from SCM to controller voltage were: opto-isolators, relays, a retail logic level converter and transistors.

A description of an *opto-isolator* is given in section 5.4.1, with the only changes being that the circuit will be connected as follows in Figure 23. In this case, the 3.3V from the SCM's output triggers the LED, which allows the phototransistor to complete the other side of the circuit, sending a high to the input of the machine controller. This method is compatible with any voltage of controller, because the controller's own voltage is used as the high signal.

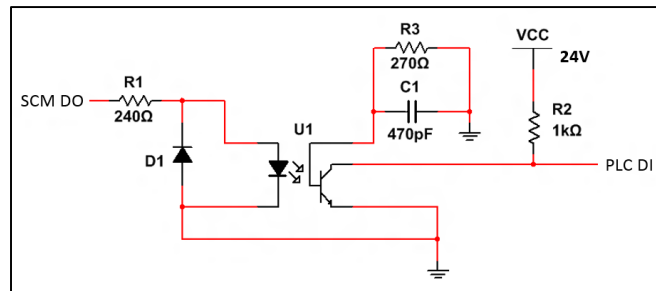


Figure 23: Opto-Isolator to Convert Logic Levels

In a similar manner to the previous concept, the *relay* concept is also very similar in nature to what was described in section 5.4.1. The workings of the relay match what was described in that section. The circuit changes slightly to what was shown in the previous section. In this case, the relay is used in its more conventional configuration, with a low voltage triggering the close of a higher voltage switch (Figure 24). Again, this method is compatible with any controller voltage.

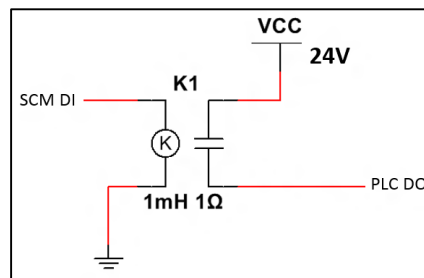


Figure 24: A Relay to Convert Logic Levels

The *retail logic level converters* are usually bi-directional and thus exactly the same applies here as in section 5.4.1. The retail logic level converter is the only concept discussed here which is not compatible with all controller voltages. No further discussion or figures were needed here.

Transistors, particularly an NPN transistor, can be used - as in section 5.4.1 and as discussed above - as a switch triggered by the high signal at the low voltage of the SCM. The circuit is just as before, but with the input and output reversed and supplied voltage coming from the controller itself (Figure 25). This method is compatible with any controller voltage and is known as a common emitter amplifier. In this case the logic levels would be inverted, so that a high from the MCU would trigger a low read by the machine controller and a low from the MCU would trigger a high at the machine controller. This was simple to handle in software on the SCM.

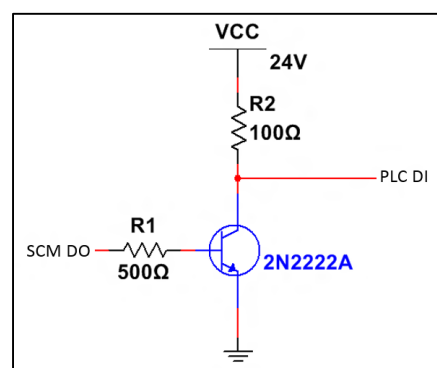


Figure 25: A Transistor to Convert Logic Levels

5.5.2 Comparison of Electrical Concepts (Step Up)

Making use of the same technique as section 5.4.1, a table compares the conversion methods (Table 13), with the same attributes compared. This section uses slightly different weightings due to the lower voltage switching and occasional battery power needs of the SCM requiring more weight given to the efficiency.

Table 15: Comparison of Logic Level Step-Up Methods

	Attributes and Weight					Total
	Cost	Complexity	Size	Efficiency	Compatability	
Concept	31	31	15	25	8	110
Opto-Isolator	4	4	6	8	10	61,8
Physical Relay	5	6	2	7	10	62,6
Transistor	10	5	10	10	10	94,5
Retail Converter	3	10	6	4	2	60,9

The table above shows that the transistor circuit was the best choice for converting the logical high of 3.3V on the SCM to a logical high which matches that of the linked controller.

5.5.3 Electrical Design: Logic Level Step-Up

The transistor circuit was chosen for use as the logic step-up converter. For this design, a conversion circuit is required for each signal channel, so seven transistors are needed to convert all the signals.

There are many different versions of NPN transistors, each with specific properties; a book by M Margolis [100] was used as a guide for choosing the transistor model. Each possible choice was considered along with National Instruments' *Multisim* Circuit Design software to test the validity of the circuit. During the design process *RS Components South Africa* [95] was consulted for pricing and availability of the transistor model being considered. Using these tools, the author made an informed decision to use a *P2N2222A* model NPN transistor, manufactured by *ON Semiconductor*. The specifications sheet is shown in Appendix A (section 15.1), this model is capable of switching up to 40V, so is more than capable of handling industrial signal voltages. The picture below (Figure 26) shows the transistor used in the SCM construction.

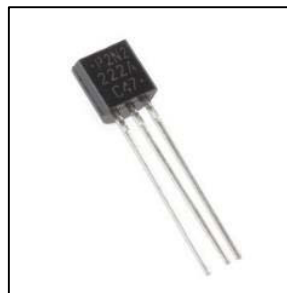


Figure 26: A P2N2222A Transistor

The following circuit shows the final seven channel step-up circuit design, used to allow the SCM to send messages to a higher signal voltage based PLC or other machine controller.

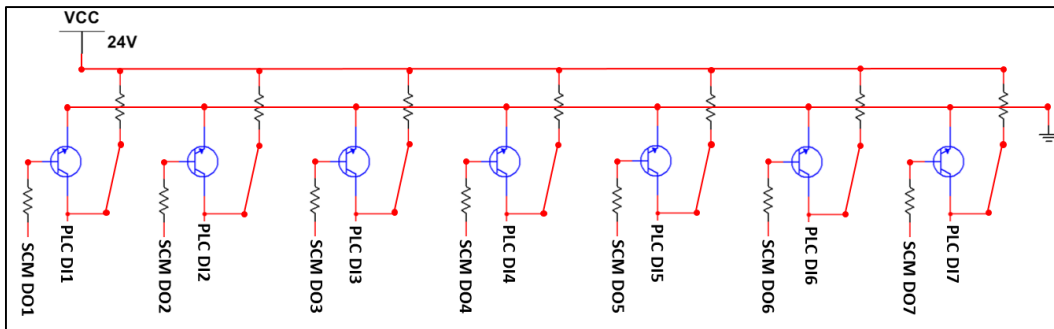


Figure 27: Circuit Diagram for Step-Down Circuit

The above design was tested in an experiment, documented in section 10.2.2, which proved its capability to provide a stepped up logic level for communication with a machine controller which uses a higher logic level voltage.

5.6 Peripheral Electrical Design

The ability of the SCM concept to be compatible with multiple machine controllers is central to the SCM design. For this reason, the SCM was designed to be able to be powered by multiple voltages. It was also designed to be used with its own on-board power when the machine controller is not powered up. These two requirements are outlined in section 3.6.7. A method was needed for the regulation of high machine operating voltages to the required microcontroller supply voltage (3.7V-5.5V see Appendix B (section 15.2)). A method was also needed which would switch to battery power upon loss of power from the machine.

5.6.1 SCM Power Supply

Different resistor combinations are needed for different machine controller logic voltages and at the same time, it may be required to step down from a different voltage for the supply. In the vast majority, the supply voltage of a machine controller is higher than or equal to the signal voltage of that controller. The most common industrial machine controller operating and supply voltages are 18V & 54V. The drop in supply voltage from those high voltages to the low voltage required by the SCM is quite large (although current is small). For this reason, a cascading voltage drop circuit was designed, to drop the voltage first to 12V and then down to 5V, which was supplied to the SCMs. This cascade reduces the regulator load, reducing their chance of overheating or failing. The current is far below what the regulators used were rated for, but the extra safety was desired. The cascading circuit made use of TO-220 shape voltage regulators, excerpts of their datasheets are shown in Appendix C (section 15.3) (12 V) and Appendix D (section 15.4) (5V). The circuit diagram below (Figure 28) shows the cascading voltage regulator circuit.

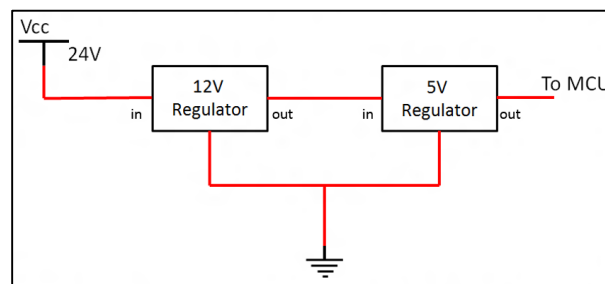


Figure 28: Cascade Regulator Circuit Diagram

The circuit shown above is simple and allows the SCM to operate with any common supply voltage greater than 14V. In the cases when the supply voltage was below that value, the first

regulator was replaced by a direct wire link, leaving only the second regulator (5V). The design shown above was tested and the experiment is documented in section 10.2.3.

There was an outset requirement of battery back-up, which lead to the need for a switch circuit, which changes to battery power when power from the machine is shut off. A reaction circuit like this can be considered to be complex, but basic use of Kirchoff's Laws leads to a very simple solution. The following circuit diagram (Figure 29) and explanation illustrates the solution used.

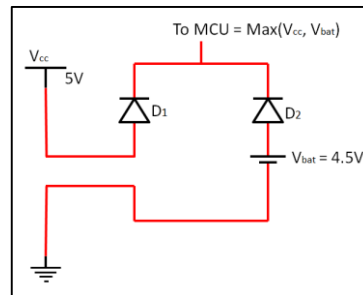


Figure 29: Battery Switching Circuit

In a simple parallel circuit, the voltage of each of the parallel arms must be equal. Therefore, by putting two sources in parallel with each other would force the larger voltage source to reverse the polarity of the other source – batteries and power supplies do not like this, to put it mildly. This is why diodes are used to ensure the single-direction flow of current. Whenever the power is on from the voltage supplied by the circuit in Figure 28, the polarity will change over D₂ and the MCU will be supplied with $V_{cc} - 0.7V$. When the voltage at V_{cc} drops to nothing (when the machine controller is unplugged), the battery voltage will cause the polarity of D₁ to switch and the MCU will be supplied with $V_{bat} - 0.7V$. This is a very simple and elegant circuit, but does require the battery voltage (minus the voltage drop over the diode (0.7V)) to be lower than the board supply voltage (5V), but higher than the minimum supply needed by the MCU (3.7V). So it must be within the range of approximately 4.4V-4.9V. This does limit the battery selection, but such batteries are still readily available.

5.6.2 Manufacture and PCB Design

The circuit had to be manufactured for testing. The manufacture of the prototype of the SCMs, of which eight were made, required the assembly of many components, as discussed in the designs in the last three sections. The board design would be complex to make by hand, so the decision was made to use a printed circuit board (PCB). A simple board was needed, to save cost and complexity, so a single layer board was chosen. The design which was developed connects all the core components of the SCM so that it can regulate supply voltage, switch between battery and auxiliary power, boost the out-signal voltage and limit the in-signal voltage. The design of the PCB is shown in Figure 30, with the key which follows it:

- ◆ Red Line – Resistor Link
- ◆ Yellow Arch – Wire Link
- ◆ Purple Line – Diode Link
- ◆ Pink Block – Off-the-shelf Module
- ◆ Light Blue Line – PCB Copper Link
- ◆ Grey Rectangle with Hole –Through Hole and Solder Pad
- ◆ Grey Outline– NPN Transistor
- ◆ Orange Outline – Voltage Regulator

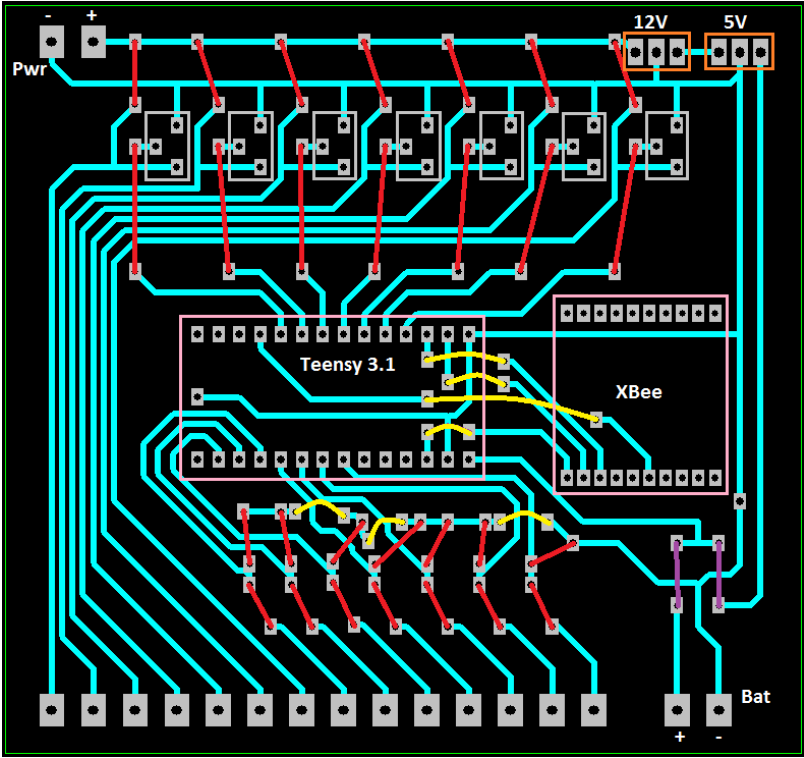


Figure 30: PCB Design

The fully assembled PCB for the SCM is shown in the following picture (Figure 31). The black terminals allow for the connections between the SCM and machine controller.

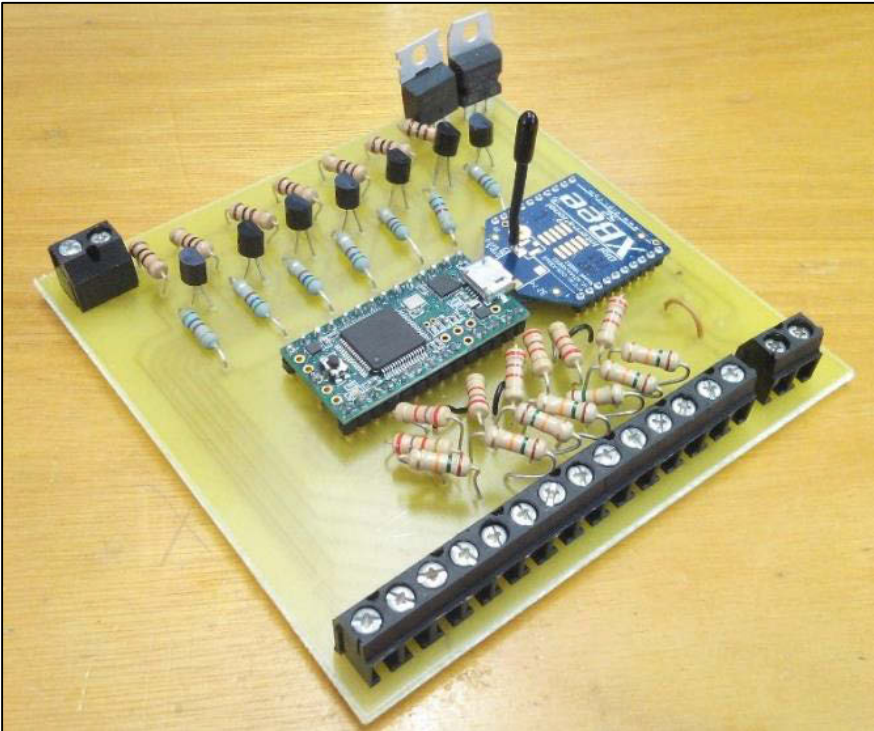


Figure 31: Assembled SCM PCB

5.6.3 SCM Battery Considerations

It was one of the design goals that the SCM was able to continue communications while its primary power source – the machine controller – was offline (see section 5.6). The SCM therefore

required a battery. The selection of a battery was governed by cost, suitability, size and safety. In the same section it was argued that the battery voltage had to be within the range of 4.4V-4.9V. It was possible to use a higher voltage battery and to construct a regulation circuit which feeds into the switch circuit described in the preceding section. This seemed wasteful, because extra size and cost would be applied without much benefit.

The primary decision to be made was whether to make use of a rechargeable or disposable battery for the reserve power source. It must be made clear that by *rechargeable* the batteries that are recharged by the modules themselves, while there is access to machine power, were implied. The term rechargeable, in this text, does not apply to a chemically rechargeable battery which needs to be removed from the module to be recharged – or which needs to have the module plugged into an external charging circuit.

The advantages of using on-board rechargeable batteries are clear – these don't need to be replaced in the life cycle envisioned for the system, rechargeable batteries provide less harmful waste and in high usage scenarios the rechargeable nature gives a lower cost [101]. The disadvantages of rechargeable systems are less clear, but prove important in this application. Rechargeable batteries are generally less stable and safe than their disposable counterparts (particularly during charging) [101] and extra circuitry and logic are needed for their recharging within the module (this is especially true for multi-cell Lithium-Polymer (Li-Po) batteries). It was decided that the use of non-rechargeable batteries in the modules would be more suitable – with reduced cost due to the lack of a required charging circuit and safe nature of no on-board charging. Discussed in the following paragraph, the modules will not be consuming much power during their non-powered operation and will thus not regularly run their batteries down.

During idle periods, when the high-intensity communications and calculations conducted during the software reconfiguration operation are not active, the microcontroller merely loops through practically blank code. It periodically checks the serial port for an input. The XBee also just waits, scanning for an input. The current draw, therefore is minimal during this time (± 20 mA) and with the typical battery capacity of an AA battery of 2300 mAh, the battery would last for ± 115 hours. It was decided that this battery life was sufficient for the testing purposes of the system (see the customer specifications in Table 1).

An on-board charging circuit was not included in the design of the SCMs because the additional logic, circuitry and design included in its implementation would yield limited impact. In addition, any advantages provided by the use of rechargeable batteries would go unnoticed in an experimental environment. The modules could, in the future have this included when industrial applications become realistic, but at this stage it was decided to leave it out and use disposable battery packs.

5.7 Chapter Summary

The design process of the SCM was described in this chapter. Each of the facets of the SCM's electrical design were introduced and developed into a design for use in the version of the SCM for the experimental RMMS. This section covered the selection of core modules, the development of methods for the voltage step-up and step-down of logic levels for communications, peripheral electronics design, and design of the PCB onto which the electronics were mounted.

6. Chapter 6: Real-Time Location System

6.1 Chapter Introduction

This chapter of the dissertation contains a highest volume of novel work and innovation of the RMMS, other than the system concept itself. At this point, as discussed in section 3.6.3, the concept of using RF signal strength for location has not been confidently proven in an industrial environment, but has shown results sufficiently promising to warrant further consideration and development. The RTLS was also the most important and most difficult attribute of factory floor state to discover. The position of machines on the factory floor is an essential indicator of the factory floor state in RMS and the use of cost effective (the modules already contained a RF device for communications) RSSI based method of gathering this information lends high feasibility to the use of this hardware-supported RMMS concept. The method of multilateration for localisation relies on a distance estimation, which in the developed technique is gathered using RSSI. The basic details of RSSI measurement are contained in section 3.6.3 with a more in depth analysis in the next section. The section after (6.3) discusses the development and analysis of the developed RTLS and RSSI use.

6.2 RSSI-Based Distance Estimation

The accurate estimation of distance between the node (a SCM) and each beacon (a BPM) is fundamental to the use of the multilateration algorithm, or any triangulation-style localisation technique. The development of a rugged and reliable distance estimation based on RSSI was required for the localisation technique development. The workings of RSSI distance estimation are briefly discussed in section 3.6.3. The most important attribute to note is the predictable logarithmic drop in wireless signal strength observed with an increasing distance travelled.

6.2.1 Built-In RSSI Command

The first concept consideration was to use the built-in RSSI serial output of the XBee modules by requesting it using the ATDB command. An experiment was conducted and is documented in section 10.3. The experiment successfully confirmed the logarithmic drop off but did not show sufficient reliability or consistency to be used as a distance estimation technique. The reason for the unreliability, as discussed in the experiment documentation, was concluded to be due to the sampling and type of output given by the XBee module. The resolution of the physical output was too low to be effective. The resulting experimental data's unusable nature lead to more research into the nature of the RSSI from the physical pin built into the XBee. The output only allowed for steps of 1 dBm, over a range of 22 steps, this was a very low resolution and the implications are discussed in more detail in the experiment's discussion sub-section.

6.2.2 RSSI Using PWM Signal

The XBee modules come with a built in RSSI pin, which outputs a PWM based on the unprocessed received signal strength of any message. The RSSI pin on the XBee module (pin6), according to the data sheet (Appendix E, section 15.5), has a resolution of 2400 counts over a 200ms signal. That is, the PWM signal output of the pin has 2400 different lengths representing the strength. This is a significantly higher resolution than the output of the ATDB command used in the first experiment. These counts happen in intervals and can be detected using the *pulseIn()* library command in Arduino, as seen in the following code snippet (Figure 32):

```
rssiVal = pulseIn(rssi, LOW, 200); //gathers RSSI
```

Figure 32: PulseIn Code Snippet

This code records the duration of the PWM high from the XBee and saves it to a float (cast earlier in the code). For the RMMS's RTLS, the code in Figure 32 was run on all SCMs. The BPM was designed to broadcast its unique character and each SCM receives these characters and does calculations based on the *rssiVal* to estimate the distance from each BPM. This allowed the BPMs to be extremely simple, with minimal processing requirements.

The second experiment used this method of gathering RSSI to further test the feasibility of a RSSI-based RTLS. The experiment is documented in section 10.4 and shows very positive results. The feedback of signal strength gives a very accurate, straight line mapping of distance to signal strength. This mapping implies that the inverse should be easily applicable and that distance should be able to be accurately gathered using signal strength. That was the focus of the next (third) experiment.

6.2.3 Using RSSI To Measure Distance

The third experiment used the average relationship gathered in the experiment shown in section 10.4. That experiment produced the following linear relationship between distance and signal strength (where y is the signal strength, in ms, implied by the distance between the XBees, x):

$$y = 9x - 5.46 \quad [\text{Eq. 12}]$$

This can be rearranged and made clearer to give the following estimation of distance when a PWM ms signal strength is recorded:

$$\text{distance} = \frac{\text{RSSI} + 5.46}{9} \quad [\text{Eq. 13}]$$

For reasons discussed in the next section (6.3), a constant was added to this to ensure the success of the multilateration algorithm 500mm was chosen for this purpose to provide some offset without skewing results too much. The MCU used millimetre values to give better resolution, this produced the equation used in the RMMS's RTLS ([Eq. 14]).

$$\text{distance} = \left(\frac{\text{RSSI} + 5.46}{9} \times 1000 \right) + 500 \quad [\text{Eq. 14}]$$

For the third RSSI experiment, the remote XBee was loaded with code (shown in the experiment documentation in section 10.5) which returned distance estimation when prompted. The results showed that the distance estimations fall into a reliable band of outputs. In the six distance estimations for each of the distance marks there was a maximum estimation variation of 1411 mm which proved, along with the multilateration experiment (10.6), that the RSSI distance estimation could be used for the positioning system to the satisfaction of the DBSCAN algorithm. The average over the course of nine runs proved to be well within the accuracy range needed. Even so, to improve the performance of the distance estimation later tests with the multilateration algorithm (discussed below) implemented ten, rather than five, samples. This was done to help improve the accuracy by making outliers less of an effect.

6.3 Multilateration Localisation

The multilateration technique of localisation is a form of triangulation, the basics of which are described in section 3.6.3. Multilateration is an especially good choice because the SCMs will, by requirement of a factory floor, be contained within a multilateral shape bordered by beacons.

Since the factory floor can be considered as a two dimensional surface, the position of a node can be found, in theory and with extremely accurate distance estimations, with only two beacons, as is shown in Figure 33. Distance estimation (between a node and a beacon) produces a circle with a radius which is equal to that estimation, so that the circle passes through the node and is centred on the beacon. The detected position will lie at the intersection of their radii which lies within the floor space (shown as a red cross). Highly accurate distance estimations, which produce the radii, will result in exact position detection (or localisation).

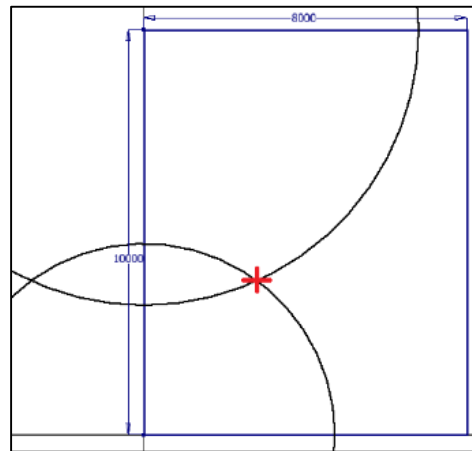


Figure 33: Localisation with Two Beacons

In reality, however, the distance estimations may be inaccurate. The inaccuracies, when using the two-beacon method will cause the estimation to be out by a large margin. The solution is to use multiple beacons with multiple intersection pairs and to average these points into an accurate estimation. When using multiple beacons there are many intersections which are not relevant to localisation. Intersections between diagonally opposite circles and intersections which lie outside of the factory floor area need to be weeded out to get an accurate estimation the relevant intersections are averaged to give the estimated position. Figure 34 shows an example of a four-beacon multilateration system with inaccurate distance estimations with a key which precedes it:

- ◆ Red cross – Actual node position
- ◆ Green circle – Used intersections
- ◆ Blue circle – Rejected intersections
- ◆ Black Dot – Detected SCM position

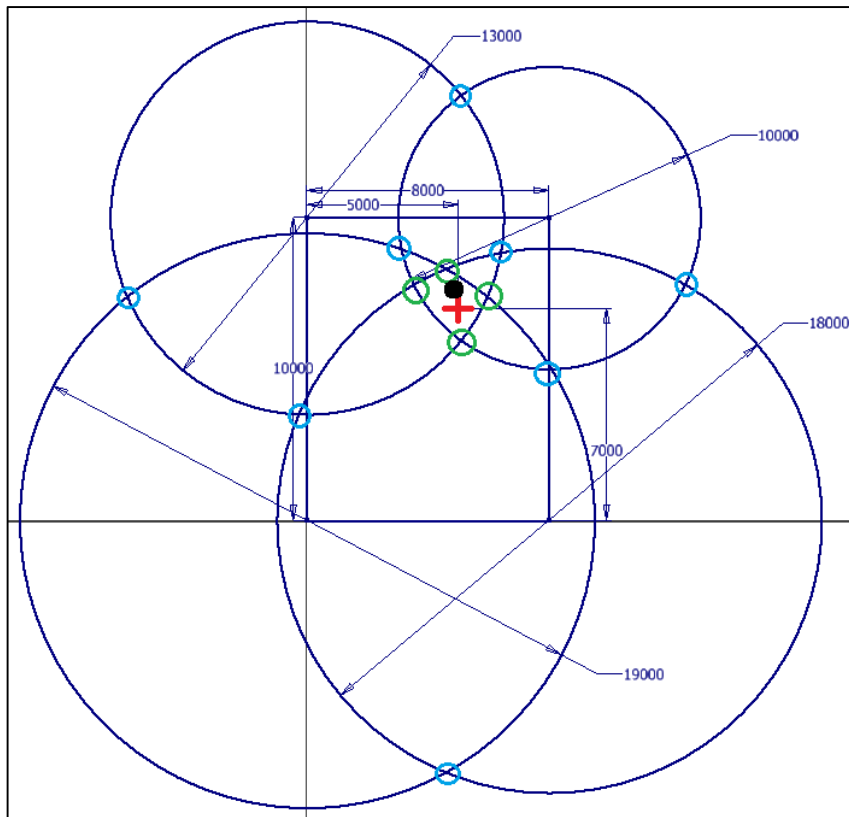


Figure 34: Four Beacon Multilateration

Figure 34 shows that even though the distance estimations are out by a large margin (over 1m), the actual estimated position is less than 0.5m away from the true position. The pseudo-code for this algorithm is shown in Figure 35 and is implemented and tested in the section 10.6.

```

1.   for (each beacon) do
2.       request and receive beacon position as:  $(x_b, y_b)$ 
3.       for (number for average) do
4.           request and receive RSSI ping
5.       end for
6.       find average RSSI
7.       find estimated distance according to:  $r = \frac{pI+5.56}{9}$ 
8.       end for
9.       draw Circles from to:  $(x - a)^2 + (y - b)^2 = p_j^2$ 
10.      find the intersection points of circles (not diagonally opposite)
11.      remove points outside of the factory floor region
12.      find average 2D point and store the estimated position

```

Figure 35: Pseudo-Code for Multilateration

‘Drawing’ (recording the equations of) the circles centred on the beacons and finding the appropriate intersections for all applicable cases was fairly complex, involving lengthy mathematics. The solution to this problem is contained in the following equations ([Eq. 15] to [Eq. 18]) and refers to Figure 36.

The program is designed to only analyse the intersections for beacons which are not diagonally opposite to each other. Referring to Figure 36, it only analysed the following intersections:

A & B; B & C; C & D; D & A

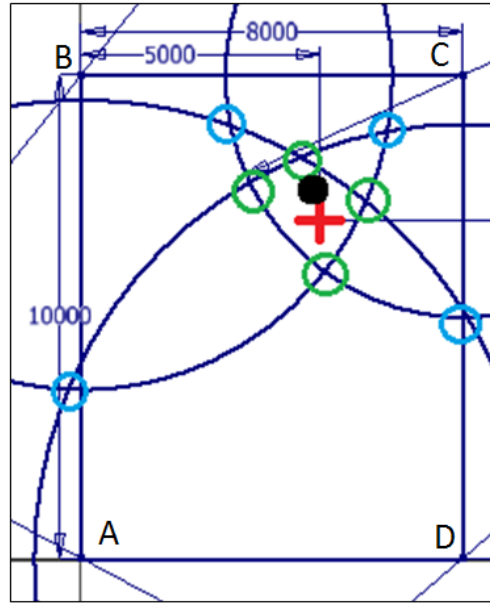


Figure 36: Labelled RTLS Diagram for Equation Generation

This produced eight points, four inside the factory floor bounds and four outside. These points are found using the following equations with the variables are defines as follows:

- ♦ a = X coordinate – beacon 1
- ♦ b = Y coordinate – beacon 1
- ♦ c = X coordinate – beacon 2
- ♦ d = Y coordinate – beacon 2
- ♦ r = distance estimation – beacon 1 to SCM
- ♦ p = distance estimation – beacon 2 to SCM

For the case where the beacons share an X-axis component:

$$x = \frac{1}{2(b-d)^2} \left(2ab^2 - 4abd + 2ad^2 \pm \sqrt{-(b-d)^2 ((b-d-p-r)(b-d+p-r)(b-d-p+r)(b-d+p+r))} \right) \quad [\text{Eq. 15}]$$

$$y = \frac{b^2 - d^2 + p^2 - r^2}{2(b-d)} \quad [\text{Eq. 16}]$$

For the case where the beacons share a Y-axis component:

$$x = \frac{a^2 - c^2 + p^2 - r^2}{2(a-c)} \quad [\text{Eq. 17}]$$

$$y = \frac{1}{2} \left(2b \pm 2 \sqrt{\frac{a(a^2 - c^2 + p^2 - r^2)}{a-c} - \frac{(a^2 - c^2 + p^2 - r^2)^2}{4(a-c)^2} - a^2 + r^2} \right) \quad [\text{Eq. 18}]$$

Each of these cases covers two beacon pairs and each pair will produce two intersections. One intersection lies within the factory floor boundaries and one lies outside. The point which lies outside is disregarded and the point which lies inside the bounds is stored and used for location estimation.

The results of the initial experiment were positive and are discussed in detail in the experiment write-up (10.6). In short, the algorithm showed much promise and thus further development was conducted. The experiment shows an accuracy pass/fail criteria in the results (within a 1200mm span in both dimensions). This criteria was decided on based on the DBSCAN simulation experiment (10.7), discussed in section 8.2.2. This experiment showed that accurate and consistent clustering could be obtained within the 1200mm error range and so the multilateration algorithm had to match this accuracy.

The computer programs displaying the technique are shown in Appendix F (15.6) (C code as contained on the SCM's MCU) and Appendix G (15.7) (Matlab Code as was used to test the technique in the experiment contained in section 10.6).

The multilateration algorithm requires the distance estimation circles to intersect to provide a non-erroneous result. Logic therefore dictates that the circles should rather be over-estimations than under-estimations – this will ensure intersections. If all the estimations are offset by a pre-determined, fixed value, must still give the same result will be produced as if there was no offset, thus the decision was made to offset the distance estimations by a fixed value, 0.5m, as shown in [Eq. 14].

Once the initial experiment was used to test the multilateration algorithm in principle and to find bugs in its implementation, a physical experiment was done to prove the concept in reality and to ensure validity of the technique for the RMMS. The next section, 6.4, discusses how the system was implemented. This experiment was conducted in the laboratory in which the system was designed to run, one roaming SCM was placed in various positions on the factory floor and the RTLS was run. The RTLS estimations were compared to the true positions to gauge accuracy. The results are discussed in detail in section 10.7. The experiment shows that the RTLS managed to satisfy the needs of the RMMS by providing results sufficiently accurate for downstream processing to handle.

6.4 Implementation of the RTLS

The RTLS worked according to the following flowchart (Figure 37). Even though ten RSSI readings are used by the SCMs for each distance estimations, the BPMs were programmed to send their character out twenty times. This was done to make sure that each SCM would receive all the data it needed. Once a SCM has received ten of a BPM's signals it stops accepting that beacon's unique character.

It was decided that multiple RSSI readings would be taken for each BPM so that the effect of any reading variations could be lessened. A delay is included between each of the BPM's unique character broadcasts in order to allow for the readings to be processed and to extend the RSSI reading process over some time to remove possible short-duration anomalous effects. Because the individual readings have proved to be varied (see the experiment in sections 10.4 to 10.6), so the algorithm relies on taking averages to help make the estimations more accurate.

The RTLS was designed to run on the TeensyDuino 3.1 microcontroller board, chosen and discussed in section 5.3.2, but requires interaction between the CRC, SCMs and BPMs.

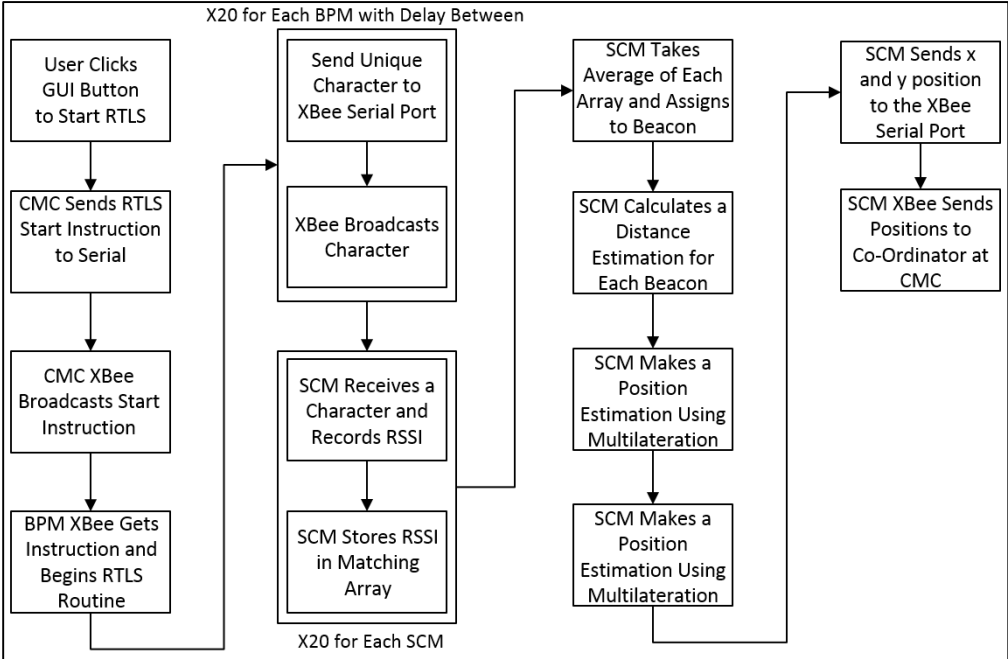


Figure 37: Flow Chart of RTLS Operation

Figure 37 shows that the current implementation of the RTLS does have the caveat of relying on constant and reliable communications, but this disadvantage was outweighed by the ease of use and cost-effectiveness of the RSSI-based RTLS.

6.5 Chapter Summary

The development of a cost-effective means of detecting the location of a module (attached to a machine) on the factory floor was crucial to the RMMS’s success. The area had been researched, but this chapter showed the further development of the concepts contained in literature (see section 3.6.2) into a system developed for localisation on an industrial factory floor. This chapter discusses the advantages and caveats of the RSSI-based multilateration localisation system (including references to multiple experiments documented in this dissertation – sections 10.3 to 10.7), and describes its implementation. A large amount of the research period was spent in the development of the RTLS and its successful design and operation (especially when combined with the processing system described in detail later – sections 8.2 and 8.3) is what makes the RMMS attractive and novel.

7. Chapter 7: Central RMMS Control Software

7.1 Chapter Introduction

This chapter discusses the structure and inner working of the Central Reconfiguration Management and Middleware System Control Software, or CRC. This chapter discusses the development of the CRC, but does not delve into the deeper functionality of its two most important constituent aspects, location data processing and program assignment engine (discussed in Chapter 8). The integration of these aspects, as well as the other supporting sections of the CRC are discussed and so is the programming methodology used throughout the coding of the system. The user interface during operation and complete CRC structure is shown and discussed in section 7.4.

7.2 Structure of the CRC

The CRC is the where all of the data processing is conducted and where the middleware system operates. The CRC communicates directly, via serial wireless to the SCM network. There are several software sub-systems in the CRC software:

- ◆ communication software;
 - converting decisions and instructions into set commands;
- ◆ intelligence for the processing of factory floor data;
 - position-based clustering for cellular layout;
 - intelligence program assignment based on factor floor configuration;
- ◆ SCADA;
 - capable of monitoring the factory floor from network communication;
- ◆ interaction with ERP, MES and control engineers to update production plans, inventory requirements and other functions

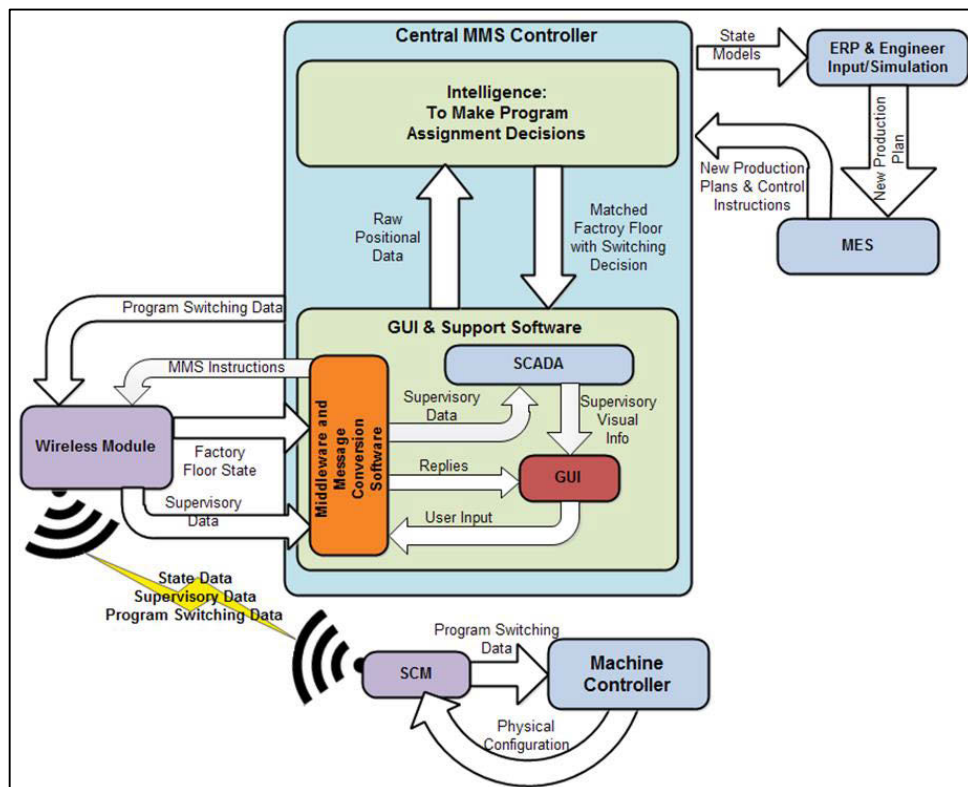


Figure 38: Communications & Software Infrastructure

The most complex parts of the CRC are contained in the block labelled “Intelligence: to Make Program Assignment Decisions” in Figure 38. This process contains two operations, a clustering method (to form cells based on location information) followed by an intelligent method to assign appropriate programs to each machine on the floor based on the cells found by the clustering and configuration information gathered by the SCMs. Two aspects of the CRC, covered in sections 8.2 and 8.3, contain a great amount of novel development and discuss management and middleware in the RMMS.

7.3 Programming Methodology of the CRC

The software was programmed using *MS Visual Studio* in the object orientated *C#* language. The program was designed from the outset to be completely object orientated, with many unique objects defined in classes. The strategy of the RMMS software was to use one parent class, described in the next paragraph to contain all data about the factory floor, with specific methods invoked from within the class itself and from the Graphical User Interface (GUI) classes which receive input from a human user.

The dominant unit in the program is the object of class *Machine*, with many classes (for each different type of machine) inheriting the basic attributes from this base or parent class. Most of the basic functionality of the RMMS can be achieved by using only the objects of type *Machine*, the additional specific machine types (the children of *Machine*) are reserved for use in more specific applications of the RMMS. The *Machine* class contains attributes which are present in every type of machine which could be on the factory floor, listed below. Child classes of class *Machine*, such as the class *Press* (which covers various types of press or stamp machines), contain additional attributes unique to that type of machine. The class *Machine* has the following basic attributes (an explanation of each is given, with the type of variable contained in parenthesis):

- ◆ *machineName* – gives the machine a name to be called by (*string*)
- ◆ *machineType* – type of machine (*string*)
- ◆ *controllerType* – controller architecture (*Controller*)
- ◆ *xDim* and *yDim* – x & y dimension of the machine in mm (*int*)
- ◆ *xPos* and *yPos* – x & y position of the machine on the factory floor in mm (*int*)
- ◆ *hardwareConfigNum* – pre-defined hardware configuration number that the machine is currently set to (*int*)
- ◆ *softwareConfigNum* – pre-defined software routine number that the machine is currently set to (*int*)
- ◆ *softwareConfigs[]* – numbered set of possible software configurations that are known for the machine (array of *int*)
- ◆ *ID*, *unclass*, *noise* – these three are for use by the clustering algorithm (*int*)

The structure listed above shows that all attributes specific to the RMMS (especially the reconfiguration operations) are contained in the class. Further attributes specific to SCADA operation of the RMMS are still to be added to the program and are left for further development of the system. The structure for the *Machine*, *Cell*, and *Layout* classes, as implemented in the CRC, code is shown in Appendix H (section 15.8).

7.4 Complete Structure and Operations GUI

The CRC runs as a GUI on top of the operating system on a MS Windows PC, with mouse-click selections available to a user, allowing for human control of the software. The GUI provides a selection of options for the user to choose from and gives direct instructions to the rest of the software structure. In the future this could be converted to a more automated software which receives instructions from some other software, such as the MES or ERP. The CRC was designed to include a type of intelligence for the autonomous assignment of programs to machine controllers based on machine position and configuration (factory floor state) models. This was designed as a location system processor and an assignment engine – a clustering algorithm (DBSCAN) and an intelligent assignment engine (Knowledge-Based System). This intelligence structure is shown in Figure 39. In the development during this research, the software was designed to have the complete structure shown below (Figure 39), which was an evolution of Figure 9 and Figure 38. A controller operation description follows the figure.

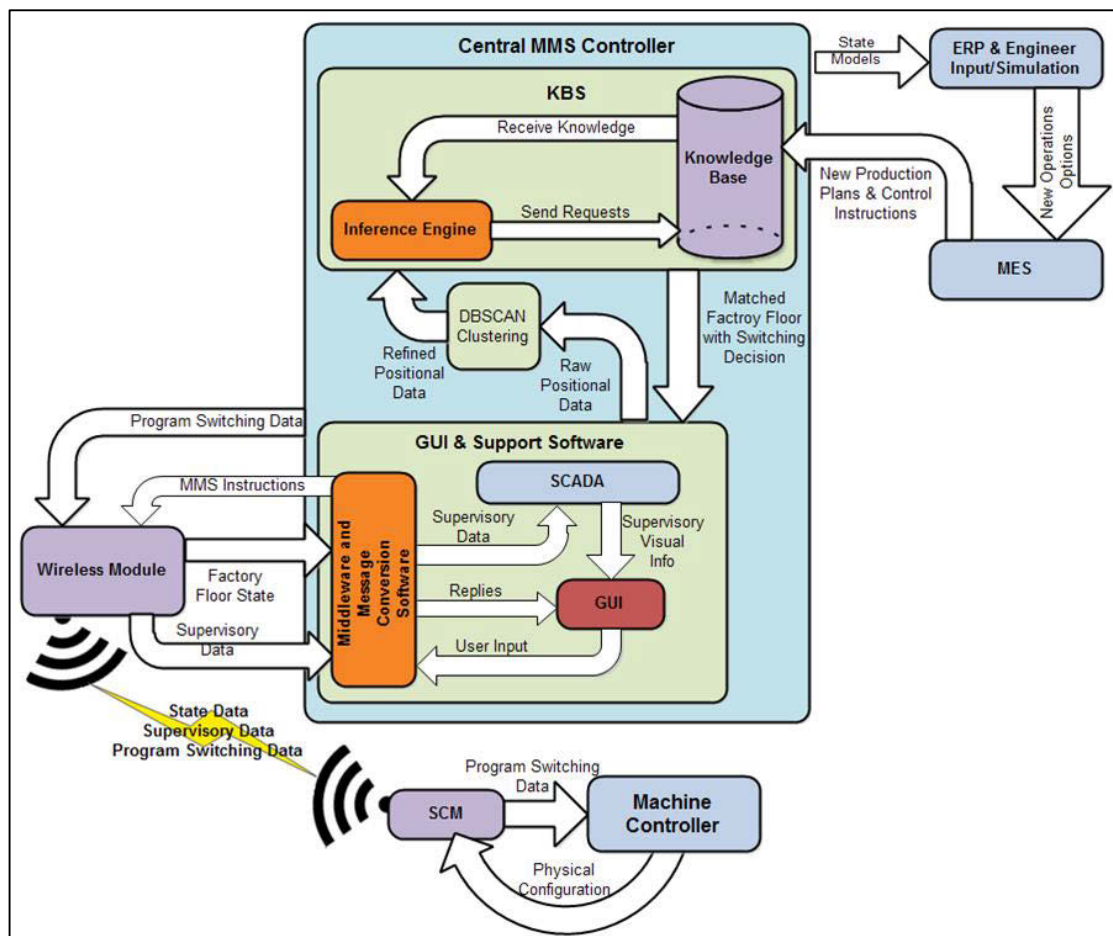


Figure 39: Software Structure of the CMC

The CRC acts as the middleware and intelligence layer for the RMMS. The controller was designed to allow for the user control and management of the reconfiguration process. The software runs over the operating system on a PC, so does not fit exactly within the definition of traditional middleware. As stated before, however, it mimics the functionality of middleware, hence the name. The CRC prompts the user for an instruction, via a GUI (see Figure 40) and allows the user to begin the ramp-up process.

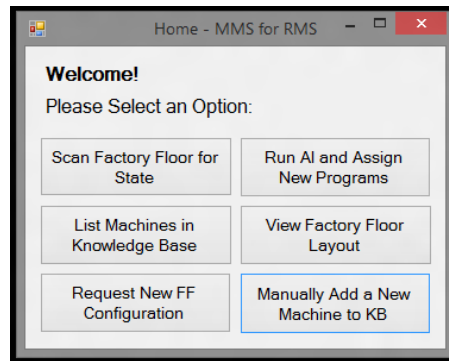


Figure 40: GUI Start Window

The CRC manages the ramp-up process, when prompted by the user, by triggering the RTLS. It does this by broadcasting a RTLS instruction (the character ‘x’, seen by the nodes as the ASCII Hexadecimal 78) to all nodes of the mesh network. The method of RTLS has been discussed before (in sections 3.6.2 and 3.6.3) and was described in greater detail in section 6.4. When the RTLS has completed its estimations, the user is prompted to request the state from the SCMs, as shown in Figure 41. This operation requests a state message from each SCM and creates a *Machine* object with the information. The objects are stored as a *List<Machine>* which represents all the machines on the factory floor. The user is then prompted to run the intelligence portion of the CRC, which converts the individual states gathered from the SCMs into a factory floor model for use in assigning new control programs.

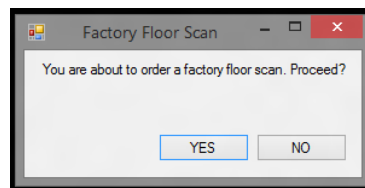


Figure 41: Request State Window

After a RTLS scan is completed and clustering (discussed below) is done by the CRC the following window (Figure 42) appears in order to show the user results. The results appear as a physical factory floor state in text representation.

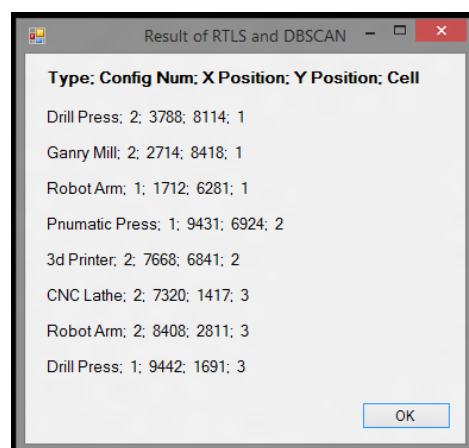


Figure 42: Physical Factory Floor State Window

Once the state is accepted by the user, the program returns to the home window. The next step is for the user to choose the ‘Run AI and Assign New Programs’ option, which brings up the following window (Figure 43).

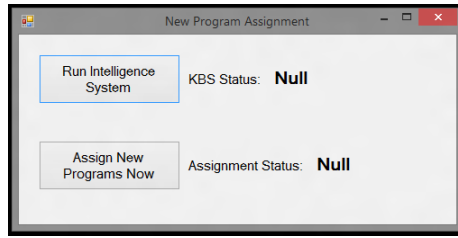


Figure 43: Run AI and Assign New Program Window

When the user selects the ‘Run Intelligence System’ option, the KBS is triggered and the CRC searches the database for a matching factory floor. The window updates to one of the windows in Figure 44, below, depending on whether or not a match for the state was found.

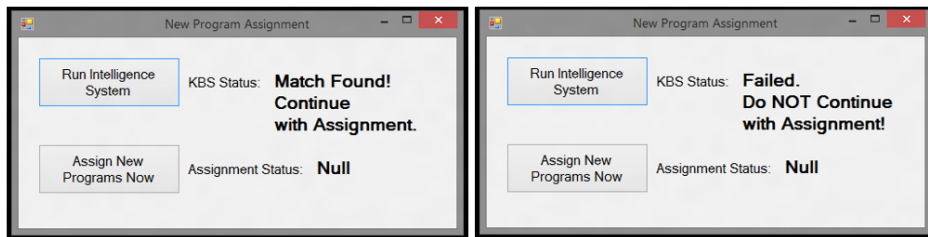


Figure 44: Windows for Successful (left) and Unsuccessful (right) KBS Matches

If the match was unsuccessful, the user must restart the RTLS routine in order to update the data. The user may need to update the knowledge base if the state is totally unknown to the CRC. If the match is successful then the user can run the assignment process, which sends the new software selection instructions to each of the SCMs to be passed to the machine controllers on the factory floor. This process takes a few minutes, so the window updates to show that it is in progress (Figure 45).

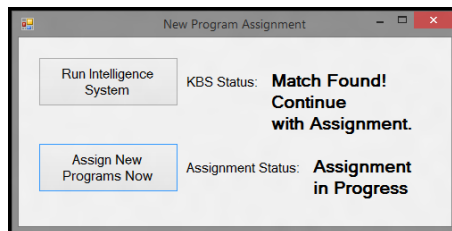


Figure 45: Window Shown During Assignment Process

When the process is completed or for some reason the assignment did not complete successfully, the window updates to show one of the two options shown below (Figure 46). If the assignment was unsuccessful, it was probably due to a communications failure and not a system failure, so the user is prompted to retry the process.

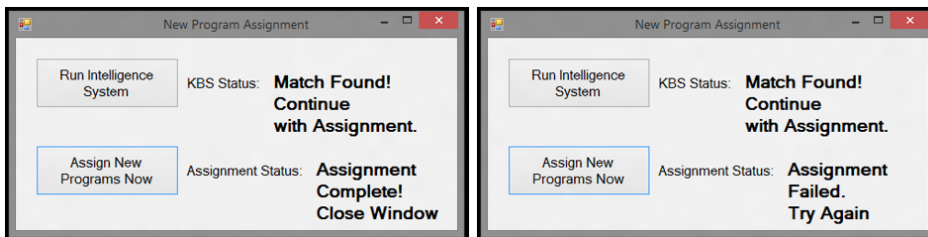


Figure 46: Windows for Successful (left) and Unsuccessful (right) Assignment

7.5 Chapter Summary

This section showed how the different aspects of the CRC interact with each other and how a user would interact with the RMMS during a reconfiguration operation. The programming methodology and software structure were also discussed. The importance of the object orientated architecture used in the coding was discussed. This chapter should have illustrated how the processing system described in the next chapter accepts and digests data from the RTLS and produces new factory floor switching instructions in the context of the RMMS as a whole.

8. Chapter 8: Data Processing and Program Assignment

8.1 Chapter Introduction

This chapter discusses the methods used for the development of a factory floor model for use in the program assignment process described in the previous chapter. Two discrete processes are implemented when processing data gathered on the factory floor; a clustering algorithm is used to create a cellular model of the factory floor model and an artificial intelligence method is then used to process this data into a known and complete model. The complete model contains all the program switching information needed for the re-configuration of control and once developed by the AI it can be sent to machine controllers. The clustering process (section 8.2) and AI used to assign programs to machine controllers (section 8.3) and their design and implementation are discussed.

8.2 Clustering Technique for Processing Data

Some clustering methods were introduced and discussed in section 3.6.6. From the different methods that were considered, the density-based methods were considered the best choice because those techniques are generally simple to implement and deterministic in nature. Also, density-based techniques are generally capable of identifying non-spherical (or circular in the two-dimensional case) clusters, which is important for cell structure discovery. With density based clustering, the number of dimensions in the data determines much of the complexity of the density-based methods, so the 2-dimensional floor position data is well suited.

There are several density based clustering alternatives available, with Density-Based Spatial Clustering Applications with Noise (DBSCAN) being the most commonly implemented [84]. Most of the other methods such as Ordering Points to Identify the Clustering Structure (OPTICS) and Entropy-DBSCAN (EnDBSCAN) are variations of the DCSCAN algorithms. The OPTICS technique produces almost no noise, usually forcing a wayward point to be included in a cluster and the EnDBSCAN method uses an entropy-based judgment of distance rather than a Euclidian measure, making it well suited for uncertain data [102]. Each of the variations, especially OPTICS have their advantages, but the simplicity of the DBSCAN method, combined with good experimental results (shown later) and a large amount of pseudo-code and guidance available in the literature makes it a good choice for the clustering of location data.

DBSCAN works by finding clusters of data points which have a much higher density, in a Euclidian definition, than the surrounding area. This method is very similar to the natural, human way of recognising clusters [103]. This method aims to be able to identify between true clusters and noise and to be able to create clusters which are not geometrically near-circular (two-dimensionally spherical). DBSCAN is a technique which uses two variables, EPS-neighbourhood (EPS) and minimum points (MinPts). The EPS governs the size of the neighbourhood of each point and can be visualised as the maximum distance between two points for them to be considered in the same cluster. MinPts is the noise identifier; it defines the minimum number of data points needed to form a cluster. For example, if the MinPts variable is three and two points are within each other's EPS, but not within the EPS of another cluster or point, the points will be considered noise [103]. Figure 47, below, shows an example where there are four clusters of points and some noise, with the MinPts parameter set to three.

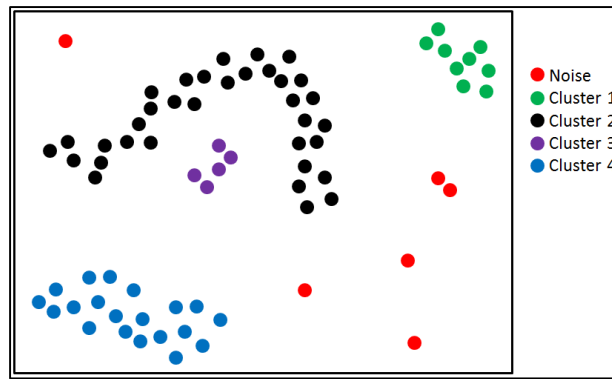


Figure 47: DBSCAN Result Map Showing Noise

The results clearly show that DBSCAN is capable of identifying complex, non-spherical clusters. This map is a typical example of a point map which cannot be solved by other, non-density-based methods of clustering, but which is efficiently found using a density-based method. The case in Figure 47 shows a map which is not particularly similar to a typical factory floor, but which illustrates the strengths of the algorithm.

One of the caveats of the DBSCAN algorithm is a tendency to link up different clusters when a thin link of positions, which may be considered noise, is between two distinct clusters. This can usually be solved by choosing a smaller EPS value and decreasing the MinPts value so that the particles between are assigned their own cell. This phenomenon is shown in Figure 48 with each of the three cases illustrating the effect of EPS and MinPts variables.

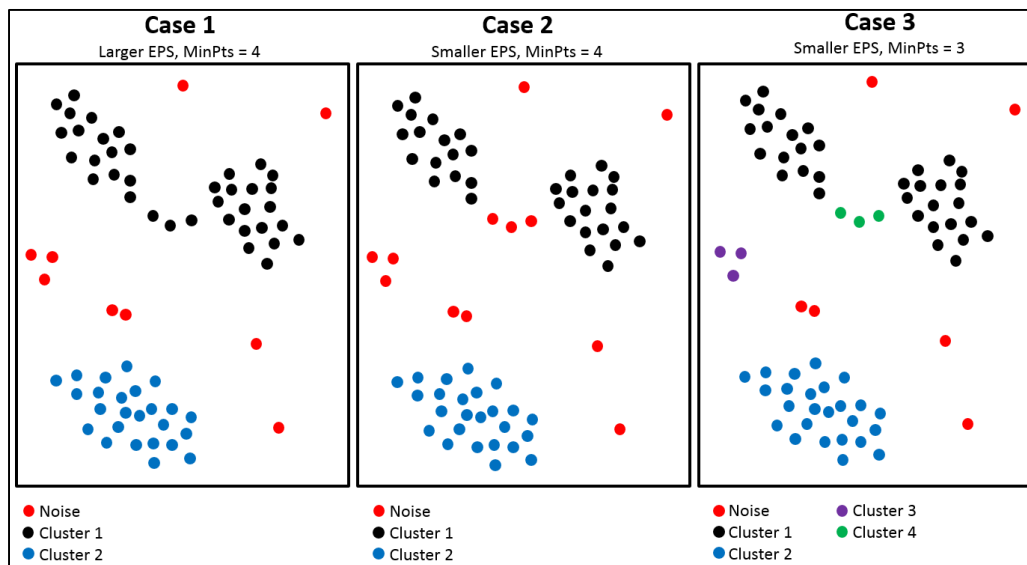


Figure 48: DBSCAN Linking Separate Clusters

For the application in the RMMS, machine clusters are likely to be low in number (≤ 6). The processing variables for the RMMS had to be designed appropriately and the variables chosen to match the application in order to give sensible results. These choices are given, discussed, and supported in 8.2.2.

8.2.1 Workings of the DBSCAN Algorithm

The details of the working of DBSCAN are complex, with a reasonably simple implementation. The paper by Ester et al [103] contains an in-depth explanation of the DBSCAN algorithm and mathematical workings and assumptions within the algorithm for further reading. This paper is

also the source of the outline of the pseudo-code examples contained in Figure 49 and Figure 50. This work and pseudo-code was also the guide for the final version of the code used in the CRC, which is contained in Appendix I (15.9). The general case pseudo-code for the DBSCAN algorithm is as follows in Figure 49 (the Main method), and Figure 50 (the ClusterExp method) [103].

The ClusterExp method used in line 9 of the pseudo-code in Figure 49, above, is the most important function of the DBSCAN algorithm. The pseudo-code for this function is outlined below (Figure 50) and returns a Boolean true/false value. In the ClusterExp code, the call of regionQuery returns a list of other points in SetOfPoints which share the EPS neighbourhood of the associated point.

An experiment was conducted to test the effectiveness of the DBSCAN algorithm in clustering a simulated factory floor – see section 10.7. The conducted experiment aimed to find the most effective MinPts and EPS variables for the factory floor and to gauge the absolute effectiveness of the algorithm. This experiment proved the effectiveness of the DBSCAN algorithm in clustering location data prone to some inaccuracies, as is typical of the RTLS (see section 10.6).

The DBSCAN algorithm was adapted for both initial testing (discussed in detail in section 10.8) and for use in the actual CRC, where it was made object orientated, discussed in section 8.2.2.

```
1. EPS and MinPts are defined
2. Main Method
3.   SetOfPoints is an array of dimensions (number of points and dimensions of data)
4.   SetOfPoints has no classification of ClusterID
5.   set ClusterID equal to NextID(noise)
6.   for (i from 1 to size of SetOfPoints)
7.     set P to ith element of SetOfPoints
8.     if (P's ClusterID is unclassified)
9.       if (ExpandCluster(SetOfPoints, P, ClusterID, EPS, MinPts) is true) do
10.        set ClusterID equal to NextID(ClusterID)
11.      end if
12.    end if
13.  end for
14. end Main method
```

Figure 49: Pseudo-Code for DBSCAN

```

1. Start ExpandCluster
2.   Set array Seeds equal to regionQuery
3.   if (size of Seeds is less than MinPts) do
4.     set currentP's ClusterID to noise
5.     return FALSE
6.   else
7.     set the ClusterID of point currentP current ClusterID
8.     delete the currentP from Seeds
9.     while (Seeds is not empty) do
10.      set first point in Seeds as the currentP
11.      set the array Result to currentP's regionQuery
12.      if (size of Result is less than MinPts) do
13.        for (i from 1 to size of Result) do
14.          set resultP to first element in Result
15.          if (ClusterID of resultP is unclassified or noise) do
16.            if (ClusterID of resultP is unclassified) do
17.              add resultP back into Seeds
18.            end if
19.            set the ClusterID of each point in Seeds to the current ClusterID
20.          end if
21.        end for
22.      end if
23.      delete the currentP from Seeds
24.    end while
25.    return TRUE
26.  end if
27. end ExpandCluster

```

Figure 50: Pseudo-Code for DBSCAN Method: ExpandCluster

8.2.2 Adapting DBSCAN for the RMMS

The general form shown in the two pseudo-codes had to be adapted to suit the central control software for the RMMS. It had to be made to handle the specific requirements of factory floor cell clustering and for handling the data it had to work with – the location attributes ($xPos$ & $yPos$) of a *Machine* object in the program. It had to use the variables found in the DBSCAN simulation experiment in order to cluster machines into cells effectively. The DBSCAN algorithm implemented in the RMMS used the variables $EPS = 4750\text{mm}$ & $MinPts = 2$, as was found in the experiment documented in section 10.8.

The DBSCAN code as implemented in the CRC is contained in Appendix I (section 15.9) the program was adapted from the pseudo code shown in Figure 49 and Figure 50 by creating a list of machine objects in the form *List<Machine>*. This list is created from the data gathered by the state discovery operation (the information passed on from each SCM to the CRC after the RTLS operation). The machine objects are then clustered according to their positions in the same way as the pseudo-code clusters the points by theirs.

A list in the form *List<List<Machine>>* is created in the program, which assigns an object for the storage of clusters (clusters are lists of machines, thus this object was created as a list of clusters. The list of clusters (list of machine object lists) is what is fed to the KBS for processing. The DBSCAN method returns a *Layout* object, which is created by using the list *List<List<Machine>>* which was returned in each use of the *getClust* method. In this way, each list of machine lists takes the form of a *Layout*, using the class constructors. This *Layout* object

is used by the intelligent assignment method, discussed below (see Chapter 8.3), to find the factory floor state and find the correct software routines.

8.3 A Knowledge-Based Method for Program Assignment

The information gathered by the RTLS and processed into cells by the DBSCAN algorithm needs to be combined with the physical configuration of each machine (gathered by the SCMs) to get the factory floor's physical state. The factory floor's physical state then needs to be assigned the appropriate software, in the form of software switching instructions sent out to the machine controllers. The chosen artificial intelligence strategy for this research was the Knowledge-Based System, or KBS, which was introduced in section 3.6.6.

It is this essential process of assigning programs to machine controllers (in the form of switching between known routines) which makes up the management aspect of the system developed in this research. The process of re-establishing control for the ramp-up of a manufacturing system needed to be made more rapid and more automated in order to increase the reconfigurability of the system [27] and the system's ability to be more reactive because with a shorter ramp-up time reconfigurations can be economically conducted more often. The management of this process by a form of software control system was a primary goal of this research and the software switching process, governed by the KBS discussed here, was the final step in re-establishing control and reducing the system's ramp-up time.

8.3.1 Inference Engine of the KBS

The KBS uses an inference engine and a base of known facts. In this system, the knowledge base is a database of *Layout* objects, each, in turn, made up of *Cell* and *Machine* objects. There are equivalence test methods built in to each of these three objects, named *sameAs*. The KBS's inference engine was designed to present the *Layout* as an argument to the *sameAs* method of the inference engine. The inference engine checks for equality as shown in Figure 51.

The equivalence methods mentioned above and used in the pseudo-code (Figure 51) feed in to each other in the implemented CRC in order to match the pseudo-code in an object orientated fashion. The true inference engine code is shown in Appendix J (15.10) and uses object orientated methods to provide the pseudo-code's functionality – the inference engine of the KBS only calls the *sameAs*, feeding it each known *Layout* as an argument for comparison with the true layout found by the RTLS. The *sameAs* method within the *Layout* object in turn calls the *sameAs* method, handing it each *Machine* object in the layout as an argument. The *sameAs* method in the *Machine* object calls for equivalence in each of the *Machines* to confirm its own equivalence. *Machine* objects are considered equivalent if the two are of the same machine type and are in the same physical configuration. So that a two layouts are equivalent if and only if the layouts contain the same number of cells and that each of those cells contain the same number of machines and that each of the machines in each of the cells are equivalent.

```

1. Receive true layout from DBSCAN
2. Declare Integer cellCheck & machineCheck
3. Declare Boolean flag and set to false
4. Foreach (Layout in the Knowledge Base)
5.   If (current layout has the same number of cells as the true layout)
6.     Foreach (cell in the current layout)
7.       If (current cell has the same number of machines as true layout's cell)
8.         Foreach (machine in current cell)
9.           If (current machine is equivalent to true layout's cell's machine)
10.            Add 1 to machineCheck
11.           End if
12.         End Foreach
13.       If (machineCheck is equal to number of machines in true layout's cell)
14.         Add 1 to cellCheck
15.       End if
16.     End if
17.   End Foreach
18. End if
19. If (cellCheck is equal to number of cells in true layout)
20.   Make flag true
21.   Set a current layout object to be equal to the matching object from the KB
22. End if
23. End Foreach

```

Figure 51: KBS Pseudo-Code

8.3.2 The Knowledge-Base

The KBS was designed to draw a knowledge base from a locally stored file, in the form of a comma-separated value (CSV) file, which is created in MS Excel. The CRC contains code which retrieves values from the CSV file and converts them into layout objects.

The KBS thus provides a method for the assignment of programs to machine controllers, based on factory floor state. The inference engine has the aim of finding a known factory floor state which matches the findings of the factory floor scan. The factory floor scan does not contain all the information necessary for the assignment of programs – the complete factory floor states are only known in the knowledge base. Thus, once a matching floor is found in the knowledge base the full details can be inferred to the incomplete state which was discovered by the first stage of the RMMS process. The known state found in the database, which matches the findings of the scan, contains all the software switching instructions necessary for the new programs to be assigned. Once the new assignments have been inferred, the RMMS sends the appropriate assignment number to each of the SCMs on the floor as discussed below.

8.3.3 Program Assignment Strategy

In order to complete the process of reconfiguring the control of an RMS, messages had to be sent from the CRC to each module on the factory floor. The messages had to be configured to contain the SCM/Machine Controller's name and the program to which it had to switch. The switching of programs is the essential outcome of the system, it is process which updates the control at a factory floor level.

The process is handled by the CRC, when the KBS has found a matching factory floor state it places a pointer to its position in memory and essentially stores the matched, knowledge base state as the current form of the factory floor. The CRC then accesses the program assignment data stored in the matched state. Contained in the knowledge base is the program needed for each

machine by the state. The CRC draws a machine name and its required program number and broadcasts it to the factory floor one-by-one. Each message is then received by each SCM, but only the one which is connected to the named machine reacts and sets its output to the appropriate binary combination. Once each message is sent out to the machine controllers, the process is complete and control is reconfigured for use in the new production period.

8.4 Chapter Summary

The processing of raw factory floor data, gathered during the operation of the RTLS, into software switching instructions is the core function of the RMMS. It is this processing which allows a physically reconfigured factory floor to reconfigure its software with minimal human interference. The development and testing of the methods chosen for this duty were discussed in detail in this chapter, the DBSCAN algorithm, Knowledge-Based System and program switching strategy.

9. Chapter 9: Other Design Aspects of the RMMS

9.1 Chapter Introduction

In this chapter, the physical design of the SCMs are discussed and the possible expansion of the RMMS to include SCADA is discussed in more detail than in earlier sections. The nature and implementation of the wireless network is also discussed. This chapter discussed design aspects of the RMMS which do not warrant their own chapter. These RMMS aspects are no less important than the others in the previous chapters, but did not involve the same level of design work, experimentation and discussion as the other sub-systems.

9.2 Implementation of the Wireless Network

The wireless network serves a dual-duty as the primary line of communication and real-time location system. The wireless Zigbee communication protocol was used to provide links between the modules and central RMMS controller and to provide distance estimations for use in the RTLS. This section discusses the nature of the Zigbee wireless network and its implementation in the RMMS.

The Zigbee protocol, using the XBee Series 2 modules was chosen for use as the wireless communication in the RMMS (discussed in section 5.3.1). Specifically, these modules made use of the ZigbeeMesh protocol, which operated on the IEEE 802.15.4-2003 Zigbee specification. The operation of this specification was discussed in the following paragraphs.

A major advantage of using the XBee Series 2 modules (see section 5.3.1) for the Zigbee wireless communications is the ability to facilitate a mesh network. The ZigbeeMesh protocol is a three part system, where Zigbee modules are configured to behave as *Co-ordinators*, *Routers*, and *End Devices* depending on the required mesh topography [96]. A single co-ordinator must always be present in a network, but the rest of the mesh network can be made up of any combination of the other two types of device, depending on the system needs.

A *Co-ordinator* forms the network and assigns addresses to the other network devices. Once the network is formed, the co-ordinator controls data routes and keeps the network healthy. The *Routers* in the network are joined to the network and can send and receive messages of their own as well as pass messages on to the rest of the network. Both routers and co-ordinators must be active at all times. An *End Device* is the simplest element of the network, only capable of sending and receiving messages and not capable of routing other message. An end device consumes the least power because it can be put into sleep mode when not in use [96].

The network of many devices can be arranged, depending on the module configuration make up chosen in three basic topologies, *Star*, *Mesh* and *Cluster Tree* as well as various off-shoots of these designs. The following figure suggested by Faludi in *Building Wireless Sensor Networks* [96], illustrates the different topologies (Figure 52).

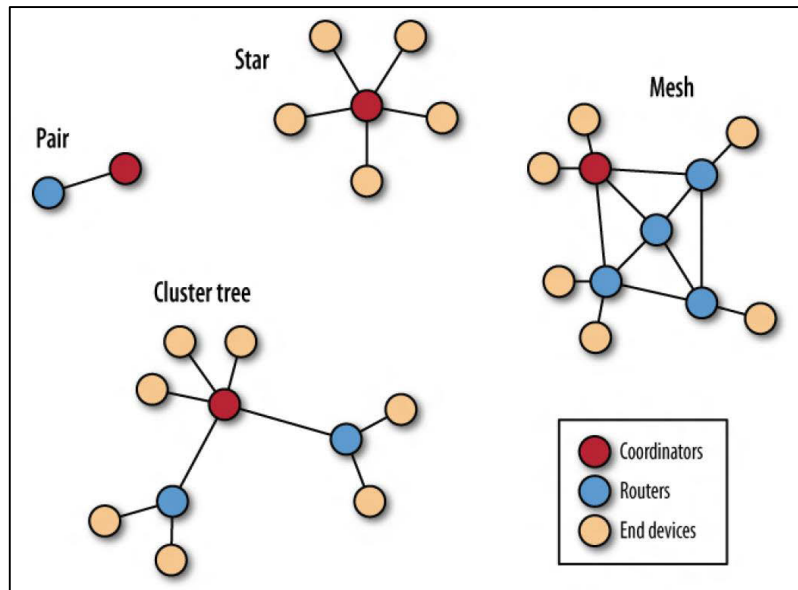


Figure 52: Zigbee Mesh Network Topologies [96]

The RMMS mesh network is made up of the CRC, SCMs and BPMs. The CRC is connected to the co-ordinator device, each of the SCMs is an end device and BPMs are routers. The topology is of a Mesh structure, but with the routers around the outside, linking inwards with end devices. This combination of devices means that a path from any SCM to the CRC is always available, provided the SCM is within the area bounded by BPMs and that the BPMs are spaced within the working range of the XBee modules – these are prerequisites of the system.

The illustration below (Figure 53) depicts a few typical data paths, the two red paths show that, depending on the network traffic load, different paths may be used. One way arrows (depicting communications from the SCMs on machines to the CRC) are shown in the picture, but similar paths are used for communications in the reverse direction.

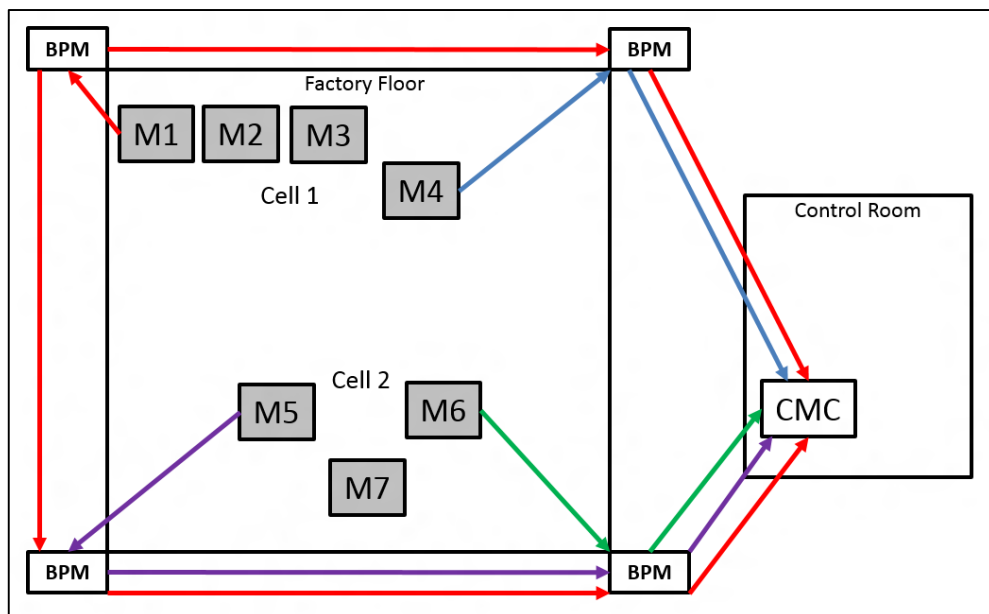


Figure 53: Example MMS Mesh Network Paths

The red arrows in Figure 53, as mentioned before, show that multiple paths are possible in the wireless network. There are multiple reasons, all controlled by the network co-ordinator XBee that longer paths may be taken, usually due to the load of traffic on a particular path.

Making both SCMs and BPMs router type devices would give more paths to the network, making it more stable, and this was a considered option. This was decided against because of the specific communications needed during the positioning operation. The positioning requires direct communication between each of the BPMs and each of the SCMs, if all the SCMs were made to act as routers, a direct path between all BPMs and SCMs would not occur.

9.3 Physical Design of the SCM

The physical design of the SCM and BPM are guided by the electronics that need to be housed and by the cooling needed to account for voltage regulation. The design process is unimportant in the scope of the project as a whole, so this section has been made short and concise and only contains the overview of the final design. The design had the following constraints:

- ◆ contain completed PCB and components (98×92×25mm);
- ◆ contain 4×AA battery (55×16×48mm);
- ◆ contain a 50×50mm brushless fan to provide airflow cooling;
- ◆ must be small enough to print on the UP! Plus 3D printer;

It was decided early on that the containers would be printed using the *UP! Plus* 3D printers in the Mechatronics laboratory at UKZN Mechanical Engineering to create accurate prototypes. This created a limitation to size, with a printable volume of 135×140×140mm. This size limitation, along with the wall thicknesses (set to 4mm to be strong but still easy and economical to print) and housing requirements led to the designs shown in the pictures of CAD models in Figure 54 and Figure 55.

The case was designed in two parts, box and lid. The box was designed to house the components so that there was no interference with each other and provided mounting points for the fan, which was configured to push air into the box. This push configuration was chosen to provide positive air pressure which would force air out of vents left in the lid and other holes in the box which allow for cable input and output. This means that for a dusty environment a dust filter could be fitted to the fan vent and dust should not be a major problem. The same is not true for a configuration where the fan drives air out of the box, where dust can enter through all the gaps which draw air due to the negative pressure in the case. The fan mounting point at the bottom and direction of flow were the reason for the addition of small feet at the bottom of the box – to allow for the draw of air. The lid was designed to be screwed to the box, so holes were placed in the lid and screw holes were printed into the four corners of the box – so no drilling would be needed for assembly. The lid was designed with vents for airflow and a hole for the antenna of the XBee module inside. The vents were placed near the voltage regulators and the microcontroller mounting position to provide cooling. Even though the current was designed to be low, there is always a heat output expected when a voltage is regulated, so the cooling was added as a precaution to avoid component failure due to overheating.

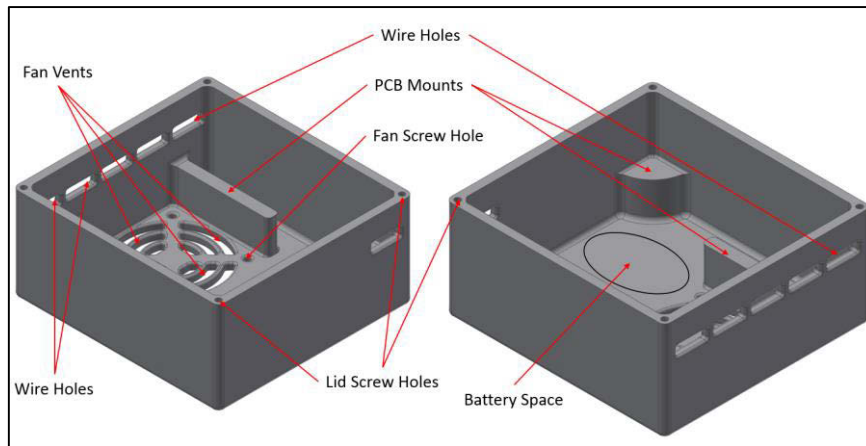


Figure 54: CAD of Box

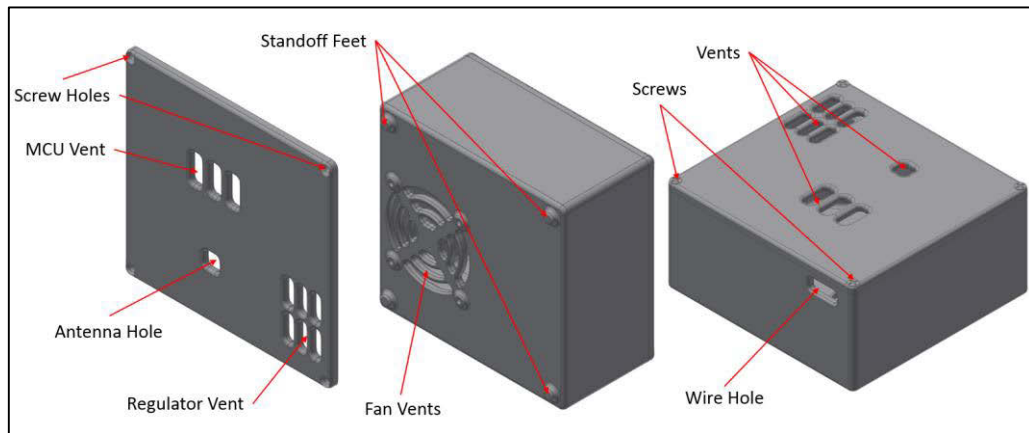


Figure 55: CAD of Lid (left) and Assembly (middle & right)

Figure 56 and Figure 57 shows a CAD assembly of the box and its components in various states of construction.

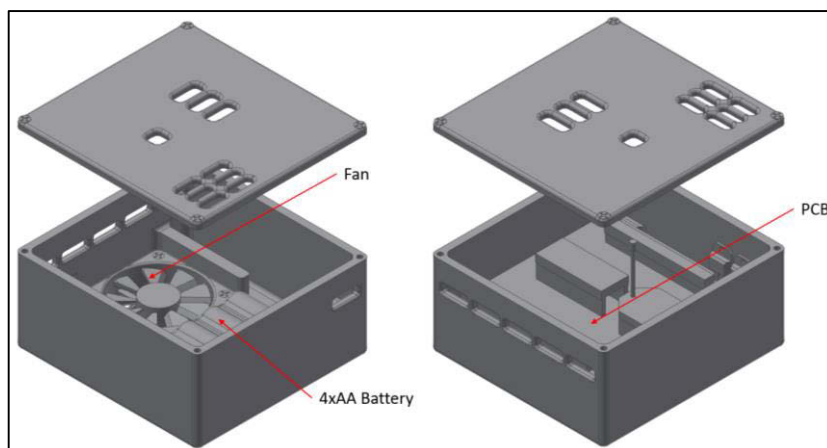


Figure 56: CAD of Case Assemblies (left without PCB, right with)

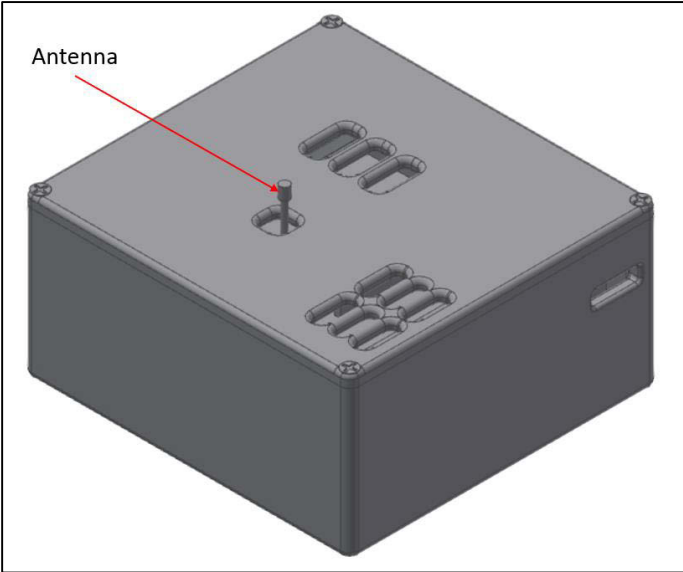


Figure 57: CAD of Closed Case Assembly

Figure 58 shows the box in reality, connected to a machine controller, with the lid off.

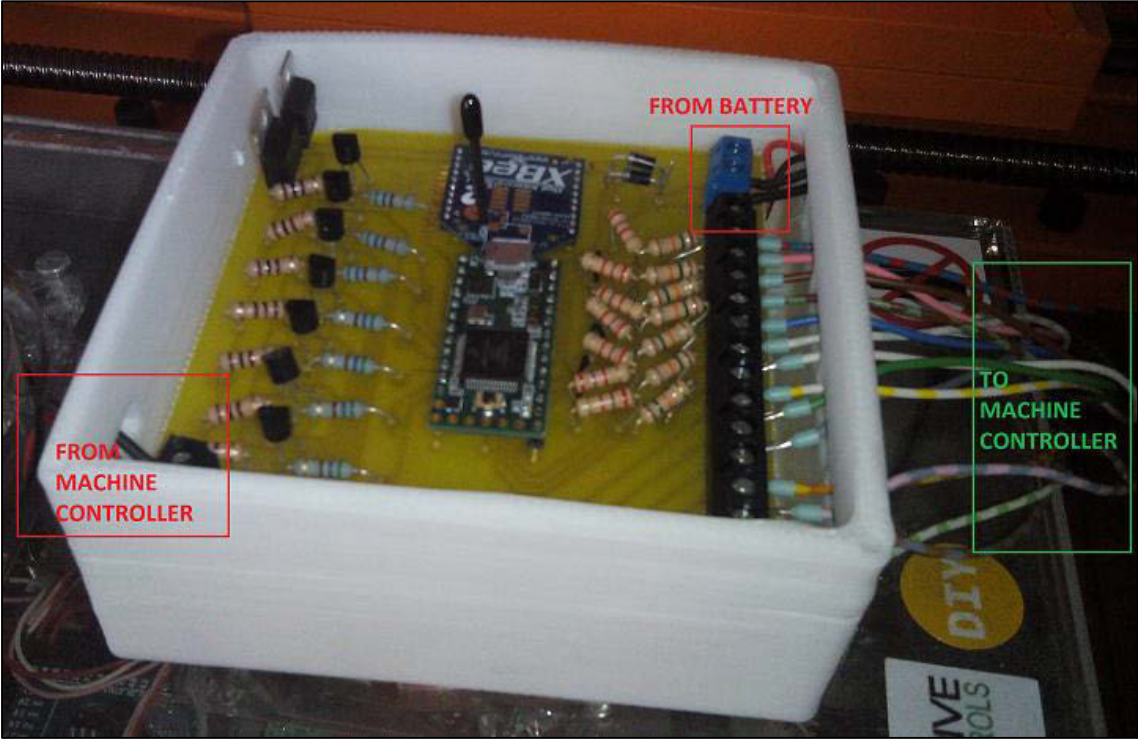


Figure 58: Photograph of the Module (Lid Removed)

9.4 SCADA in the RMMS

A simple pseudo-code example of what runs on the SCM for this is shown below (Figure 59). It is clear that the SCADA operation of the RMMS requires additional programming and setup for a unique RMS or machine, but this is no more intensive than the setup of traditional SCADA systems. The pseudo-code shown here is discussed after.

```

1. if (serial port reads "SCADA") do
2.   if (pin14 is high & timer start bit is low) do
3.     start timer
4.     set timer start bit to high
5.     if (pin15 is high) do
6.       stop timer
7.       set timer start bit to low
8.       increment part counter
9.     end if
10.  end if
11.  if (pin7-pin13 reads [error patterns]) do
12.    send error message on serial port
13.    wait 5s
14.    if (serial port reads "EStop") do
15.      set pin0-pin6 to: [0 0 0 0 0 1]
16.    end if
17.    if (serial port reads "SStop") do
18.      set pin0-pin6 to: [0 0 0 0 1 0]
19.    end if
20.  end if
21.  if (pin16 is high) do
22.    set pin0-pin6 to: [0 0 0 0 0 1]
23.    send critical error message on serial port
24.  end if
25.  send part count and time to serial port
26. end if

```

Figure 59: Pseudo-Code for SCADA Operation

This information was designed to be communicated along additional I/O pins and monitored by the SCM. For example, the machine controller is programmed to set one of its pins to high for 500ms every time a part is started and each time it is finished. The SCM is programmed to, when in SCADA mode, read this as a start and completion for a part and time the duration of the part operation as well as what number part it is in the batch.

It is left to further development to build upon the idea displayed in the pseudo code to provide full-fledged SCADA functionality to this RMMS. This only serves as a guide and a visualisation of how SCADA can be implemented in the SCM-based RMMS.

9.5 Chapter Summary

This chapter discussed the facets of the RMMS which did not have sufficient work for their own chapters. The nature of the XBee wireless network implemented in the RMMS discussed and the configuration of each module type used in the system was described. The physical design was expanded from the constraints laid out in section 3.6.7 and the final prototype of the module and its housing were shown. The section on SCADA introduced a possible strategy for the implementation of SCADA in the RMMS, while direct programming was left for future work.

10. Chapter 10: Sub-Systems Experimentation

10.1 Chapter Introduction

This chapter describes the experiments used to supplement the design process in section and should be consulted during its reading. Experiments were conducted both to guide the design process, as was the case with distance estimation experiments and to verify that the designs would perform within the performance requirements of the RMMS and other subsystems. The experiments on distance estimations (sections 10.3-10.5) and the first part of the DBSCAN experiment (section 10.8) were used to guide the design process and the others were primarily verification tests, although their results helped guide the design of other facets of the system.

10.2 Testing the Electronics of the SCM

This section of experiments contains three sections, each to test one aspect of the SCM electrical design: voltage step-up for communication from SCM to machine controller, voltage step-down for communication from machine controller to SCM and voltage regulation for power supply using a machine controller as the source.

10.2.1 Voltage Step-Down for Communication

The SCM was required to buck the voltage of signals coming from the attached machine controller in order to safely receive messages (discussed in section 5.4). The method had to step down the voltage supplied by a machine controller to one

Aim:

This experiment aimed to test the effectiveness of the electrical channel step-down design in reality and determine if the results were suitable for the SCM.

Apparatus:

- ◆ Prototype SCMs with 24V, 18V, 12V & 5V conversion resistor pairs,
- ◆ Variable DC voltage supply,
- ◆ PC with Arduino IDE and MS Excel,
- ◆ Voltmeter.

Method:

A two-step verification was used to check the effectiveness of the step-down conversion. The outputs of the resistor pairs were checked with a voltmeter and a pass/fail test was done in the form of a verification routine on the Teensy 3.1. The routine was written so as to illuminate the on-board LED when it received the correct input on its pins (101010). The following was done for each of the common voltages (24V, 18V and 12V).

1. Remove the Teensy 3.1 from the SCM.
2. Set the variable DC supply to the correct voltage.
3. Connect the high to input pin 1.3 and 5 and the rest to ground.
4. Use the voltmeter to check the Teensy 3.1 slot pins and record.
5. Connect the Teensy 3.1 and check for the verification LED.

Results:

The following table (Table 13) shows the results of the experiment, displaying the output of the resistor circuit when a high was supplied to the input for each of the voltages tested. The last column shows whether the Teensy 3.1 received the verification message which confirmed the manual measurements and lit the on-board LED.

Table 16: Results of Step-Down Experiment

Logic Supply V	MCU Input	Read State
24	3.42	Pass
18	3.31	Pass
12	3.26	Pass
5	3.34	Pass

Discussion:

The step down of voltage from the machine controller's voltage level to that accepted by the SCM was done using the voltage divider rule of parallel resistors. For this method a different combination of resistors was needed for each connected machine voltage (see Table 13), so the resistor pairs for each of the supplied voltages had to be tested. It is clear that if a low (0V) is supplied to the resistor circuit input then a low will be returned and this was the case. More important was to test that the output of the circuit when a high is supplied is suitable to signal a high to the MCU without overloading its pins. A Teensy 3.1 has tolerance for up to 5V on its input pins but reads a high at anything over 3.1V (logic level 3.3V). The 'Method' section discussed how two tests were conducted – both sets of results are shown in Table 16. The table shows that the voltage supplied to the MCU was around 3.3V in each case and was above the high threshold at all supply voltages, although the 12V value was a little lower than expected. This lower reading was probably due to combined tolerances on the resistors. When the MCU was connected and the power supply voltage was connected to the correct pins the LED was lit in every case and a pass was recorded.

Conclusion:

In conclusion, the high output when the input was at the supplied high was very near the desired 3.3V and the low was at 0V. The test program confirmed the results by running correctly in each case. Thus, the circuit performed as expected and was proved suitable for the SCM.

10.2.2 Voltage Step-Up for Communication

The SCM was required to boost its voltage to any logic level supplied by the communication receiver (discussed in section 5.5). This required a voltage step up technique for each communication channel. A method using transistors, acting as relays, was chosen.

Aim:

This experiment aimed to test the effectiveness of the electrical channel step-up design in reality and determine if the results were suitable for the SCM.

Apparatus:

- ◆ Prototype SCM
- ◆ Variable DC supply

- ◆ PC with Arduino IDE and MS Excel
- ◆ Voltmeter

Method:

To test the effectiveness of the voltage step up, a manual test was used. The outputs of the transistors were checked with a voltmeter and recorded. The Teensy 3.1 was written with a program which set one output pin to ‘high’ and another to ‘low’. The following method was obeyed for each of 24V, 18V and 12V:

1. Program the verification routine to the Teensy 3.1.
2. Set the variable DC supply to the correct voltage.
3. Use the voltmeter to check the transistor outputs of both of the pins and record the results.

Results:

The following table outlines the experimental results – the output of the transistor circuit is shown for both of the MCU pin states and the last column shows whether or not the results were within the tolerance of a typical machine controller.

Table 17: Results of Step-Up Experiment

Supply V	OutputV		Winthin Parameters?
	MCU High	MCU Low	
24	0.081	24.04	Yes
18	0.048	18.08	Yes
12	0.056	11.97	Yes
5	0.051	5.21	Yes

Discussion:

The transistor technique used inverted the MCU output as expected (see section 5.5). This inversion property was dealt with in the module’s coding, so that the PLC would receive the expected message (as defined in Table 10). The output at the end of the transistor conversion circuit was, at the most, 0.21V off the supplied value, although with further investigation it was found that the variable power supply was actually slightly erroneous at 5V and so the conversion circuit was probably operating perfectly.

Conclusion:

In conclusion, the low output was very near 0V and well within the tolerance of any machine controller and the high was very near the supplied voltage and thus definitely within the range tolerated by the connected controller. The circuit worked successfully and proved to be suitable for use as the communication channel boost circuit.

10.2.3 Voltage Regulation for Power Supply

The SCM was required to operate when supplied by any voltage source from 12-24V (discussed in section 5.6). This required voltage regulation circuitry to be included on the SCM. This experiment tested the effectiveness of that circuit.

Aim:

To test the output of the voltage regulation circuitry and to confirm whether or not it was suitable for use in the SCM.

Apparatus:

- ◆ Prototype SCM
- ◆ Variable DC supply
- ◆ Voltmeter

Method:

This test was done in two stages for each of the voltage levels tested, first the output of the circuit was tested manually and then the Teensy 3.1 was connected and a test program was run to confirm the operation. The circuit was tested at each of the following, 26V, 24V, 18V, 12V & 10V to ensure that the circuit could handle variations in supply.

1. Remove the Teensy 3.1 from the SCM.
2. Set the DC supply to one of the voltages.
3. Used the voltmeter to test the output of the regulation circuit.
4. Insert the Teensy 3.1 and check that it boots and runs the test program.

Results:

The table below (Table 18) shows the output of the voltage regulation circuit for each of the different supply voltages, the regulated voltage was what was connected to the V_{in} pin of the Teensy 3.1. The last column shows whether the MCU booted and ran a test program (blinking the LED) when connected to the SCM and confirmed the effectiveness of the regulation.

Table 18: Results of Voltage Regulation Experiment

Supply V	Regulated V	Boot State
26	5.07	Pass
24	5.09	Pass
18	5.07	Pass
12	5.08	Pass
10	5.03	Pass

Discussion:

The regulation circuit discussed in section 5.6 was made using cascaded voltage regulators. The circuit was tested to beyond the foreseen cases for safety and thus goes either side of the typical 12V-24V range. In every case but 10V the regulated voltage was measured as being constant (within the voltmeter's error range). At the lowest tested voltage the regulated value began to drop slightly as the output of the first regulator (12V) dropped below the required input of the regulator two (so below 6.5V). This isn't an issue because the supply should not drop below 12V. The cases where supply is below that typical value are discussed in 5.6, and in that case the supply can be connected directly or through only the second regulator. In every case tested here the test program ran, showing that the supply was within the needs of the MCU.

Conclusion:

In conclusion, the manual readings showed that the output was very close to 5V and the boot test showed that it was usable by the MCU. Thus, both the manual readings and the pass/fail program test confirmed that the voltage regulation circuit was suitable for the SCM's power control.

10.3 Literature Logarithmic Drop-off in RSSI with Distance

Hypothesis:

The relationship between distance and command-mode RSSI is logarithmic.

Purpose:

In theory, the received signal strength indication decreases in a logarithmic fashion with increased distance. This experiment aims to verify the logarithmic drop off in RSSI with an increased distance between two XBee wireless modules which serves as a central concept in estimating radii as needed in the RTLS (see section 3.6.3).

Apparatus:

- ♦ 15m Tape measure
- ♦ Masking tape
- ♦ Laboratory with adequate space
- ♦ XBee Series 2 PCB Antenna module
- ♦ Uartsbee XBee to PC adapter
- ♦ State Communication Module as described in section 3.5
- ♦ Computer with serial communication software (Digi XCTU)

Method:

1. The tape measure was laid out along the lab floor from one wall.
2. Pieces of tape were placed at 0.5m intervals starting 2m from the wall until 13m.
3. The Uartsbee was used to connect the XBee to the computer.
4. The connected XBee was placed against the wall, 1.2m up from the ground.
5. The serial software was opened and the home XBee was connected to the SCM XBee.
6. The SCM was placed on the floor at the first mark and switched it on.
7. “+++” was typed into the serial software to enter command mode.
8. “ATDB” was typed to request the RSSI.
9. The response was recorded (given in hexadecimal).
10. The software was allowed to wait 10s to exit command mode.
11. Steps 7-10 were repeated 5 times.
12. Steps 6-11 were repeated for each of the marks.
13. The entire experiment was re-run four times in two days for accuracy.

Results:

Table 19 is a summary of the comprehensive results contained in Appendix K (15.11); it shows the average RSSI values obtained in each of the experiments as well as an overall average of all the readings taken. Figure 60 is a graph showing the relationship in each experiment and best-fit logarithmic trendline (as generated by MS Excel 2013). Figure 61 shows the relationship obtained when the average RSSI values of all the experiments were used and thus the average best-fit logarithmic trendline.

Table 19: Summary of the Experimental Results

Relationship between Distance and RSSI						
Distance	Exp 1: Ave RSSI	Exp 2: Ave RSSI	Exp 3: Ave RSSI	Exp 4: Ave RSSI	σ (dBm)	Ave RSSI
1	44.4	46.0	43.0	45.4	1.136	44.70
1.5	46.8	49.8	46.4	47.0	1.345	47.50
2	49.2	48.8	48.8	49.8	0.409	49.15
2.5	54.6	51.2	50.8	49.4	1.910	51.50
3	55.2	54.0	52.4	51.6	1.396	53.30
3.5	56.1	57.2	52.8	58.4	2.085	56.13
4	57.2	59.0	57.8	58.6	0.698	58.15
4.5	57.6	60.2	58.0	54.0	2.224	57.45
5	61.2	61.8	59.0	58.6	1.374	60.15
5.5	56.4	57.8	58.0	56.8	0.669	57.25
6	57.4	55.2	55.8	55.8	0.817	56.05
6.5	59.6	61.6	56.6	59.0	1.783	59.20
7	55.6	62.2	56.8	54.2	3.030	57.20
7.5	57.8	63.6	57.8	61.2	2.452	60.10
8	58.6	59.6	55.4	58.8	1.603	58.10
8.5	62.0	64.0	58.8	57.2	2.659	60.50
9	58.6	60.4	60.0	58.6	0.812	59.40
9.5	60.6	61.2	60.4	59.0	0.806	60.30
10	62.8	61.4	61.4	60.8	0.735	61.60
10.5	64.4	63.4	62.8	63.4	0.574	63.50
11	65.2	63.8	63.6	62.4	0.994	63.75
11.5	65.4	65	64.6	64.6	0.332	64.90
12	66	65.4	65.8	64.2	0.698	65.35

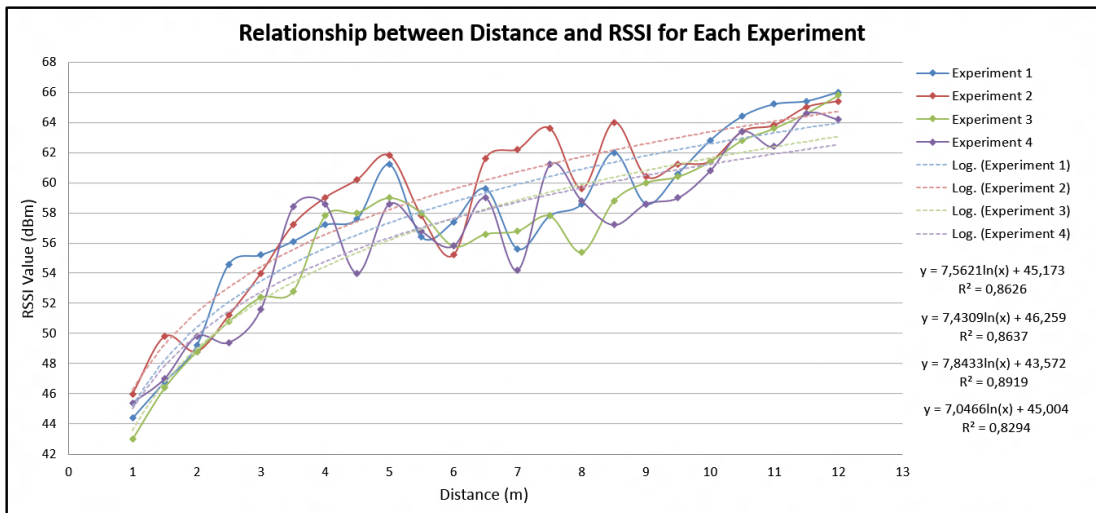


Figure 60: Distance vs. RSSI for Each Experiment

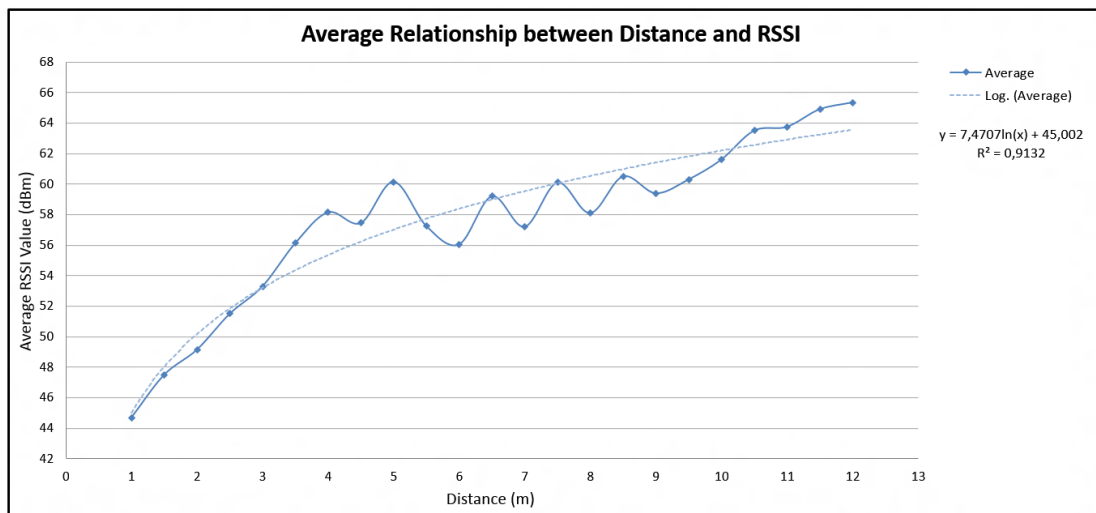


Figure 61: Distance vs. RSSI for the Average Reading

Discussion:

Firstly, some explanation is needed to make clear the decisions made in the method. Firstly, the measurements were started at 2m from the wall (and given the value of 1m) because in reality the beacon is designed to be placed at ceiling level (>3.5m) and the SCM is placed on top of a machine (usually <2m) which means the height differential is usually over the 1.2m used in the experiment. Thus the '1m' mark was placed further from the wall. In reality the factory would need a 1m reference power value to put the other readings into context (this corresponds to the p_0 in [Eq. 7]).

The results do generally show a logarithmic relationship between the distance and RSSI, but the estimations are inconsistent and noisy. The average case (shown in Figure 61) gives the following constants calculations shown in Appendix L (section 15.12):

$$\begin{aligned} p_0 &= 45.15 \text{ dBm} \\ r_0 &= 1 \text{ m} \\ r_j &= x \\ n &= 1.71 \end{aligned}$$

The goodness of fit (R^2) value given by Excel is 0.9041 which shows an acceptable fit with the theoretical prediction using the values shown above. What isn't shown by this good result is the wavelike fluctuation in values, especially between 4m and 10m. This is especially problematic due to this being a very important range for the RTLS. Problems are also found when the standard deviation of the results is considered. Appendix K (section 15.11) shows that the standard deviation within the five readings of each experiment was as high as 1.789 dBm and averaged 0.982 dBm. Furthermore, the standard deviation between the four experiments was as high as 3.03 dBm and averaged 1.273 dBm.

The wavelike fluctuations present a particular problem. When using this exact information in reverse (to predict distance based on RSSI) the results cause many of the RSSI readings to yield ambiguous results (a single RSSI value could be two distances). It also means using the trendline will likely give very unreliable results.

It is thought that part of the problem in this case is the lack of resolution in the RSSI output of the XBee using the ATDB command. The average RSSI difference over the whole range tested was 21 dBm and this is given in steps of 1dBm. This leads to a very low resolution over the tested range, leading to over- and under-reactions to distance changes and thus the wavelike patterns found in the results. The standard deviations discussed in the preceding paragraphs show why this low resolution is a particularly important problem – the standard deviation is exaggerated by the large jumps in readings given by the low resolution.

Conclusion:

The results match the theory well on average, proving that the signal strength drops logarithmically with distance. Due to large deviations in readings which should be similar and wavelike nature of some of the readings, the ATDB command used to gather the RSSI from the XBee module appeared to be an unreliable method for predicting distance in the RTLS. Another, higher resolution, method for gathering the RSSI of a XBee module and a microcontroller is documented in the literature [104] and is tested in the next section (10.4).

10.4 Linear Increase in PWM Pulse Length with Distance

Hypothesis:

The new RSSI function will produce a linear relationship between distance and *pulseIn* indication.

Apparatus:

- ◆ 15m Tape measure
- ◆ Laboratory with adequate space
- ◆ XBee Series 2 PCB Antenna connected to a PC with serial communication software
- ◆ State Communication Module as described in section 5

Method:

1. The tape measure was laid out along the lab floor from one wall.
2. Pieces of tape were placed at 0.5m intervals starting 2m from the wall until 13m.
3. The Uartsbee was used to connect the XBee to the computer.
4. The connected XBee was placed against the wall, 1.2m up from the ground.
5. The serial software was opened and the PC XBee was connected to the XBee on the SCM.
6. The SCM was placed on the floor at the first mark and switched it on.
7. “a” was typed into the serial terminal to trigger the calculation and the response recorded.
8. A delay of 5s was given to ensure the MCU was clear of the previous routine.
9. Steps 7&8 were repeated five times.
10. Steps 7-9 were repeated for each of the marks
11. The entire experiment was re-run three times in three days to ensure a good average

Results:

Table 20 is a summary of the comprehensive results contained in Appendix M (15.13); it shows the average *pulseIn* duration values obtained in the experiments and average value for each 0.5m mark. Figure 62 shows the average *pulseIn* readings for each experiment and Figure 63 shows the average over the three experiments. Both of the graphs (Figure 62 and Figure 63) contain a best fit linear trendline with a goodness of fit (R^2) indication (as generated by MS Excel 2013).

Table 20: Summary of the Experimental Results

Relationship between Distance and PulseIn Value					
Distance (m)	Exp 1: PulseIn (ms)	Exp 2: PulseIn (ms)	Exp 3: PulseIn (ms)	σ (ms)	Ave PulseIn (ms)
1	2,4	3,2	2,8	0,3	2,80
1,5	4,7	5,9	4,9	0,5	5,17
2	9,6	8,4	9,7	0,6	9,23
2,5	21,5	12,6	19,4	3,8	17,83
3	24,4	17,2	22,2	3,0	21,27
3,5	30,4	24,9	29,8	2,5	28,37
4	34,4	29,3	31,1	2,1	31,60
4,5	39,4	32,7	36,7	2,8	36,27
5	41,8	38,2	40,1	1,5	40,03
5,5	50,7	46,4	47,6	1,8	48,23
6	51,1	49,0	50,1	0,9	50,07
6,5	54,8	51,2	55,0	1,7	53,67
7	60,5	55,7	57,9	2,0	58,03
7,5	63,4	57,9	62,8	2,5	61,37
8	67,5	59,6	66,1	3,4	64,40
8,5	72,7	64,2	69,8	3,5	68,91
9	82,3	67,1	78,6	6,5	76,00
9,5	83,1	72,9	83,3	4,9	79,77
10	90,8	75,1	88,9	7,0	84,93
10,5	94,6	82,0	93,0	5,6	89,87
11	98,7	83,2	98,1	7,2	93,33
11,5	102,9	89,1	101,2	6,1	97,73
12	108,7	90,9	103,7	7,5	101,10

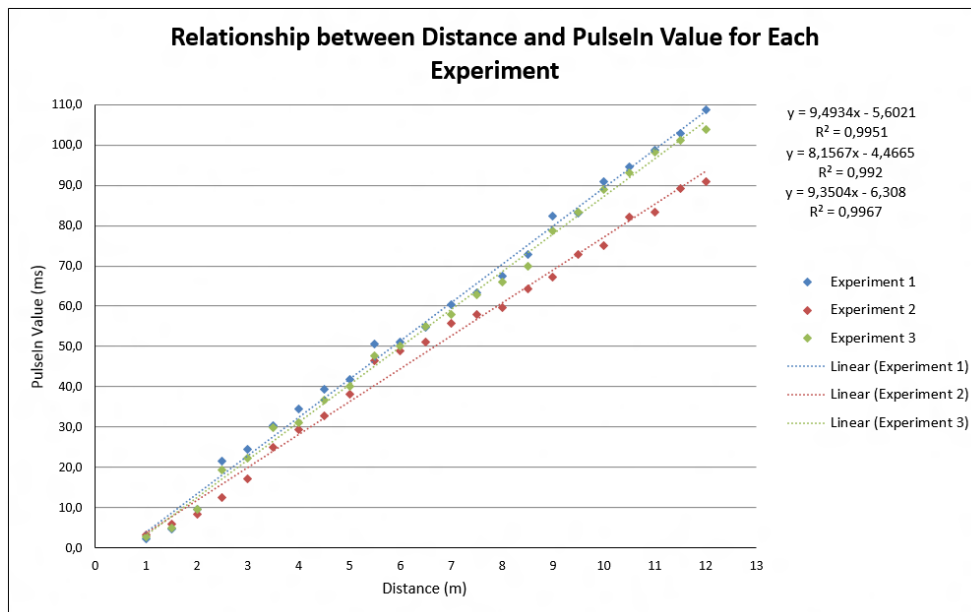


Figure 62: Distance vs. PulseIn Value for Each Experiment

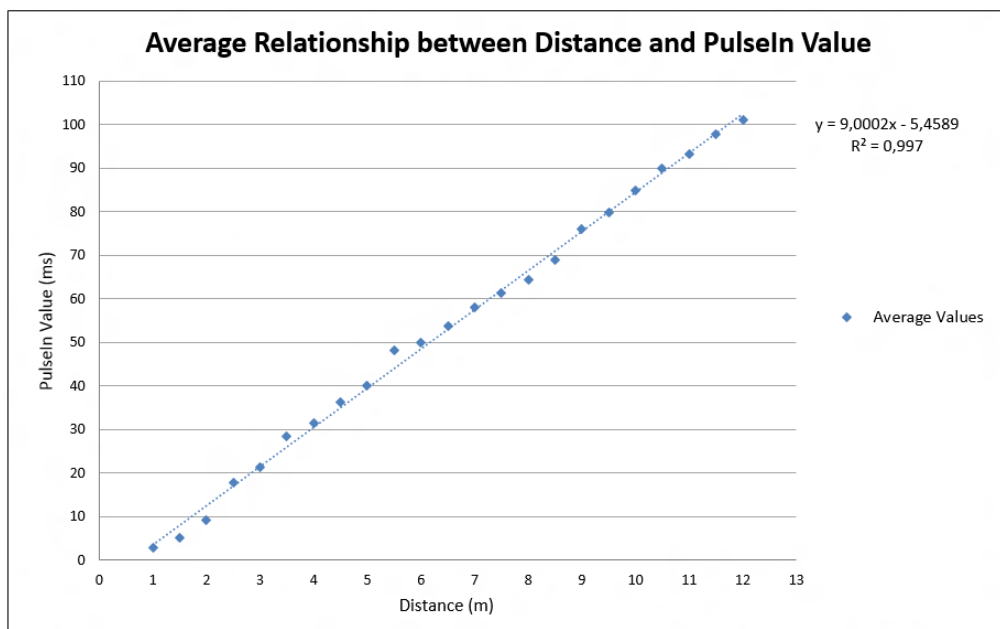


Figure 63: Distance vs. PulseIn Value for the Average Reading

Discussion:

This experiment shows that the new method for gathering the RSSI, using the *pulseIn* function, was much more reliable for use as a distance indicator. The use of the function does not follow the logarithmic pattern predicted in the literature. This was explained by the *pulseIn* function itself and the way that the PWM is produced by the XBee. Further research shows that the XBee produced an exponentially varying PWM signal which linearised the signal strength indication. For this reason the relationship is linear, as shown in Figure 62 and Figure 63.

Table 20 shows that there is some variation ($\sigma = 7$) between the experiments, but that most of the variation came from Experiment 2. The results for Experiment 2 were consistently lower at each interval onwards from the 5m mark, with a diverging trendline. This experiment must have been affected systematically in some way, perhaps due to environmental effects.

In the average and in Experiments 1 and 3 there is a significant jump in the *pulseIn* readings obtained between 2m and 2.5m. This jump does not affect the linear relationship noticeably and is not present in Experiment 2.

The most promising attribute of the results is how well the readings fit into a linear relationship. In all of the experiments (and the average) the Excel generated R^2 goodness of fit value is over 0.99. This is an almost perfect fit to the linear trendline and shows that using this technique should give very accurate mapping of *pulseIn* values to distances between XBee modules

Conclusion:

The results are much more promising for the development of a RSSI-based RTLS when using the *pulseIn* function, the fitment to a trendline is almost perfect and accurate enough to make confident predictions. There is a clear improvement in using the much higher resolution *pulseIn* method as opposed to the ATDB command method in the previous experiment.

10.5 Testing the RSSI Distance Estimation

Aim:

To test the accuracy of the distance estimation between two modules using the equation determined in the previous experiment (section 10.4). This experiment served to test the distance estimation equation for the RTLS.

Apparatus:

- ◆ 30m Tape measure
- ◆ Masking tape
- ◆ Laboratory with adequate space
- ◆ XBee Series 2 PCB Antenna module
- ◆ Uartsbee XBee to PC adapter
- ◆ State Communication Module as described in section 5
- ◆ Computer with serial communication software (Digi XCTU)

Method:

This experiment was conducted in the laboratory in which the system was designed to be implemented to ensure realistic results.

1. The tape measure was laid out along the lab floor from one wall.
2. Pieces of tape were placed at 1m intervals starting 2m from the wall until 22mm.
3. The Uartsbee was used to connect the XBee to the computer.
4. The connected XBee was placed against the wall, 1.2m up from the ground.
5. The SCM was placed on the floor at the first mark and switched it on.
6. "a" was typed into the serial terminal to trigger the calculation.
7. A delay of 3s was given to ensure the XBee buffer was clear.
8. Steps 7&8 were repeated five times to average the distance estimation.
9. "q" was typed to trigger the distance calculation (according to [Eq. 13])
10. The estimation response was recorded.
11. Steps 7-11 were repeated for each of the marks.
12. Steps 7-12 were repeated three times per experiment.
13. The entire experiment was re-run three times in two days to ensure a good average.

Results:

Table 21 shows the results of the experiment. The experiment was performed three times, with each experiment comprising of a distance estimation at each point, repeated three times. Each of the three experiments were conducted from different points to test the consistency over the whole factory floor rather than in one case. This gave nine readings for each of the tested distances. The table shows each of the distance estimations returned and average estimation for each point. The standard deviation of the estimations is included, as is the error. The 'Pass %' column shows the percentage of estimations which fell within the acceptable range (-250mm; +1250mm), the failures are highlighted in pale red. Averages of the errors, standard deviation and pass percentage are provided. The 'Max Diff' column shows the maximum difference in estimations across the nine readings for that distance.

Table 21: Results of the Distance Estimation Test

True Dist	Exp 1			Exp 2			Exp 3			Average	σ	Error	Pass %	Max Diff
	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3	Run 1	Run 2	Run 3					
2000	2847	1698	2161	3109	2721	2084	2198	2488	2884	2465.56	434.54	465.56	100	1411
4000	4681	4583	4151	3833	3986	4821	4451	4711	3761	4330.89	381.41	330.89	100	1060
6000	5774	6934	6384	7064	5932	5841	5849	5898	6112	6198.67	462.29	198.67	100	1290
8000	7970	8316	8435	8849	7900	8066	7955	8910	8227	8292.00	355.81	292.00	100	1010
10000	9870	11054	10712	10824	9918	10649	10235	10954	9841	10450.78	459.89	450.78	100	1213
12000	13124	13202	12354	12360	12826	12241	12877	12651	11906	12615.67	407.41	615.67	100	1296
14000	13756	14215	14845	13824	13721	13813	13911	14097	13987	14018.78	329.63	18.78	89	1124
16000	15798	16664	16741	15795	15862	16097	15689	16412	15796	16094.89	384.17	94.89	89	1052
18000	17811	18812	17713	18911	18230	18711	18498	18256	17781	18302.56	435.05	302.56	89	1198
20000	21068	19757	21121	19860	21040	20368	19876	21033	20527	20516.67	542.69	516.67	100	1364
										Average	419.29	328.64	96.7	

Discussion:

The pass/fail criteria of (-250mm; +1250mm) was chosen because of the nature of the RTLS's multilateration algorithm. Thinking logically, the multilateration algorithm chosen requires the distance estimation circles to intersect, for this to happen, the estimations cannot be too small. If the lines did not cross it would give erroneous results, but if the SCMs all overestimate by a fixed amount, there would be no effect on the position estimation. For this reason, a constant 500mm was added to all estimations and the pass criteria was offset to the over-estimation side.

The experiment shows that the vast majority (>96%) of the distance estimations fell within the defined error band and were therefore passes. The runs which failed were also not by large margins. It was clear that the RSSI based distance estimations were more inconsistent than what would be desired, though this was expected and was catered for with the later processing. The large standard deviations in the readings and large 'Max Diff' values showed that the distance estimation may have been too inconsistent, although the theoretical multilateration and DBSCAN experiments which yielded the pass/fail criteria show that this inaccuracy could be handled.

Conclusion:

The distance estimations were not as accurate as the earlier experiment on the RSSI relationship with distance (section 10.4) would have suggested, nor as accurate as would usually be desired for a reliable experiment. Even so it proved to be sufficiently accurate for use with the multilateration and DBSCAN algorithms (as was found in sections 10.6 and 10.8). The high pass

rate and good averages of the system showed that the RSSI-based distance estimation technique was viable for use in the RMMS's RTLS.

10.6 A Theoretical Test of the Multilateration Algorithm

Aim:

To test the multilateration algorithm based on theoretical distance estimations, including random inaccuracies.

Apparatus:

- ◆ A computer running MathWorks Matlab and Microsoft Excel
- ◆ A multilateration algorithm in Matlab

Method:

The Matlab program was run once for each of the five positions (given below in mm) on the hypothetical 10m × 10m factory floor:

(2000; 2000), (2000; 8000), (9000; 2000), (8000; 8000) & (5000; 5000).

These positions were chosen to represent many different positions on the floor and to allow for inaccurate spots to be identified. A Matlab program (shown in Appendix G, section 15.7) was written which was set to run 10 000 iterations. The program follows the pseudo-code shown in Figure 35, but uses random values to simulate the gathering of RSSI data. In order for a run of the algorithm to be considered a pass, it had to predict both of the point's co-ordinates to within 600mm above or below (1200m total). The randomised values for RSSI estimation were based on data from the experiment on distance estimation in section 10.5. This randomisation was used to provide a simulation of the probable real distance estimation data. The distance estimation data was input with the randomisation (-250mm; +1250mm) because this was found to give favourable results and was found to be a realistic expectation in the experiment on distance estimation (10.5).

1. Each iteration was set to produce 10 random numbers for each of the distance estimation (between 250mm under the known correct distance and 1250mm over). This simulates the gathering of RSSI information.
2. An average was taken of the 10 values.
3. The algorithm then follows the pseudo-code in Figure 34 from step 7.
4. The program checks if it passes and records the result.
5. After 10 000 iterations, the program worked out the standard deviation of the predicted positions.
6. The program printed the number of passes and standard deviation.
7. The results were recorded in Excel.

Results:

Table 22 shows the results of the experiment, showing the standard deviation of the position estimation results and percentage of tests which passed. The pass accuracy was based on the results of the DBSCAN experiment (discussed in section 6.3).

Table 22: Results of the Multilateration Experiment

Module Co-Ord (mm,mm)	Result Average Co-Ord (mm,mm)	Sample Size	Pass Accuracy (mm)	σ (mm)	Pass %
(2000;2000)	(2032;1944)	10000	+600	329	92
(2000;8000)	(2072;8029)			312	94
(8000;8000)	(7843,8134)			309	91
(8000;2000)	(7911;2105)			331	92
(5000;5000)	(5109;4889)			285	91

Discussion:

The table shows that, even when given highly varying distance ‘estimation’ data, over 90% of the 10 000 cases tested for each of five hypothetical positions were accurate enough to be considered a pass. A pass was decided to be a point which was within 600mm of the true position in both the X and Y dimensions, giving a square region of success with a maximum Euclidian error in each corner of the region of 848mm away from the point.

The maximum standard deviation of all the results was 331mm. The average standard deviation was 313mm over the five positions. This implies that over 66% of the 50 000 cases tested were within 313mm of the true position. This information gave valuable data for the continuation of research and as a proof of concept for the multilateration algorithm. This experiment was conducted in conjunction with the DBSCAN experiment (section 10.8) in order to define the pass requirements and ensure that the algorithms were suitable for use with each other.

Conclusion:

The multilateration algorithm produced accurate results using the simulated distance estimation. Further experimentation was needed verify the real-world performance of the RTLS, but the validity of multilateration algorithm was verified here.

10.7 Testing the RTLS in the with one Roaming NodeAim:

To test the ability of the RSSI based multilateration RTLS to estimate the position of a roaming node on a factory floor. The accuracy of the RTLS had to be checked to ensure it met the criteria for use in the RMMS.

Apparatus:

- ◆ A 10.5m×13.5m laboratory floor
- ◆ 4 BPMs
- ◆ 1 SCM
- ◆ 1 XBee and UARSBEE connection board
- ◆ 15m measuring tape
- ◆ A PC with XCTU software and MS Excel

Method:

1. Place a BPM in each of the corners of the floor space.
2. Place the SCM at the first position.
3. Type “x” into XCTU’s terminal to trigger the RTLS.
4. Wait for the SCM to send its position estimation.

5. Record the result.
6. Move the SCM to the next position and repeat steps 3-5.
7. Repeat steps 2-6 three times in three days to ensure accuracy.

The six co-ordinates shown in Figure 64 were tested, the positions were chosen because the positions were those of currently placed machines on the factory floor, making for convenient and realistic placement. The use of a proper laboratory-scale factory floor, with the usual machines as obstacles was decided upon to give the experiment a more realistic and applicable result. The photograph below (Figure 65) shows the factory floor used and the diagram (Figure 64) shows the positions used in the test.

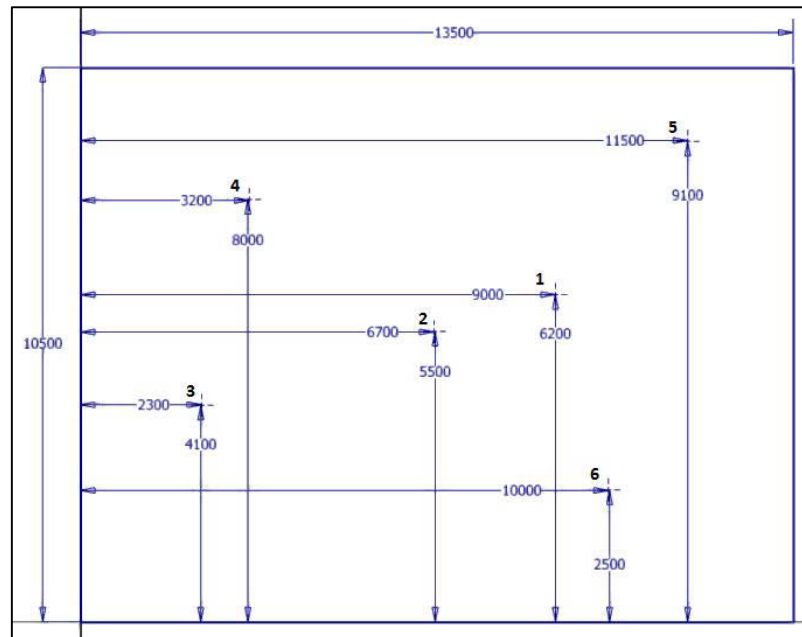


Figure 64: Positions Used in the Experiment



Figure 65: Photograph of the Factory Floor Test Space

Results:

The following diagram (Figure 66) shows the true position, each estimation and the average of the estimations in layout view of the factory floor. The table which follows gives more detailed information of the results (Table 23) shows the results that were gathered for each position in each of the three experiments. The average of two (X and Y) estimations are shown shaded in grey and for each of the three experiments the X and Y estimations are given. The Average Euclidian distance error over the three experiments is shown as the 'Ave Error'. The 'Max Error' column shows the maximum Euclidian error over the three experiments. The average of the X and Y standard deviation (σ) is also given.

Table 23: Tabularised Results of the Roaming Node Experiment

Positions	True X	True Y	Experiment 1		Experiment 2		Experiment 3		σ	Ave Error	Max Error	Ave X	Ave Y
			Est X	Est Y	Est X	Est Y	Est X	Est Y					
Position 1	9000	6200	9385	5691	8791	6619	9486	5893	352.54	560.43	638.21	9220.67	6067.67
Position 2	6700	5500	6389	5714	6877	6012	7219	5918	232.49	528.55	666.40	6828.33	5881.33
Position 3	2300	4100	1899	3764	1994	4198	2397	4261	218.41	344.14	523.16	2096.67	4074.33
Position 4	3200	8000	3114	7911	2841	8319	3415	8471	235.44	373.92	517.75	3123.33	8233.67
Position 5	11500	9100	10912	8919	12128	8879	12013	9042	308.75	599.08	665.75	11684.33	8946.67
Position 6	10000	2500	10644	2339	10614	2618	9864	2514	237.98	475.26	663.82	10374.00	2490.33

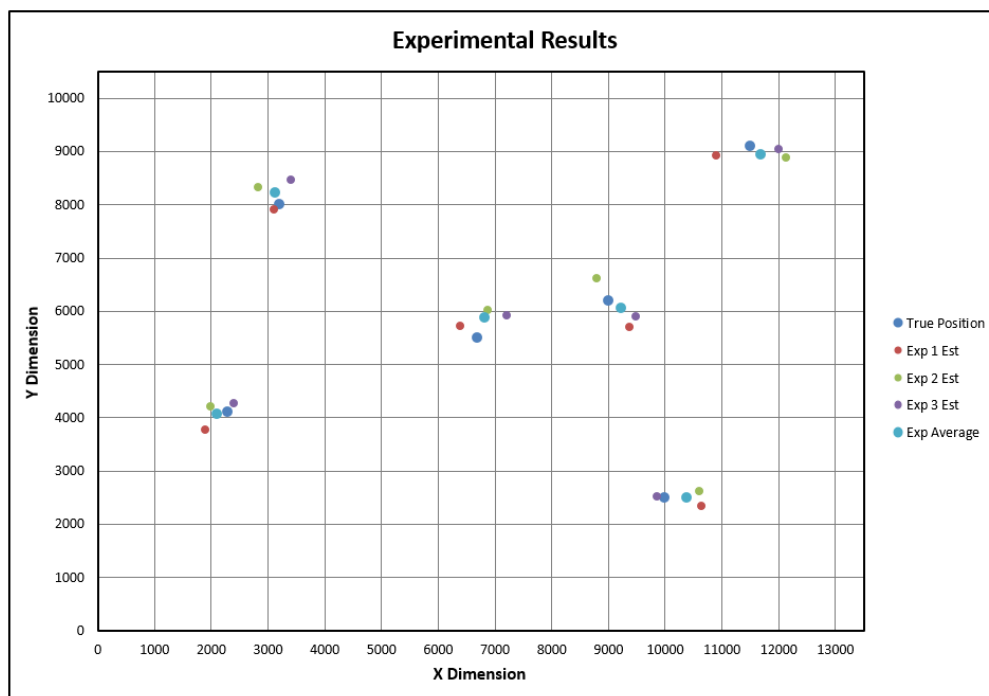


Figure 66: Results Shown in their Position on the Factory Floor

Discussion:

The results show that every one of the estimations was within 670mm of the actual position. The DBSCAN experiment (10.8) showed that even an error radius of 800mm was tolerable and so every one of the location estimates would have satisfied the DBSCAN algorithm and should be able to be processed into a cluster model.

The average error of each of the positions ranged between 344.14mm and 599.08mm, and were near the same as the maximum errors, so it was clear that most readings fell away from the true positions, but within the acceptable error band. This showed that it was likely that the estimation

pattern was a very steep bell curve, bordering on being a constrained random distribution. The sample size was too small to confirm this.

The different positions yielded different accuracies, but there was no clear pattern as to what location factors affected accuracies. The small sample size implied that this was probably just pseudo-random variation and that no particular tested position was more prone to inaccuracy than any other.

The sample number was not large, but even so, the results imply that all estimations should fall within the tolerable error band, which takes the form of a circle around the true position, of radius 800mm.

Conclusion:

This experiment was not of a large sample, but it illustrates the feasibility of the RTLS and the results were positive and showed that the RSSI-based multilateration RTLS was suitable for position estimation in the RMMS. The RTLS developed in this research was able to find the position of a roaming node, within the accuracy needed by the processing system.

10.8 Constants and Testing of the DBSCAN Algorithm

Aim:

To find EPS and MinPts values which can correctly (to a success rate of over 80%) cluster a number of simple factory floor layouts into cells, with significant variations in the location data.

Apparatus:

- ◆ A computer running MS Visual Studio and MS Excel
- ◆ The DBSCAN clustering algorithm, with variable EPS and *MinPts* properties

Method Description:

The four hypothetical factory floor layouts in Figure 67 were tested in this experiment. They are all examples of typical layouts on a 25m×18m factory floor. Table 24, which precedes Figure 67, contains the true positions of each machine in each floor.

Figure 67 shows that the MinPts should be set to 2 to cater for the two-machine cell in layout 4. Once this decision was made, only the EPS needed to be varied to find the best suited values.

Table 24: Actual Position of Machines

Machine	Configuration Number												
	1			2			3			4			
	Cell	X (m)	Y (m)	Cell	X (m)	Y (m)	Cell	X (m)	Y (m)	Cell	X (m)	Y (m)	
1	1	5	16,5	1	8	16	1	1,5	16	1	2	16,5	
2		8	16,5		10,5	16		4	16,5		4,5	17	
3		10,5	16,5		13	16		7	16,5		2	14,5	
4		13	16,5		15,5	16		1,5	13,5		2	12,5	
5		15,5	16,5		18	16		4,5	14		2	10	
6		17	14		20,5	16		7,5	14,5		2	8	
7		14	12,5		23	16		3	11,5		2	5,5	
8		11,5	12,5		2	11,5		17	16		4,5	4,5	
9	2	5	7	2	4,5	11	2	20	15	2	12	12	
10		2	7		7,5	9,5		17	14		12	15	
11		2	5		2	8,5		20	13		14,5	16	
12		2	2,5		4,5	9,5		17	12		17	15	
13		5	1,5		10	4		17	9,5		17	12	
14		7,5	1,5		12,5	3,5		17	7,5		14,5	13,5	
15	3	20,5	7	3	15	2	3	17	5,5	3	18	6	
16		22	5		12	1		20	5,5		20	6,5	
17		20,5	3		10	2		23	5,5		22	6	
18	4				4	21	10,5	3	3,5	5	4	22,5	4
19						18,5	8,5		5	4		21	2
20						21	8,5		8	5		20	4
21						21	6,5		3,5	2,5		10,5	1,5
22									5	1,5		13	2
23							8	2,5					

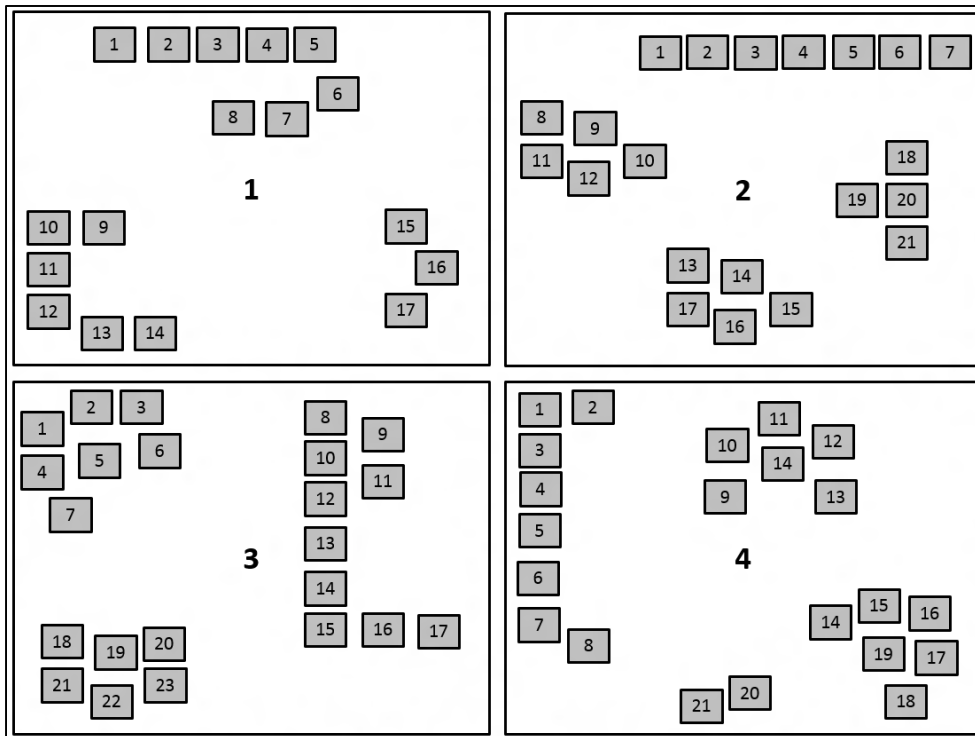


Figure 67: Hypothetical Test Layouts

Data known of the location accuracy, obtained from previous experiments (see section 10.4 and 10.6), containing the standard deviation, was used to give variations in the simulated layout. These inaccuracies are needed to ensure that the DBSCAN algorithm can handle the variations in position that may occur with inaccurate positioning information. This experiment was conducted before the final data from the roaming node experiment (10.7) was known and that the data found there fell within the defined constraints.

The tests were run with the following data values, inferred from section 10.4 (with exaggeration to pose more of a challenge to the clustering algorithm). In the formulae below, μ is the true location value and x is the value in each case and r is a random number between 0 and 1. The variations were applied to both co-ordinates, giving a possible error in position equal to square root of the sum of the squares of the co-ordinate errors. Both sets of values were generated by MS Excel.

- ◆ Random distribution: $x = \mu \pm r800$
- ◆ Gaussian distribution: $N(\mu, 500^2)$

The graphs (Figure 68) show how the values were distributed around the mean, $\mu = 10\,000$.

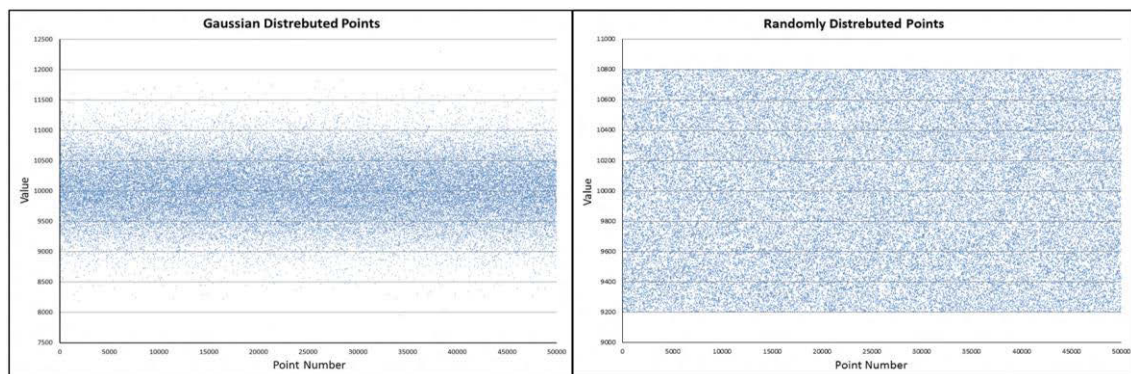


Figure 68: An Example Gaussian and Random Distribution

The figures above show how a single dimension (x or y axis coordinate) is varied according to the two randomising techniques. Figure 69 shows 5000 cases of Gaussian distribution as applied to Layout 3. This diagram serves to show how much the positions were randomised to test the DBSCAN algorithm. Randomly scattered data forms a square centred on each true position.

Figure 70, which follows Figure 71 shows the true position along with four randomly produced estimations of each factory floor. Each layout estimation is given in one colour so that it is possible to observe how varied an individual factory floor can become when constrained randomness or random Gaussian distribution is applied to many points at once. This served to demonstrate the robustness of the DBSCAN algorithm when dealing with inaccurate RTLS data.

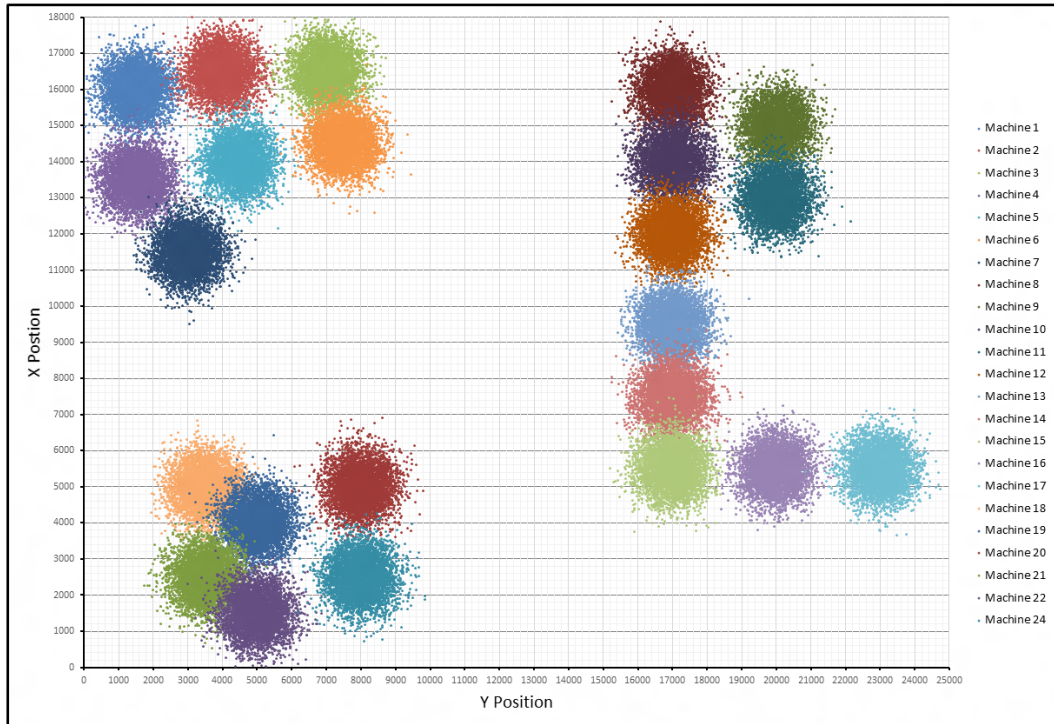


Figure 69: Gaussian Distribution of Machines in Layout 3

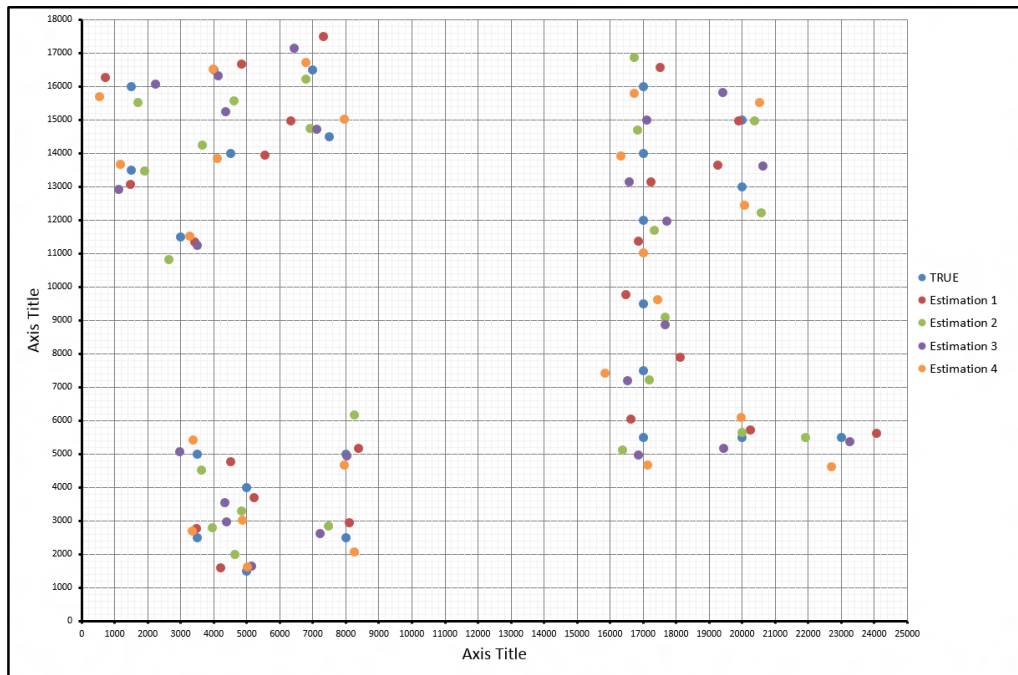


Figure 70: Random Distribution Cases of Layout 3

Part 1 Method:

1. EPS value was set to 3500.
2. Excel was used to generate 50000 randomly distributed points (within the accuracy of the RTLS).
3. Excel was used to generate 50000 points Gaussian distributed points (with a standard deviation equal to the RTLS).
4. The clustering algorithm was run for each of the Gaussian distributed data points.

5. Data on how many cases passed and how many failed was gathered and recorded.
6. Steps 2-4 were repeated for each factory floor layout.
7. The clustering algorithm was run for each of the randomly distributed data points.
8. Data on how many cases passed and how many failed was gathered and recorded.
9. Steps 2-4 were repeated for each factory floor layout.
10. EPS was incremented by 50 and steps 3-6 were repeated.
11. Until EPS reached 6000.

Part 1 Results Method:

Table 25: Results of the Variable EPS Experiment (Part 1)

Layout	Distribution	Best EPS	Passes	Pass %
1	Random	5200	50000	100,00
	Gaussian	5600	49969	99,94
2	Random	4300	50000	100,00
	Gaussian	4300	49570	99,14
3	Random	4900	49972	99,94
	Gaussian	4900	48967	97,93
4	Random	4400	48444	96,89
	Gaussian	4500	46772	93,54
Best Average EPS		4762,5		

Part 2:

1. EPS was set to the average of the bests (see Table 25) – Rounded to 4750.
2. Excel was used to generate 50000 randomly distributed points (within the accuracy of the RTLS).
3. Excel was used to generate 50000 points Gaussian distributed points (with a standard deviation equal to the RTLS).
4. The clustering algorithm was run for each of the Gaussian distributed data points.
5. Data on how many cases passed and how many failed was gathered and recorded.
6. Steps 2-4 were repeated for each factory floor layout.
7. The clustering algorithm was run for each of the randomly distributed data points.
8. Data on how many cases passed and how many failed was gathered and recorded.
9. Steps 2-4 were repeated for each factory floor layout.
10. Data (number of passes and failures, including failure reasons) was recorded.

Part 2 Results:

Table 26: Results of the DBSCAN Test (Part 2)

Test Situation					Results				
Layout	Distribution	EPS	MinPts	Tests	Pass	Fail Reason			Pass %
						Incorrect Machine Distribution	Incorrect Number of Clusters	Outliers Found	
1	Random	4750	2	50000	49835	0	164	0	99.67
	Gaussian	4750	2	50000	48615	0	737	648	97.23
2	Random	4750	2	50000	47098	0	2902	0	94.20
	Gaussian	4750	2	50000	46584	0	3414	2	93.17
3	Random	4750	2	50000	49878	0	1	121	99.76
	Gaussian	4750	2	50000	48639	3	398	960	97.28
4	Random	4750	2	50000	44977	0	5022	1	89.95
	Gaussian	4750	2	50000	44577	64	5140	219	89.15

Discussion:

Firstly, it can be assumed in most factory floors the minimum number of machines in a cell is 2 (with singular machines (singleton cells) being identified by being outliers, though if it is known that a cell will not exist with less than n machines, then the MinPts variable can be set to n . The MinPts should be set as high as possible – to the minimum number of machines which a factory has in any known state.

It was clear that the EPS has a large effect on the clustering algorithm's results. Even small variations can cause inaccuracies, as can be seen in the raw data in Appendix N (15.14). When it was properly tuned the DBSCAN algorithm worked very successfully, achieving a worst success rate of almost 90% (89.15%). This was beyond the requirement that was aimed for at the outset (80%). The constraints used for the randomised and Gaussian distribution were larger than the values found in the roaming node experiment (10.7), so the true accuracy would prove to be higher than the pass percentage found.

The variations used to test the algorithm were also very high and in a more conservative factory setting it is hoped that with some further work, the RTLS can achieve values with less deviation. Even when given inaccurate location data the clustering worked effectively.

The performance is slightly worse for each factory floor when the algorithm was fed data distributed in a Gaussian fashion. This can be attributed to the more extreme variances possible with this form of distribution and the high number of cases given (50 000). More of the values are concentrated near the centre than random distribution, but possible variations are much larger than that of constrained random distribution. The Gaussian distribution is likely to be closer to the data obtained in the real world. Table 26 shows that Gaussian distribution had a particular effect on the detection of outliers. This is because when given large amounts of data, a number of points are bound to vary enough to become un-clustered.

The table also shows that different factory floor layouts are prone to different sorts of errors and so failed for different reasons.

What is not shown in the results above, is that on multiple runs of the same set of data the algorithm produced identical results, which proved its deterministic nature.

Conclusion:

It was concluded that the DBSCAN clustering algorithm was well suited to the task of clustering a factory floor layout into cells based machine position. The high accuracy, robustness against RTLS estimation variations, easy customisability and deterministic nature of the algorithm made it a good choice for the clustering of machines into cells by the central RMMS controller.

10.9 Testing the Operation of the KBS

Aim:

To test the code and operation of the knowledge base system using hypothetical factory floors and a hypothetical base of knowledge.

Apparatus:

- ◆ A Windows PC with MS Visual C# running the KBS section of the CRC Code
- ◆ A set of hypothetical factory floors and a base of knowledge
- ◆ MS Excel

Method:

The base of knowledge (shown in Appendix O, section 15.15) was read by the CRC software and stored as the known states of the system. The CRC required a CSV file of a certain structure, which was created in Excel and read by the software. The database contains ten different factory floor layouts including ones which contain duplicate and triplicate cells. The following method was followed with each of the ten known layouts and two layouts which not contained in the knowledge base, in random order:

1. Test layout is given to the KBS to find a match.
2. KBS runs and returns a result.
3. Result is recorded in Excel.

Proving the functionality of the KBS in an experiment does not seem very interesting, because of the pass/fail nature of the test, but it was still very important for verifying the system's aspects.

Results:

The following table (Table 27) shows the results for each of the twelve runs (ten known layouts and two unknown – shown as x and y).

Table 27: Results of the KBS Experiment

Layout No.	Match Found?	Matched Layout	Result
2	Yes	2	Pass
4	Yes	4	Pass
7	Yes	7	Pass
x	No	null	Pass
10	Yes	10	Pass
1	Yes	1	Pass
y	No	null	Pass
5	Yes	5	Pass
3	Yes	3	Pass
6	Yes	6	Pass
9	Yes	9	Pass
8	Yes	8	Pass

Discussion:

The KBS showed perfect results in the simulated test, matching each of the hypothetical floors exactly as expected and identifying that some of the floors did not exist in the database. Even though the results were perfect in the end, the experiment proved very useful as during tests leading up to this experiment some flaws were found and rectified. At first, the KBS had trouble matching duplicate and triplicate cell layouts which would have proven problematic later on. The KBS took significant time (± 1 min per machine) to set up, as was discussed in previous chapters, outlining one of the potential disadvantages of the system. The time spent, however, was once-off, so once set up, the system operated rapidly. The system is also simplistic and incapable of learning, but this was known from the outset and was not a requirement of this system.

Conclusion:

The KBS performed as expected and proved to be a suitable method for program assignment in the RMMS. The one hundred percent success rate proved both the functionality of the code and feasibility of the KBS architecture for providing factory floor model assignment to data gathered by the RTLS and DBSCAN combination system.

10.10 Chapter Summary

This chapter documented experiments conducted during the research and included design guidance experiments and performance tests on each of the aspects of the RMMS. Each experiment was constructed in the standard format and included a more detailed discussion and conclusion than was contained in the body of the design work.

PART C

This part covers the detailed laboratory-scale experiment conducted on the integrated RMMS. This part serves as proof of the functionality of the RMMS and aims to showcase the ability of the system to rapidly re-configure the control of a laboratory-scale system after a physical reconfiguration.

11. Chapter 11: Experimentation on the Integrated RMMS

11.1 Chapter Introduction

The experimentation on and testing of the RMMS was conducted in the form of a hypothetical case study representing a real-world type of dynamic factory floor. The system as a whole was tested by performing a factory floor reconfiguration and using the RMMS to respond to the change by switching program switching instructions and thus reconfigure the factory floor control system.

11.2 Introduction to the Experiment:

The experiment aimed to demonstrate the ability of the RMMS to solve system ramp-up problems in a supplier making use of a Reconfigurable Manufacturing System. The problem presented in this experiment was the changeover between optimal layouts, which came about when the product or product mix changed. This experimental process investigated the role of the RMMS in machine controller routine changing, driven by a change in production needs and therefore factory floor layout. The ability of the RMMS to autonomously changeover software routines, based on the discovered factory floor state alone, was to be proven.

11.3 Situation:

The hypothetical supplier provides part manufacturing services to multiple clients and the supplied parts undergo a large amount of customisation with each batch. The nature of the product demand is beyond the scope of this research and experiment.

The manufacturing system had, in a database, all the production plans necessary for the range of products which can be produced. The company aimed to highly streamline and customise their production facility so that the layout is always optimal for the product or mix being produced at the time, easing workflow and materials handling. The exact nature of the optimisation is beyond the scope of this research and experiment. This experiment uses output from a hypothetical generation and aims to prove only the speed and functionality of the RMMS in production system ramp-up.

The geometry and nature of the products was unimportant in the development and testing of this experiment, because, at the level investigated in this research, the RMMS did not require implicit knowledge of the products being produced. All that is required to verify the functionality of the RMMS is the ability to rapidly change programs based on factory floor state. The situation can be visualised as a seasonal change in products, such as heaters and fans.

Operations planning produced the stylised Gantt chart for the year's predicted production shown in Figure 71. The numbers in the bars represent the number of parts to be produced during the Period. The theory and process of the generation of this chart and the factory floors that it produces were beyond the scope of this experiment. The role is only to achieve the reconfigurations predicted by these results. This chart illustrates that four distinct periods of production are needed, each separated by a factory reconfiguration. In the sessions where Product 2 and Product 4 are produced alone (first and third), the factory made use of duplicate cells configured in the same way, to double the factory's output and speed up production. This form of MCM production is only possible with a RMS which features vary fast ramp-up times. Without this attribute, the short production periods would be impractical. The moving of machines allows the materials handling and workflow of the production system to be optimised, allowing faster

unit production and therefore lower unit cost. There exists a trade-off between the cost and time used up in a factory reconfiguration and increased production, the RMMS allows production time needed to decrease and thus allows for a more agile system.

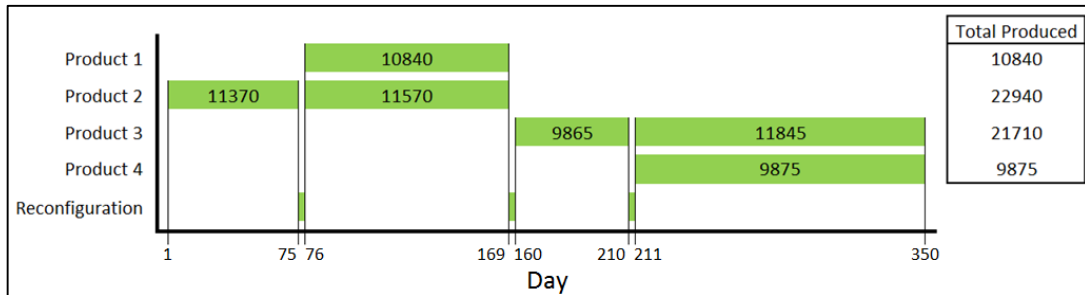


Figure 71: Stylised Gantt Chart for Production

The following part-machine matrices (with cell formation transformations) (Table 28) are known for each product. An explanation follows:

Table 28: Machine-Part Matrices and Cell Formation

Product 1														
		Machine Type							Machine Type					
Part #		1	2	5	6	8	→	Part #		8	5	2	6	1
1		1			1			1		1	1			
2			1	1		1		2		1	1			
3		1				1		3					1	1
4				1		1		4					1	1

Product 2										
		Machine #					Machine #			
Part #		3	5	8	→	Part #		3	5	8
1		1	1			1		1	1	
2			1	1		2		1	1	

Product 3										
		Machine #					Machine #			
Part #		3	5	8	→	Part #		3	5	8
1		1	1			1		1	1	
2			1	1		2		1	1	

Product 4														
		Machine Type							Machine Type					
Part #		1	2	2	4	5	→	Part #		5	2	1	2	4
1		1	1			1		1		1	1			
2				1	1			2				1	1	

The grey areas represent the combinations of machines which form cells. Each product needs a different combination of machines and cells in order to correctly function. This table does not represent the layouts which are formed, but only the cells formed by product-specific needs. These cell combinations were added to each other or for use in the layouts shown in the figure which follows (Figure 72).

Table 29 shows what machines are available to the facility and in the CRC’s knowledge base:

Table 29: Machines Available to the Factory

Machine Name	Quantity Available	Description	Known/Available	
			# of Hardware States	# of Software States
1 (a,b)	2	3D Printer	3	8
2 (a,b,c)	3	CNC Mill	4	9
3 (a,b)	2	CNC Lathe	4	8
4	1	Engraver	2	8
5 (a,b)	2	Robot Arm	4	8
6 (a,b)	2	Press	2	5
7	1	Spot Welder	4	4
8 (a,b)	2	Drill Press	1	4
9	1	Band Saw	2	4
10	1	Punch/Bend	6	8

Appendix P (section 15.16) shows the entire knowledge base as it was stored for the purpose of this test. The database contained multiple different factory floor layouts for the KBS to find a match for the detected layout. The database was designed to be large enough to prove the functionality of the KBS while not being too large to be tedious to design.

The following factory floor models (Figure 72) are taken to be the optimised layouts (produced from Table 28) for the corresponding product cycles. Figure 73, which follows it, shows what each of the reconfigurations requires in terms of machine moving. The generation of these layouts is beyond the scope of this research – only handling of the layout changes is required in this system test. Table 30, included below the figure, gives values for the position of each machine, cellular arrangement, software and hardware configurations required, and duty of each cell. The table serves to describe the layouts in full, with Figure 72 giving a visual representation. The table also serves to show how the software requirements of the system which were outlined – in order to pass the case study, the RMMS had to produce the software outputs shown in this table when given a physical layout.

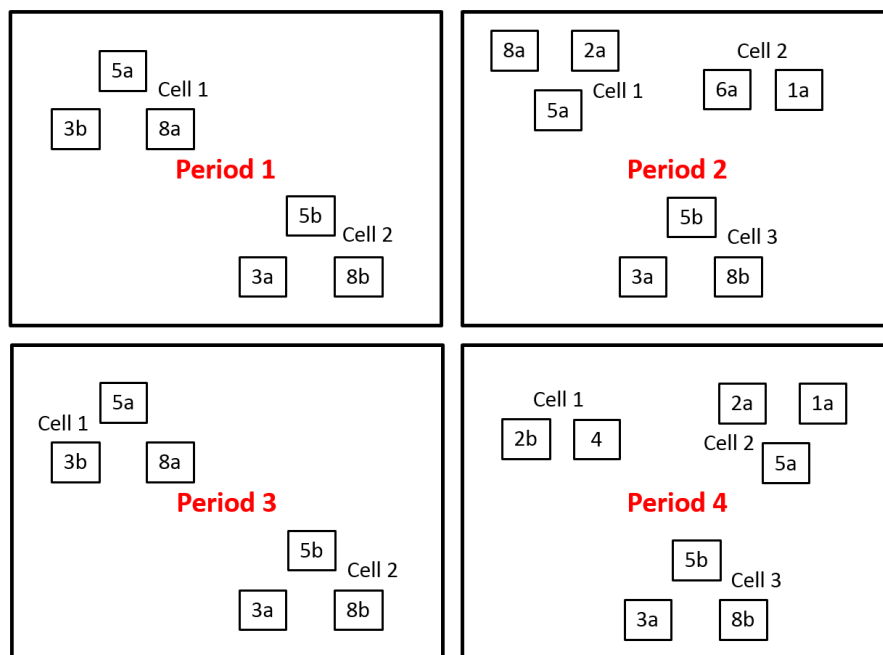


Figure 72: Machine Layout in Each Period

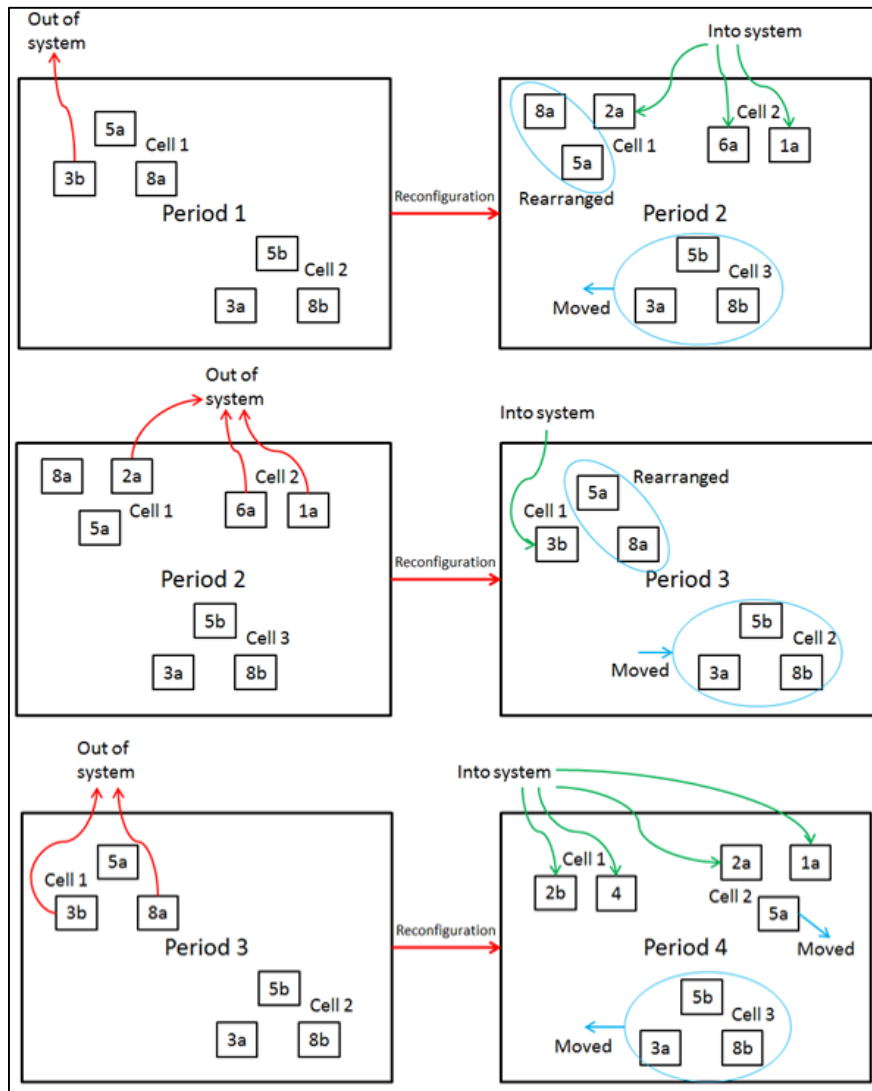


Figure 73: Reconfiguration Operations

Table 30: Layout and Configuration Data for the Machines

Layout 1						Layout 2					
Cell	Machine Name	Position		Configuration #		Cell	Machine Name	Position		Configuration #	
		X	Y	Physical	Software			X	Y	Physical	Software
1	3b	2000	5500	2	4	1	2a	4000	8000	2	5
	5a	3300	7000	2	6		5a	2750	8000	2	4
	8a	4600	5500	1	2		8a	1500	6500	1	1
3a	7000	1500	2	4	2		1a	9500	7000	1	5
2	5b	8300	3000	2		6	6a	7500	7000	2	3
	8b	9600	1500	1		2	3a	7000	1500	2	4

Layout 3						Layout 4					
Cell	Machine Name	Position		Configuration #		Cell	Machine Name	Position		Configuration #	
		X	Y	Physical	Software			X	Y	Physical	Software
1	3b	2000	5500	3	4	1	2b	2000	6500	1	3
	5a	3300	7000	3	7		4	4000	6500	1	2
	8a	4600	5500	1	3	2	1a	10000	7500	1	6
2	3a	7000	1500	3	4		2a	8000	7500	1	3
	5b	8300	3000	3	7		5a	6000	9000	3	3
2	8b	9600	1500	1	3	3	3a	7000	1500	3	4
	5b	8300	3000	3	7		5b	8300	3000	3	7
	8b	9600	1500	1	3		8b	9600	1500	1	3

11.4 Aim

The experiment associated with the situation described above aimed to prove that it is possible, using the technology developed in this research, to rapidly provide software switching instructions to all the machines on a factory floor based on the physical state of that factory floor.

This experiment aims for a 100% success rate. That is, every one of the four factory floor layouts has to be successfully analysed and have the correct software switching instructions assigned to the correct machines.

11.5 Experimental Methodology

In order to verify the success of the RMMS as a system the three reconfigurations (and the initial layout) needed to be physically laid out and the RMMS must have sent program switching instructions to each machine.

The experiment was conducted in the UKZN Manufacturing Mechatronics and Robotics Laboratory by building the configurations as described before and physically running the RMMS in order to change the output of the SCMs. Actual production programs were not implemented. Simplified, hypothetical programs were loaded onto each of the machine controllers and these were switched between to demonstrate that the functionality was obtained. This proved the ability of the SCMs to reconfigure software on the machine controllers and the RMMS as a whole. Thus the methodology for the experiment was as follows:

1. Configure Layout 1.
2. Use the Central RMMS Control software to initiate a factory floor scan.
3. Allow the CRC to build a factory floor state model.
4. Record the factory floor model as given by the CRC.
5. Use the CRC software to initiate a software reconfiguration.
6. Check and record the SCM outputs to confirm software switching.
7. Reconfigure the factory floor into the next Layout and repeat steps 2-6.

11.6 Results

The four tables which follow (Table 31 to Table 34) show the results gathered in each of the four reconfiguration processes. The 'Measured' columns represent what was received by the CRC after the factory scan was completed, the raw results are contained in Appendix Q (section 15.17), as represented in the CRC. The calculated cell number is shown in the table and shows what cell was produced by the DBSCAN algorithm from the raw measured data. The cell number is also contained in the CRC representation shown in Appendix O and was recorded in the tables that follow. The 'Output' column shows the output of the SCMs terminals, as supplied the machine controllers, and as was measured manually. The result column shows whether or not the test was successful.

Table 31: Results of Layout 1 Reconfiguration

Machine Number	Result	Measured				Calculated	Sent	Result
		Machine Type	Physical Configuration	X Position	Y Position	Cell Number	Output	
1	Expected	CNC Lathe	2	2000	5500	1	0001100	Pass
	Experimental	CNC Lathe	2	2324	5462	1	0001100	
2	Expected	Robot Arm	2	3300	7000	1	0001110	Pass
	Experimental	Robot Arm	2	3152	7278	1	0001110	
3	Expected	Drill Press	1	4600	5500	1	0001010	Pass
	Experimental	Drill Press	1	4986	5486	1	0001010	
4	Expected	CNC Lathe	2	7000	1500	2	0001100	Pass
	Experimental	CNC Lathe	2	6831	1271	2	0001100	
5	Expected	Robot Arm	2	8300	3000	2	0001110	Pass
	Experimental	Robot Arm	2	8417	2584	2	0001110	
6	Expected	Drill Press	1	9600	1500	2	0001010	Pass
	Experimental	Drill Press	1	9174	1098	2	0001010	

Table 32: Results of Layout 2 Reconfiguration

Machine Number	Result	Measured				Calculated	Sent	Result
		Machine Type	Physical Configuration	X Position	Y Position	Cell Number	Output	
1	Expected	Drill Press	2	4000	8000	1	0001101	Pass
	Experimental	Drill Press	2	3788	8114	1	0001101	
2	Expected	Gantry Mill	2	2750	8000	1	0001100	Pass
	Experimental	Gantry Mill	2	2714	8418	1	0001100	
3	Expected	Robot Arm	1	1500	6500	1	0001000	Pass
	Experimental	Robot Arm	1	1712	6281	1	0001000	
4	Expected	Pnumatic Press	1	9500	7000	2	0001101	Pass
	Experimental	Pnumatic Press	1	9431	6924	2	0001101	
5	Expected	3d Printer	2	7500	7000	2	0001011	Pass
	Experimental	3d Printer	2	7668	6841	2	0001011	
6	Expected	CNC Lathe	2	7000	1500	3	0001100	Pass
	Experimental	CNC Lathe	2	7320	1417	3	0001100	
7	Expected	Robot Arm	2	8300	3000	3	0001110	Pass
	Experimental	Robot Arm	2	8408	2811	3	0001110	
8	Expected	Drill Press	1	9600	1500	3	0001010	Pass
	Experimental	Drill Press	1	9442	1691	3	0001010	

Table 33: Results of Layout 3 Reconfiguration

Machine Number	Result	Measured				Calculated	Sent	Result
		Machine Type	Physical Configuration	X Position	Y Position	Cell Number	Output	
1	Expected	CNC Lathe	3	2000	5500	1	0001100	Pass
	Experimental	CNC Lathe	3	1891	5668	1	0001100	
2	Expected	Robot Arm	3	3300	7000	1	0001111	Pass
	Experimental	Robot Arm	3	3516	7664	1	0001111	
3	Expected	Drill Press	1	4600	5500	1	0001011	Pass
	Experimental	Drill Press	1	4483	5715	1	0001011	
4	Expected	CNC Lathe	3	7000	1500	2	0001100	Pass
	Experimental	CNC Lathe	3	7235	1698	2	0001100	
5	Expected	Robot Arm	3	8300	3000	2	0001111	Pass
	Experimental	Robot Arm	3	8029	3281	2	0001111	
6	Expected	Drill Press	1	9600	1500	2	0001011	Pass
	Experimental	Drill Press	1	10074	1492	2	0001011	

Table 34: Results of Layout 4 Reconfiguration

Machine Number	Result	Measured				Calculated	Sent	Result
		Machine Type	Physical Configuration	X Position	Y Position	Cell Number	Output	
1	Expected	CNC Lathe	1	2000	6500	1	0 0 0 1 0 1 1	Pass
	Experimental	CNC Lathe	1	1912	6187	1	0 0 0 1 0 1 1	
2	Expected	Gantry Engraver	1	4000	6500	1	0 0 0 1 0 1 0	Pass
	Experimental	Gantry Engraver	1	4220	6321	1	0 0 0 1 0 1 0	
3	Expected	CNC Lathe	1	10000	7500	2	0 0 0 1 1 1 0	Pass
	Experimental	CNC Lathe	1	10229	7695	2	0 0 0 1 1 1 0	
4	Expected	3d Printer	1	8000	7500	2	0 0 0 1 0 1 1	Pass
	Experimental	3d Printer	1	7659	7774	2	0 0 0 1 0 1 1	
5	Expected	Robot Arm	3	6000	9000	2	0 0 0 1 0 1 1	Pass
	Experimental	Robot Arm	3	5845	9410	2	0 0 0 1 0 1 1	
6	Expected	CNC Lathe	3	7000	1500	3	0 0 0 1 1 0 0	Pass
	Experimental	CNC Lathe	3	7184	1529	3	0 0 0 1 1 0 0	
7	Expected	Robot Arm	3	8300	3000	3	0 0 0 1 1 1 1	Pass
	Experimental	Robot Arm	3	8029	2684	3	0 0 0 1 1 1 1	
8	Expected	Drill Press	1	9600	1500	3	0 0 0 1 0 1 1	Pass
	Experimental	Drill Press	1	100144	1655	3	0 0 0 1 0 1 1	

11.7 Experimental Discussion

The tables shown in the previous section (11.6) show that the RMMS was successful in its factory floor scanning, model assignment and ability to provide software switching instructions.

Each process, from the initialisation of the program to when the assignment of program switching instructions took between 16min and 21min to complete. The process took longer for the layouts with more machines, primarily because the receipt of factory floor information from each module happens in a serial manner, as does the sending out of software switching instructions. Thus the size of the factory floor affects the software reconfiguration time. This process is still very quick in industrial terms and is far more time and labour efficient than manual methods and it operated well faster than was dictated in the specifications set out in the beginning and shown in Table 1.

There were, unfortunately, a number of issues experienced during the experimentation. Just under 50% of the time the process failed at some point due to a breakdown of communications. This research did not focus on the development of the communications protocol, but rather used the in-built transparent, or AT, mode of the XBee modules. The process itself worked very well when communications were transmitted correctly, so the experiment was deemed a success. It is known that some improvement is required in order to improve the robustness of the system. This is left to further research.

The system failed to find the factory floor on its first attempt of Layout 2. There was an uncertain anomaly which caused two of the machines to have very inaccurate location data. This was probably because there was some anomalous effect on signal strength readings from one of the beacons and because the two machines lay near the edge of the system and one or more of the required intersections fell outside the factory floor boundaries and were rejected (as is discussed in section 3.6.3). The modules in question were 8a (Drill Press) and 3a (CNC Lathe), see Figure 72, Table 30 and Table 32. In this case the system was simply re-run and it produced an accurate result. The failure was picked up during the KBS matching process (see Figure 44) and so only about ten minutes extra was used in the reconfiguration process. The system was re-run and was successful on its second attempt. This gave the system a success rate of 4/5, or 80%, which was slightly lower than expected, but still within an acceptable range (see the customer specifications

in Table 1). It was difficult to judge the true value of this figure with a small data set, but the fact that it managed to perform all four reconfiguration ramp-up operations indicated a success.

The short time needed for the completion of the ramp-up/software reconfiguration process meant that it could afford some failures, especially at the prototype/experimental phase of its life cycle, where it currently is. With some improvements to the communications this system would speed the process up even more, because the average time needed would go down as the success rate goes up.

It can be seen in each of the tables that the position estimations were very similar in their inaccuracies to the data found and used in previous experiments (10.6, 10.7 & 10.8). The DBSCAN handled these more accurately than was found before (in section 10.8), with an accuracy of 100% rather than around 90-95%. This is likely because of the more ideal system layout, which provided a larger margin for error than the previous experiment, which was designed to test the limitations of the system, and allows the DBSCAN algorithm to perform with a higher rate of success.

It could be argued that the test was done in favourable conditions, the cells were compact and relatively far apart, but it should be remembered that this was done as a proof of concept and not an outright industry performance test. In this test the system performed very well and could be made industry ready with a few minor changes – especially a further refined RTLS and better communications.

The CRC was set up to involve some user input at every stage (basically just to check that each step was completed successfully before going on to the next stage), but this process could very easily be automated. That is, with slightly better reliability this process could be entirely without user interaction from the point of factory floor scan initialisation to when programs have been assigned.

11.8 Experimental Conclusion

The system experienced some difficulties with communication breakdowns and one inaccurate RTLS function, but even so, it managed to successfully re-assign programs to the appropriate machines based on the factory floor state in every scenario. Although some of the attributes of the RMMS did not function with total effectiveness, the core functions of the system were successful for each of the four tested factory floors.

The system operated rapidly (in minutes, rather than the hours required for manual reconfiguration and within the customer specifications) and with sufficient accuracy to be deemed a success. Although this experiment, and research in general, does not address speed of physical reconfiguration, it can be seen that this makes the ramp up much faster, so that there is more room to work with on the physical side. This experiment proved that it is indeed possible to provide a rapid ramp-up of operations after a reconfiguration with minimal human input. The system was low cost (each prototype module cost under R900) in comparison to much more expensive alternatives (see sections 3.6.2 and 12.7.1) – which, in fact, have less functionality than the RMMS.

Every one of the four layouts was successfully detected and in each case all of the SCMs had the correct binary output for switching software, so the aim of the experiment was met. The

successful result of this system experiment proved the functionality of the RMMS and demonstrated how the RMMS will improve the feasibility of the RMS paradigm.

11.9 Chapter Summary

This chapter introduced a realistic problem at a laboratory-scale which would cause down-time in a traditional RMS, but which could be made feasible by the RMMS. The situation was explained in sufficient detail and the way the RMMS could aid the problem was explained. Each of the factory floor states was laid out in the laboratory and the experiment documented in this chapter discussed the operation of the RMMS. The previous two sections (11.7 & 11.8) discussed and drew conclusions on the results of the experiment, which were successful and proved the RMMS to be a good solution to the introduced problem.

PART D

This part of the dissertation includes a detailed discussion on the performance of the RMMS, the research findings, shortcomings and contributions and the conclusion of the research, including a discussion on possible future work.

12. Chapter 12: Discussion

12.1 Chapter Introduction

This section discusses the findings of this research and contributions made to the state of the art in reconfigurable manufacturing technology. The possibilities for further system improvements which could warrant further work are outlined in this chapter. This chapter is essentially a detailed summary of the research, including insights and interpretations. The necessity and role of the RMMS is discussed and a review of the experimental results is conducted. The two primary functions, determination of factory state and intelligent program assignment, of the RMMS are discussed and summarised. Each of the objectives laid out in the introduction (section 1.5) is addressed here and the research contributions and their impact on the RMS paradigm are outlined and considered. This chapter also introduces the system cost discussion and its weaknesses, caveats, and shortcomings are examined. At the end of the chapter there is a discussion on the possible improvements and further research which could aid the current implementation of the RMMS in its industrial feasibility.

12.2 The RMMS in the Context of Reconfigurable Manufacturing

The aim of this research was to identify and solve problems and address research gaps surrounding the reconfiguration of RMSs. This research started with a summary of the current body of knowledge regarding the RMS paradigm, the goal of which was to get to know the manufacturing technique and its supporting technologies and to find gaps in the research regarding the ramp-up time and heterogeneity in machine controllers. The literature review found that there had been research into handling heterogeneity, but that it was based on complex computer science methods. It was decided that a simpler and less software-intensive method should be designed. Further research revealed that one of the problems associated with reconfiguration operations is the slow process of assigning new programs to each of the machine controllers and setting them to run the right program for the right physical state. This helped define the core project aim, to speed up the ramp-up of RMS operations after a reconfiguration which had these objectives:

- ◆ handle control heterogeneity;
- ◆ gather the factory floor state;
- ◆ develop a factory floor model based on the gathered state;
- ◆ use the model to re-assign programs to machine controllers;
- ◆ autonomously re-establish control through program assignment;

Research and consideration showed that this process is traditionally slow and labour intensive and that it makes reconfigurations less financially viable due to down time and production ramp-up complications. At the end of the literature review it was decided that the research would be focussed on removing as much human intervention during the control system reconfiguration.

During this research a technique and its supporting technology was developed to gather factory floor information and communicate it back to the controller in order to make a control decision and feed these new control instructions back to the factory floor. This required that the system would be able to, with no human-aided measurement techniques, build a software-level factory floor model based on what is physically present on the factory floor and to use this model to make control decisions. Re-establishing control on individual machines by implementing appropriate program switches, according to new physical configurations, was the core goal of the RMMS and

is the final outcome of the research. The ability to change control instructions with an automated system reduces the need for human management of the software reconfiguration of the RMS during the ramp-up process.

12.3 Experimentation Results

It has been stated throughout this dissertation that the RMMS was made up of distinct, interacting sub-systems, each with its own objectives. Each of the sub-systems can be divided into two groups, one for gathering of the physical state of the factory floor and one for the modelling of that gathered configuration and updating of control information of that factory floor, based on that model. Broad experimentation on each of the sub-systems was conducted throughout the design process (Chapter 10) so that weaknesses and other caveats in the system could be identified early on and solutions could be found. The table below (Table 35) shows a summary of the results of each of the sub-systems tested in section 10 and results of the final system test (the last row). The table which follows is a reminder of the results of the experiments for this chapter. Each experiment has an indication of the ‘level’ of success, stated subjectively – for instance, if a test met the requirements it was deemed ‘successful’ and if it exceeded them it was deemed ‘very successful’.

Table 35: Summary of All Experimental Results

Test	Results Summary		
	Accuracy of Results	Comments	Result
SCM Electronics Tests	100% Pass for each aspect	Each of the electronic subsystems worked as designed	Successful
Logarithmic Drop-Off of RSSI with Distance	Proved concept, but was inaccurate (did not fit a standard function well)	The experiment proved the logarithmic data, but the results were scattered ($R^2=0.9$). It showed that the ATDB method was not suitable	Somewhat Successful
Linear Change in PWM with Distance	Proved concept within error margins ($\pm 600\text{mm}$)	The results showed much better consistency ($\sigma < 7.5$) and a closer match to a function ($R^2=0.99$) the method proved to be feasible for use in the RTLS	Successful
Test of RSSI Distance Estimation	Passed 96% of the time (fell between -250mm and $+1250\text{mm}$ of the true distance)	The results showed some inconsistency, but were within the needs of the DBSCAN algorithm almost all the time. The sample size was not large, but showed Gaussian distribution	Mostly Successful
Theoretical Test of Multilateration	Within 600mm of the true position $>90\%$ of the time, with approximately Gaussian distribution	The results from the distance estimation experiment were more accurate than those in this experiment. The examples were designed to stress the algorithm and so the high pass rate proved the feasibility of	Successful
Roaming Node of RTLS	Passed 100% of the time	Every case tested showed good results with the needs of DBSCAN, even though the sample was small. This proved the feasibility of the RTLS	Very Successful
DBSCAN Test	Passed 90% of the time	The examples were specifically designed to be difficult (worse than what was expected), to stress the algorithm and test its boundaries, so the high pass rate was a	Very Successful
KBS Test	Passed 100% of the time	The examples were reasonably difficult and were used to simulate a more difficult environment than the final test	Successful
RMMS Overall System Test	Passed 80% of the time	Communications breakdowns occurred, but when the comms were successful, the system operated within the expectations. The system produced program assignments based on system state for every factory floor	Successful

12.4 Determination of the Factory Floor State

A core objective of the research and a requirement of the RMMS was the ability of the system to determine the physical configuration of the factory floor for the creation of a model. The creation of a model required a real-time location system, a refinement technique and a method for gathering the physical configuration of the machine.

12.4.1 Gathering Machine Configuration

The gathering of the physical configuration of the machine attached to the module was not investigated in detail in this research. Previous research on controllers which are aware of their configuration has been conducted [8] and a slight modification on this technique was adopted for use in the RMMS. The system used binary messages programed into an aware controller to convey the physical state for modelling purposes.

12.4.2 The Real-Time Location System

An objective of the RMMS was the development of a suitable RTLS, which was needed to aid in the determination of the factory floor state. The process of researching and developing a suitable RTLS and its refining technology for the RMS environment took up much of the research period. There was ample literature on the broad nature of various techniques for localisation even though the details published on the implementation were minimal. For this reason a lot of testing, troubleshooting and adapting was documented during the design process and only the most relevant and final implementations are shown in the design and experiment sections on the subject (10.3 – 10.7). Along with the design of the RTLS, a suitable published technique for clustering (DBSCAN) was adapted for use in the object orientated environment of the CRC. This took less work than the development of the RTLS but its implementation was very important to make the low cost (almost no additional expense over what was already available because of the wireless communications) RTLS a realistic option for the RMMS. Even though the RSSI technique for distance estimation did prove to be less reliable than initially thought, it was accurate enough that when paired with a tolerant multilateration algorithm and the DBSCAN process it produced results which were within the accuracy band able to produce the required outcome. Thus this research contributed an improvement in low cost indoor RTLSs.

The RTLS was able to locate machines on the experimental factory floor to within a radius of 600mm over 95% of the time (over the two experiments, see sections 10.7 and 11.7). Experiments were also conducted in the design phase of the RTLS where the performance which was gathered in reality was predicted. The performance of the multilateration algorithm was tested (section 10.6) using position estimation data gathered in a previous experiment (section 10.5) and reflected results which were well matched in the later experiments on the RTLS with a roaming node (section 10.7). The results of the RTLS functionality are summarised in Figure 74.

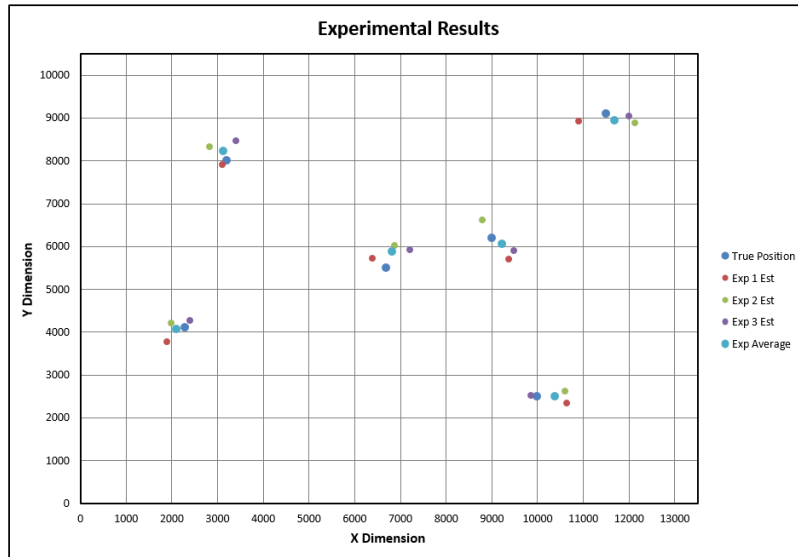


Figure 74: Results of RTLS Roaming Node Test

12.4.3 The DBSCAN Refinement Algorithm

The addition of DBSCAN to the RTLS in order to refine results in a way useful in the RMS paradigm was an essential adaption which made a reasonably inaccurate but very cost-effective method for localisation feasible for the RMMS. The DBSCAN algorithm proved (in section 10.8) to be able to cluster the modules effectively with variations larger than those found in the RTLS experiments discussed above. The DBSCAN algorithm was able to correctly cluster simulated modules at an accuracy of around 90% when variations were limited to the extremes found in the RTLS tests (to within a radius of 600mm of the true position).

12.4.4 Performance of the State Determination System

The combination of the methods proved to be within the accuracy band which allows the KBS to make accurate software assignments for every one of the factory floor layouts tested in the system experimentation (Chapter 11). The system required a re-run when the RTLS failed on one occasion, but the second attempt worked correctly. The system was considered a success and met the objectives needed for the correct operation of the KBS.

12.5 The Intelligence System and Program Assignment

There was a problem identified in the research with the re-establishment of control after a reconfiguration, which was dealt with in this research by program assignment based on a factory floor model. One of the core objectives of the research was to deal with the actual reconfiguration of control programs on heterogeneous controllers (a switching system was used) when the physical state of the system was changed. This system for the assignment of programs required a factory floor model to be built from information gathered by the RTLS and DBSCAN and physical configuration read from the machine in order to reconfigure the control system.

12.5.1 Hardware-Supported Middleware

The hardware-supported middleware was the part of the research which had to handle heterogeneity in the factory floor controllers. This problem was originally dealt with using complex computer science techniques, but it was decided that a simpler, more universal, system would be needed for this research. The SCM, along with the definition of instructions in binary terms, allowed programs to be switched to on practically any factory floor machine controller.

This method would take a measure of time and effort to set up (the controllers all need to be programmed to hold all the necessary routines and the mapped instructions need to be known to the KBSs knowledge base), once completed it allows the factory floor to ramp-up to a working, controlled state much more rapidly than would otherwise be possible. The time needed for setup can be approximated to 1 minute per machine on each layout. The experiment on the system shown in Chapter 11 discusses how the system was able to reconfigure the control of a factory in under 25mm, as opposed to many hours of manual work using traditional methods.

The simple, but effective, method switching of software was implemented to maximise compatibility and remove heterogeneity on the factory floor, from the view of the central controller. Even though the method of switching programs based on set binary messages (see section 5.5) does take substantial work to set up, it makes the adaption of factory floor control after a physical reconfiguration rapid. A core objective of this research was to quickly ramp-up a factory floor after a reconfiguration so that changes and adaptations of the RMS can be made more economically viable and the flexibility of the system could be increased.

12.5.2 The Knowledge-Based System

The KBS acted as a simple form of intelligence, relying on a set of equivalence rules in an inference engine and data in a knowledge base to make program assignments. The system worked by matching the physical state found by the RTLS and DBSCAN and using the matched state to provide program assignment information. The KBS also served to send this information out to the correct controllers via the wireless network.

The standalone KBS was tested (10.9) in a simple way, by using a hypothetical database and a simulated factory floor which matched one of the database layouts and testing whether or not the system made the match. The KBS made every match as expected and made no miss-assignments during the experimentation. These results were mirrored when the KBS was tested as part of the whole RMMS. In each of the cases where the correct factory floor information was passed to the KBS, it made the appropriate match and was able to reconfigure the control system successfully. Thus the KBS had no weaknesses from an operations point of view.

12.5.3 The Reconfiguration of Control

Control reconfiguration was achieved through the binary software switching instructions output by the SCMs, these were sent to the modules by the CRC's KBS. The sending of these instruction was the final objective for the system in a reconfiguration operation. The system test covered in Chapter 11 showed that the correct assignment of programs was made in every case where the state determination system (the configuration detection, RTLS and DBSCAN) worked correctly. Because of the 100% success rate of the program assignment system, the objective of control reconfiguration was met by the KBS and middleware sub-systems of the RMMS.

12.6 Post Reconfiguration Control Reconnection

This problem, identified in the literature review, was not focussed on in the research. The RMMS actually never loses long-term communications between the central controller and SCMs but this has not been developed to a state of live factory control, so machines still have to be re-connected to a control network after a reconfiguration. This is something that could be addressed in later research (see section 13.4)

12.7 Discussion on the RMMS

The RMMS was developed to meet a need in the emerging RMS paradigm for more rapid, less error-prone, and more economically viable reconfiguration operations. The implementation of the RMMS in this research was able to reduce the room for human error in the process, while decreasing the time and amount of employment hours needed to perform a reconfiguration. The system performance, while not at the level of a marketable industry product, was good for a first generation prototype. Room for further development has been identified (see section 12.10, below), but as an experimental system, the RMMS was successful.

12.7.1 On the Cost of the System

Because the RMMS (specifically the SCMs and BPMs) was built from off-the-shelf components and was designed from the outset to aim for low cost, it proved to be an affordable option which should prove to be attractive to RMS designers. Each SCM used in the prototype system cost approximately R900, each BPM cost approximately R650, and equipment for communication with the control computer cost under R500.

There is no system currently available which meets the functionality of the RMMS, so its cost is difficult to compare. Even when the RTLS and its associated processing engine (DBSCAN) are considered alone the system proves to be cost effective. When compared to commercial localisation systems such as Zebra DART® [105] and BeSpoon® [106], the developed RTLS and DBSCAN system costs an order of magnitude less. These systems, while more accurate than the standalone RTLS, would have a similar ability to produce a factory floor model for use by the KBS as the combination of the developed RTLS and DBSCAN. The positioning system is essentially added at almost no extra cost because the SCMs have no added cost for the added functionality of the RTLS when XBee wireless modules were used for communications already.

12.7.2 Management System Performance

The data gathered in the simulated RMS production environment, documented in the previous chapter (11), showed that the system was able to switch control instructions based on the factory floor state, with minimal human interaction or guidance. The experiment showed the outcome of the management by manually checking that the modules sent out their particular control switching instructions, which would complete the system ramp-up after a physical reconfiguration. The experiment proved the management capabilities of the system and how it can reduce the direct role of a human towards supervision while also reducing time consumption and preventing errors.

The system performance has been discussed in this dissertation, from the point of completion of the system experiment and can be summarised as well within the objectives of the project. The system performed well for a prototype and with some further development promises to bring a marketable level of technology. The current implementation of the RMMS can be described as experimental, but further development could lead it to be an industrially feasible system which will improve the reconfigurability of a RMS.

Testing done during this research was directly aimed at performance in the UKZN Manufacturing Mechatronics and Robotics Laboratory. This was done because this research was aimed at being part of the factory of the future which is currently under-developed in that research space. This method may not correlate directly to industrial conditions, but the modular and adaptable nature of the RMMS means that it can be altered and scaled for use in many different industrial scenarios.

Unfortunately, the communication system caveats were not identified because the nature of the early tests did not stress the communication network as much as the actual operation did. For this reason the breakdowns in communication, which proved to be the biggest problem in the proof of concept experiment (see section 11.7), were not encountered before and at the system experiment stage the decision was made to leave it for further development.

12.8 Research Contributions

At the beginning of this research a gap in the literature was identified and any reconfiguration operation required a manual gathering and recording of the factory floor and manual upload of programs onto each machine controller. This research aimed to develop a system, including its supporting technology, capable of completing this process with minimal human intervention.

The major contributions included in the research were the development of; a ‘physical’ (a thin hardware-supported layer) middleware layer to deal with heterogeneity the factory floor control, a factory floor positioning system, a data processing system for the refinement of location data, and an intelligence system for the assignment of new programming instructions based on the physical state of the factory floor. The following was developed during the research:

- ◆ State Communication Module (SCM) (see Chapter 5) and Central Reconfiguration Management and Middleware Control software (CRC) (see Chapter 7) for a hardware-supported heterogeneous communication middleware layer. The technique made use of XBee Series 2 modules for wireless communication (see sections 3.6.1 and 5.3.1). This middleware system can deal simply, but effectively, with multiple different controller types.
- ◆ A Received Signal Strength Indication (RSSI) based multilateration system built into the SCM as a Real-Time Location System (RTLS) (see sections 3.6.2 & 6.4). This system was adapted from other examples in the literature, but much of the development went beyond what was found in the research body at the time.
- ◆ The RTLS data needed to be processed in order to represent the factory floor as a model. The DBSCAN algorithm in the literature (see section 3.6.6) was adapted for use in the object orientated CRC program so that it was able to cluster machine objects based on their position and from those clusters create cell objects (see section 8.2.2)
- ◆ A Knowledge-Base System was implemented as the system intelligence to re-assign control programs to machine controllers (see section 3.6.6). It uses an equivalence method to compare the state of the discovered factory floor to a database of known states to find a match (see Chapter 8.3). The system then uses that match to assign programs to each of the machines in that layout.

Minor contributions of this research include a method for enumerating RMS factory floor states (see section 2.7) and identification of research gaps in the literature on RMSs (see section 2.8), allowing other researchers to focus their efforts.

12.9 Impact of the Research Contributions

The contributions of the research discussed above will have an effect on the RMS paradigm as a whole. The decrease in ramp-up time achieved by the RMMS makes every reconfiguration operation more economically viable, because down time is reduced. With the more economic

reconfiguration option available to a RMS manager, the ability of the RMS to rapidly adapt to market changes is increased. With faster ramp-up times, a RMS can take better advantage of opportunities in manufacturing and changes in customer driven demand.

The management ability of the RMMS also effectively reduces the number of man-hours needed to complete a reconfiguration, which would save money during the process. These savings also make reconfiguration a more attractive notion and increase the attractiveness of the RMS paradigm for any customer-driven mass customisable manufacturing environment.

12.10 Opportunity for Further Research and Improvements

There is room for improvement of the RMMS with the communication protocol and technique for communication during the RTLS operation. Some workarounds were needed to get the system to function and consistency was an issue, as discussed in section 11.7. As a concept, the RSSI based RTLS showed great results, it is only the implementation of a more consistent communication protocol which would benefit from further work. One option would be to implement API mode on the XBees as opposed to the AT mode [59] and another would be to adopt another protocol such as DASH7 [44], which is a maturing technology. The breakdowns in communications which were experienced in the testing were not identified early on and the development or modification of a communication protocol were deemed beyond the scope of this research (see sections 3.6.1 & 5.3.1). Thus the communications problems experienced, while a design fault in hindsight, should not reduce the feasibility of the system as a technique for the management of a reconfiguration process.

On the note of the RTLS, as can be seen in the various experimental results that there is room for improvement in the consistency and accuracy of the distance estimation. The results proved within the acceptable range for the system and the goals set out in the research. Even so, the system could benefit from a better technique for distance estimation. Some alternatives are available which may be more accurate (see section 3.6.3) and further research into these could improve the system's stability and applicability. This, again, was not an error in the design, just room for future work improving the implementation.

Further work is needed for the implementation of SCADA in the system. The framework is built into the CRC software, but the development and implementation of SCADA was beyond the scope of this research. Some developmental research was done on the subject of SCADA (see section 9.4), but this was left at the conceptual stage in order to control the scope of this research.

The development of PLC and other control software which can operate with this system would not be a very advanced step up from what is usually contained on the controllers. The implementation of this is discussed in slight detail in section 5.5, including some conceptual pseudo-code which could be implemented on any machine controller. The actual implementation of control software routines and particular methods possible for use in switching programs based on a bit-wise signal was beyond the scope of the research and is left for further development.

Sections 2.8 and 12.6 mentioned that the problem of reconnection was not addressed in this research, it was removed from the scope. Some improvement could be made to the ramp-up speed of an RMS by using wireless methods and through further research into faster control network reconnection.

12.11 Chapter Summary

This chapter discussed all of the key points necessary for the consideration of the RMMS and this research in general. The performance of the RMMS as a whole was summarised and the links between the objectives defined at the beginning of the research and the outcomes were made throughout this chapter and specifically in section 12.2. This chapter identified clearly the contributions of this research and discussed how these contributions will affect the RMS paradigm as a whole. The possible improvements to the system were discussed in some detail in this chapter and this serves to show the holistic view adopted by the author.

13. Chapter 13: Conclusion

13.1 Chapter Introduction

This chapter summarises and makes conclusions based on the various results gathered in the design and experimentation documented in this work. The aims and objectives of the research are reviewed and compared to the findings and results documented in this dissertation. Conclusions on these findings are drawn to judge the success of the research using properties discussed in the previous chapter. The final section discusses the possible further work on the RMS paradigm which this research will aid and which is implied by the findings of this dissertation.

13.2 Aims, Objectives, Findings and Results

The primary aim of the project was to reduce the ramp-up time of a RMS after a reconfiguration. This was to be achieved with the minimising of human intervention, while eliminating errors and dealing with control heterogeneity.

There was a clear link between the objectives set out in the beginning of this dissertation and the construction of the experiment documented in Chapter 11. The system experiment set out to prove the actual functionality of the RMMS in its entirety, in a scaled down, realistic setting. Thus the successful results found in that section show that the research and development of *a state communication and software switching module and thin middleware layer for reconfiguration management in reconfigurable manufacturing systems* was successful. The results found in the system test were accurate enough to prove the concept and the time savings involved would streamline any regularly changing RMS, increasing its flexibility.

The research presented here found applicable gaps in the current literature and aimed to find solutions to these gaps, the particular objectives revolved around the improvement of the system ramp-up process and the findings showed this. The research proved that it is possible to implement a modular, cost-effective system which can speed up the ramp-up of a reconfigured RMS, while reducing labour costs and human error. This research also found that it is possible to enumerate the possible states in which a RMS can be configured to. During the system development it was found that a very cost-effective RTLS (RSSI-based multilateration) can be complimented by an intelligent clustering algorithm (DBSCAN) to produce usable results. The research found that it is possible to re-assign programs to the appropriate machines by using a KBS intelligence to match the model found by the factory floor scan to a known program state. It was found that adequate homogeneous communications could be established through the use of multi-channel binary signalling, with known messages.

Even though some of the particular aspects of the RMMS did not perform as well as the initial experiments and research had predicted, as a system as a whole it performed as well as was required and was considered a success. Notably, the communications protocol did not perform as well as was expected and the problem of control network reconnection was not addressed. Initially, this research set out to find and solve problems in the implementation of RMSs and gaps left in the body of research. It was found that an improvement could be made to the body of research in the RMS paradigm by developing technology to support the ramp-up of a RMS after a reconfiguration. A research gap was identified: there was an opportunity to improve the ramp-up process by speeding up and automating the tedious task of gathering the factory floor state and updating the control instructions to match the newly configured layout of the RMS. It was decided

that the current method for handling heterogeneity (a thick middleware layer) in machine controllers on the factory floor was overly complex and could be simplified if the heterogeneous data was made homogeneous through simple programming and hardware support.

The findings in the literature review and problem identification led to the development of the system presented in this paper – the thin, hardware-supported Reconfiguration Management and Middleware System or RMMS. The system, which is summarised in Chapter 4 was designed to remove the need for constant human interaction and need for any manual measurements or data entering. The system proved (in Chapter 11) that it was able to gather the state of the factory floor and output software switching instructions without the need for constant human interaction.

This technology was not developed to a marketable level in this research, and it did not aim to do so. This research aimed to test a prototype, laboratory-scale system for aiding the reconfiguration process, with the goal of creating a marketable framework for further development (Chapter 1). The RMMS performed within the requirements to prove the concept of using cost effective methods to rapidly ramp-up a reconfigurable production system after a change in state.

13.3 Conclusions

This research set out to and did successfully develop a hardware-supported reconfiguration management middleware system, and required supporting technology, for the completion of a reconfiguration operation by the implementation of a factory-floor-wide software reconfiguration after the physical change. The system did this based on what physical configuration had been adopted during the reconfiguration and thus on the actual physical state of the system, which removes the need to base the software reconfiguration decision on a small amount of human input.

The system was tested to perform the specified duty in four out of four tests, which gave it a 100% success rate. The system required one re-run during these tests and thus had an 80% efficiency rate, with a cost at least an order of magnitude below even partially competing technologies. The system took around twenty minutes to re-assign control programs, which is a drastic reduction in time compared to manual methods. The research thereby adequately addressed the literature gaps and problems which were identified at the end of the literature review (sections 2.8 – 2.10).

In conclusion, this research helped to make reconfigurations of a RMS more economically viable and streamlined. This was obtained through the reduction of labour hours, human error, and ramp-up time. This streamlining was achieved through the development of the Reconfiguration Management and Middleware System. The RMMS was developed to handle the ramp-up of the system, through the rapid reconfiguration of software, thus speeding up the process. The RMMS developed in this research proved to be a desirable addition to the RMS paradigm, increasing the flexibility of a system in which it is implemented.

13.4 Avenues for Further Research

This research aimed to be directly focussed on the future of manufacturing, designed to be part of a true factory of the future. The RMMS should be envisioned by the reader as part of a highly automated system, which operates almost completely independent of human interaction. The applicability of the current artificial intelligence in the system in may be criticised when referring to this true factory of the future. It is true that at this stage the programs needed for any known state of the RMS would be known independent of the factory floor state. In a more automated system the RMMS (in terms of factory floor state gathering and dealing with the heterogeneous

nature of the machines on the factory floor) may play a different role and the AI would play a more important role. This research looks into the future and aimed to create the framework necessary for the highly automated factory. This area of operational intelligence research is open and could yield profitable results in the future.

In the future, the implementation of the system in the abovementioned factory could make further use of the technologies developed in this research. The state determination technology in the system (the RTLS, DBSCAN and heterogeneous communication abilities) could be used to update the high level software rapidly for a system approaching live control update capabilities. This same technology could be used to aid re-establishment after a system crash or to update the high-level software to aid decision making on the most efficient configuration for the RMS to assume in the next period.

14. References

- [1] Y. Koren, U. Heisel, F. Jovane, T. Moriwaki, G. Pritschow, G. Ulsoy, *et al.*, "Reconfigurable Manufacturing Systems," *Annals of the CIRP*, vol. 48, pp. 527-540, 1999.
- [2] V. Malhotra, T. Raj, and A. Arora, "Excellent Techniques of Manufacturing Systems: RMS and FMS," *International Journal of Engineering Science and Technology*, vol. 2, 2010.
- [3] V. Malhotra, T. Raj, and A. Arora, "Reconfigurable manufacturing system: an overview," *International Journal of Machine Intelligence*, vol. 1, pp. 38-46, 2009.
- [4] B. Xing, G. Bright, N. S. Tlale, and J. Potgieter, "Reconfigurable Modular Machine Design for Reconfigurable Manufacturing System," 2005.
- [5] B. Xing, G. Bright, N. S. Tlale, and J. Potgieter, "Reconfigurable Manufacturing System for Agile Mass Customization Manufacturing," in *International Conference on CAD/CAM, Robotics and Factories of the Future*, India, 2006, pp. 473-482.
- [6] Z. M. Bi, S. Y. T. Lang, W. Shen, and L. Wang, "Reconfigurable manufacturing systems: the state of the art," *International Journal of Production Research*, vol. 46, pp. 967-992, 2007.
- [7] Z. M. Bi, S. Y. T. Lang, M. Verner, and P. Orban, "Development of Reconfigurable Machines," *The International Journal of Advanced Manufacturing Technology*, vol. 39, pp. 1227-1251, 2007.
- [8] B. Xing, G. Bright, and N. S. Tlale, "Modular Mechatronic Control of Reconfigurable Manufacturing System for Mass Customisation Manufacturing," in *International Conference on Competitive Manufacturing*, Stellenbosch, South Africa, 2007, pp. 223-228.
- [9] R. Katz, "Design Principles of Reconfigurable Machines," *The International Journal of Advanced Manufacturing Technology*, vol. 34, pp. 430-439, 2007.
- [10] O. Roulet-Dubonnet, M. Lund, and A. Skavhaug, "IceHMS, a Middleware for Distrebuted Control of Manufacturing Systems," in *Industrial Applications of Holonic and Multi-Agent Systems*, 1 ed Berlin, Germany: Springer Berlin Heidelberg, 2013, pp. 95-105.
- [11] K. Gumasta, S. Gupta, L. Benyoucef, and M. K. Tiwari, "Developing a Reconfigurability Index Using Multi-Attribute Utility Theory," *International Journal of Production Research*, vol. 49, pp. 1669-1683, 2011.
- [12] Y. Koren and M. Shpitalni, "Design of Reconfigurable Manufacturing Systems," *Journal of Manufacturing Systems*, vol. 29, pp. 130-141, 2010.
- [13] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren, "Reconfigurable Manufacturing Systems: Key to Future Manufacturing," *Journal of Intelligent Manufacturing*, vol. 11, pp. 403 - 419, 2000.
- [14] D. Barr, "Supervisory Control and Data Acquisition (SCADA) Systems," N. C. System, Ed., ed. Chantilly, USA, 2004.
- [15] E. Csanyi. (2013, 15/02/2014). *Three generations of SCADA system architectures*. Available: <http://electrical-engineering-portal.com/three-generations-of-scada-system-architectures>
- [16] J. P. Lahti, A. Shamsuzzoha, and T. Kankaanpaa, "Web-based technologies in Power Plant Automation and SCADA Systems: A Review and Evaluation," in *IEEE International Conference on Control System, Computing and Engineering*, Penang, China, 2011.
- [17] S. Karnouskos and T. Bangemann, presented at the IEEE International Conference on Industrial Technology Cape Town, South Africa, 2013.
- [18] *Festo OPC EasyServer Users Manual*. Esslingen am Neckar, Germany: Festo AG & Co. KG, 2004.
- [19] (2014, 15/03/2014). *OPC Foundation Specifications*. Available: <http://opcfoundation.org/Products/Specifications.aspx?CM=-1&CU=1>

- [20] A. Bajahzar, A. Alqahtani, and A. Baslem, "A survey study of the Enterprise Resource Planning system," presented at the International Conference on Advanced Computer Science Applications and Technologies, Kuala Lumpur, Malaysia, 2012.
- [21] M. Younus, L. Hu, F. Yuqing, and C. P. Yong, "Manufacturing Execution System for a Subsidiary of Aerospace Manufacturing Industry," in *International Conference on Computer and Automation Engineering*, Bangkok, Thailand, 2009.
- [22] J. Hua, H. Sun, T. Liang, and Z. Lei, "The Design of Manufacturing Execution System Based On RFID," in *Workshop on Power Electronics and Intelligent Transportation System*, Guangzhou, China, 2008.
- [23] R. Landers, B. Min, and Y. Koren, "Reconfigurable Machine Tools," *CIRP Annals - Manufacturing Technology*, vol. 50, pp. 269-274, 2001.
- [24] T. Lorenzer, "Augmented Flexibility in Manufacturing by Reconfigurable Machine Tools," Swiss Federal Institute of Technology, Zurich, Switzerland, 2009.
- [25] R. G. Landers, B. K. Min, and Y. Koren, "Reconfigurable Machine Tools," *CIRP Annals - Manufacturing Technology*, vol. 50, pp. 269-274, 2001.
- [26] A. I. Dashchenko, "Choice of Optimal Configurations for Flexible (Adjustable) Assembly Lines by Purposeful Search," *Annals of the CIRP*, vol. 40, pp. 13-16, 1991.
- [27] Y. Koren, S. J. Hu, and T. W. Weber, "Impact of Manufacturing System Configuration on Performance," *Annals of the CIRP*, vol. 48, pp. 369-372, 1998 1998.
- [28] K. Hon and F. Lopezjaquez, "Configuration of Manufacturing Cells for Dynamic Manufacturing," *CIRP Annals - Manufacturing Technology*, vol. 51, pp. 391-394, 2002.
- [29] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren, "Reconfigurable Manufacturing System and Their Enabling Technologies," *International Journal of Manufacturing Technology and Management*, vol. 1, pp. 113 - 130, 2000.
- [30] B. Xing, F. V. Nelwamondo, K. Battle, W. Gao, and T. Marwala, "Application of Artificial Intelligence (AI) Methods for Designing and Analysis of Reconfigurable Cellular Manufacturing System (RCMS)," presented at the 2nd International Conference on Adaptive Science & Technology, Accra, Ghana, 2009.
- [31] L. Tang, Y. Koren, D. M. Yip-Hoi, and W. Wang, "Computer-Aided Reconfiguration Planning: An Artificial Intelligence-Based Approach," *Journal of Computing and Information Science in Engineering*, vol. 6, pp. 230-240, 2006.
- [32] A. O. Oke, K. Abou-El-Hossein, and N. J. Theron, "The Design and Development of a Reconfigurable Manufacturing System," *South African Journal of Industrial Engineering*, vol. 22, pp. 121-132, 2011.
- [33] M. G. Mehrabi, A. G. Ulsoy, and Y. Koren, "Reconfigurable Manufacturing Systems: Key to Future Manufacturing," *Journal of Intelligent Manufacturing*, vol. 11, pp. 403 - 419, 2000.
- [34] L. Monostori, J. Váncza, and S. R. T. Kumara, "Agent-Based Systems for Manufacturing," *CIRP Annals - Manufacturing Technology*, vol. 55, pp. 697-720, 2006.
- [35] M. Yu, W. Zhang, and P. Llemm, "Multi-Agent Based Reconfigurable Manufacturing Execution System," presented at the IEEE International Conference of Industrial Engineering and Engineering Management, Singapore, 2007.
- [36] O. Roulet-Dubonnet, "Distributed Control of Flexible Manufacturing Systems: Implementation of a Specialized Multi-Agent Middleware and Applications of Holonic Concepts," PhD, Norwegian University of Science and Technology, 2011.
- [37] H. Naji, "Applying multi agent techniques to reconfigurable systems," *Advances in Engineering Software*, vol. 35, pp. 401-413, 2004.
- [38] A. Willig, K. Matheus, and A. Wolisz, "Wireless Technology in Industrial Networks," *Proceedings of the IEEE*, vol. 93, pp. 1130-1151, 2005.
- [39] A. Willig, M. Kubisch, C. Hoene, and A. Wolisz, "Measurements of a Wireless Link in an Industrial Environment Using an IEEE 802.11-Compliant Physical Layer," *IEEE Transactions on industrial Electronics*, vol. 49, pp. 1265 - 1281, 2002.
- [40] P. L. Fuhr, W. Boyes, P. Chen, H. Kagan, D. Kaufman, and W. Manges, "Wireless Technology Review: Radios, Frequencies, and Implications for Industry," Oak Ridge National Laboratory 2010.

- [41] (2012, 22/02/2014). *Industrial Wireless - Selecting a Wireless Technology*. Available: <http://www.bb-elec.com/Learning-Center/All-White-Papers/Wireless-Cellular/Industrial-Wireless-Selecting-a-Wireless-Technolog.aspx>
- [42] A. Low, "Evolution of Wireless Sensor Networks for Industrial Control," in *ISA Calgary Show*, 2013.
- [43] M. Nixon, "A Comparison of WirelessHART™ and ISA100.11a," Emerson Process Management 2012.
- [44] M. Weyn, G. Ergeets, L. Wante, C. Vercauteren, and P. Hellinckx, "Survey of the DASH7 Alliance Protocol for 433MHz Wireless Sensor Communication," *International Journal or Distrebuted Sensor Networks*, vol. 1, pp. 1-9, 2013.
- [45] J. Stevens, R. Berkvens, W. Looockx, and M. Weyn, "Robot Localization With DASH7 Technology," in *Third International Conference on Ambient Computing, Applications, Services and Technologies*, Porto, Portugal, 2013.
- [46] P. S. Neelakanta and H. Dighe, "Robust Factory Wireless Communications: A Performance Appraisal of the Bluetooth and the ZigBee Colocated on an Industrial Floor," in *29th Annual Conference of the IEEE Industrial Electronics Society*, Roanoke, USA, 2003, pp. 2381 - 2386.
- [47] S. Muenstermann, "Interference and Security Considerations for Wireless Communications in an Industrial Environment," Honeywell Process Solutions 2007.
- [48] J. Song, S. Han, A. K. Mok, D. Chen, M. Lucus, M. Nixon, *et al.*, "WirelessHART: Applying Wireless Technology in Real-Time Industrial Process Control," *IEEE Real-Time and Embedded Technology and Applications Symposium*, 2008.
- [49] G. Wang, "Comparison and Evaluation of Industrial Wireless Sensor Network Standards ISA100.11a and WirelessHART," MSc, Department of Signals and Systems, Chalmers university of Technology, Gothenburg, Sweden, 2011.
- [50] P. Milgrom, J. Levin, and A. Eilat, "The Case for Unlicensed Spectrum," Stanford Institute for Economic Policy Research, Stanford University 2011.
- [51] S. N. Gupta, S. Abraham, and Y. Gyulkhandanyan, "Unlicensed Spectrum Policy Brief," G. o. I. Communications, Ed., ed, 2012.
- [52] "Comparison of Licensed and Unlicensed Spectrum for Utility-Owned Distribution Area Communication Networks," ABB Inc, Sunnyvale, USA 2010.
- [53] R. D. Meininger, J. W. Shank, and R. Shankar, "Electromagnetic and Radio Frequency Interference (EMURFI) Qualification of Digital Equipment for Nuclear Power Generating Stations," in *Nuclear Science Symposium and Medical Imaging Conference*, Norfolk, USA, 1995, pp. 1056-1060.
- [54] X. Zhou, J. J. Shea, and B. Pahl, "Characterization of EMI/RFI in Commercial and Industrial Electrical Systems," in *59th IEEE Holm Conference on Electrical Contacts*, Newport, USA, 2013.
- [55] W. Guo, W. M. Healy, and Z. MengChu, "Impacts of 2.4-GHz ISM Band Interference on IEEE 802.15.4 Wireless Sensor Network Reliability in Buildings," *IEEE Transactions on Instrumentation and Measurement*, vol. 61, pp. 2533-2544, 2012.
- [56] "Considerations for Operation within the 260–470MHz Band," Linx Technologies, Merlin, USA 2012.
- [57] Anritsu, "RF Interference Hunting Techniques: Spot, Find, Fix," Anritsu 2013.
- [58] Motorola, "Spectrum Analysis for Future LTE Deployments," Motorola, Inc 2007.
- [59] T. Lennvall, S. Svensson, and F. Hekland, "A Comparison of WirelessHART and ZigBee for Industrial Applications," presented at the IEEE International Workshop on Factory Communication Systems, Dresden, Germany, 2008.
- [60] "Wi-Fi Technology," Telecom Regulatory Authority: Technical Affairs & Technology Sector 2003.
- [61] G. Lohmann, "Wireless Technology: WirelessHART," Pepperl+Fuchs, Germany 2012.
- [62] T. Bourke. (2007) ISA100.11a Completely Obviates the Need for WirelessHART. *Petro Industry News: Measurement and Testing*. 18-19.
- [63] D. Dickinson, "Industrial Wireless Technology," Phoenix Contact 2006.

- [64] I. S. o. Automation. (2014). *ISA100, Wireless Systems for Automation*. Available: <http://www.isa.org/MSTemplate.cfm?MicrositeID=1134&CommitteeID=6891>
- [65] I. W. C. Institute, "The Technology Behind the ISA100.11a Standard – An Exploration," I. W. C. Institute, Ed., ed.
- [66] S. Y. Shin, "Extending CAN Protocol with ISA100.11a Wireless Network," presented at the International Conference on ICT Convergence, Jeju Island, 2012.
- [67] M. Arsalan, A. Umair, and V. K. Verma, "Dash7: Performance," *IOSR Journal of Electronics and Communication Engineering*, vol. 2, pp. 08-11, 2012.
- [68] R. Mautz, "Indoor Positioning Technologies," Habilitation PhD, Civil, Environmental and Geomatic Engineering, ETH Zurich, 2012.
- [69] R. Mautz, "Indoor Positioning Technologies," *Sechsendachtzigster Band*, vol. 86, 2012.
- [70] C. Rizos and B. Li, "Independent Testing of Locata: A New High Accuracy Indoor Positioning System," U. o. N. S. Wales, Ed., ed, 2011.
- [71] S. P. Banerjee, "Improving Accuracy in Ultra-Wideband Indoor Position Tracking through Noise Modeling and Augmentation," PhD, Electrical Engineering, 2012.
- [72] J. Blumenthal, R. Grossmann, F. Golasowski, and D. Timmermann, "Weighted Centroid Localization in Zigbee-based Sensor Networks," in *IEEE International Symposium on Intelligent Signal Processing*, Alcalá de Henares, Spain, 2007.
- [73] P. Morávek, "Localization in Wireless Energy-Constrained Networks," PhD Engineering, Faculty of Electrical Engineering and Communication Technology, Department of Telecommunications, Czech University of Technology, 2012.
- [74] H. Liu, H. Darabi, P. Banerjee, and J. Liu, "Survey of Wireless Indoor Positioning Techniques and Systems," *IEEE Transactions on Systems, Man, and Cybernetics - Part C: Applications and Reviews*, vol. 37, pp. 1067 - 1080, 2007.
- [75] S. Schwarzer, V. Martin, M. Pichler, and A. Stelzer, "Precise Distance Measurement with IEEE 802.15.4 (ZigBee) Devices," in *IEEE Radio and Wireless Symposium*, Orlando, USA, 2008, pp. 779-782
- [76] M. Saxena, P. Gupta, and B. N. Jain, "Experimental Analysis of RSSI-based Location Estimation in Wireless Sensor Networks," in *3rd International Conference on Communication Systems Software and Middleware and Workshops*, Bangalore, India, 2008, pp. 503 - 510.
- [77] C. Papamanthou, F. P. Preparata, and R. Tamassia, *Algorithms for Location Estimation Based on RSSI Sampling*: Springer Berlin Heidelberg, 2008.
- [78] G. Zanca, F. Zorzi, A. Zanella, and M. Zorzi, "Experimental comparison of RSSI-based localization algorithms for indoor wireless sensor networks," in *Workshop on Real-world Wireless Sensor Networks*, Glasgow, Scotland, 2008.
- [79] Y. Chen, C. Yang, Y. Chang, and C. Chu, "A RSSI-based Algorithm for Indoor Localization Using ZigBee in Wireless Sensor Network," *International Journal of Software Engineering and Knowledge Engineering*, vol. 15, pp. pp. 70-75, 2009.
- [80] O. Oguejiofor, V. Okorogu, A. Adewale, and B. Osuesu, "Outdoor Localization System Using RSSI Measurement of Wireless Sensor Network," *International Journal of Innovative Technology and Exploring Engineering*, vol. 2, pp. 1-6, 2013.
- [81] X. Nguyen and T. Rattentbury, "Localization algorithms for sensor networks using RF signal strength," University of California at Berkeley, Berkeley, USA2003.
- [82] T. Wu, Y. Zeng, H. Zhou, and B. Li, "Wireless Sensor Network Node Location Based on Improved APIT," *Journal of Surveying and Mapping Engineering* vol. 1, pp. 15-19, 2012.
- [83] P. Berkhin, "Survey of Clustering Data Mining Techniques," Accrue Software, Inc 2006.
- [84] R. Xu and D. Wunsch, "Survey of Clustering Algorithms," *IEEE Transactions on Neural networks*, vol. 16, May 2005.
- [85] S. K. Ayramo, T, "Introduction to partitioning-based clustering methods with a robust example," University of Jyväskylä 2006.
- [86] Siemens, *Simatic S7-1200 Easy Book*. Nurnberg, Germany: Siemens AG, 2013.

- [87] H. Eberhardt, J. Haußmann, and R. Jedelhauser, *EduTrainer Compact/Universal* Denkendorf, Germany: Festo Didactic GmbH & Co, 2011.
- [88] Allen-Bradley, *Logix5000 Controllers Quick Start*. Milwaukee, USA: Rockwell Automation, 2009.
- [89] H. Huang, *PIC Microcontroller: An Introduction to Software & Hardware Interfacing*. Clifton Park, USA: Thomson Delmar Learning, 2009.
- [90] A. E. Eiben and J. E. Smith, *Introduction to Evolutionary Computing* New York, USA: Springer-Verlag Berlin Heidelberg, 2010.
- [91] L. C. Jain and J. Kacprzyk, *Soft Computing Techniques in Knowledge-Based Intelligent Engineering Systems: Approaches and Applications*. Heidelberg, Germany: Springer Physica-Verlag, 1997.
- [92] Z. Pawlak, "Rough Set Approach to Knowledge-Based Decision Support," *European journal of Operational Research*, vol. 99, pp. 48-57, 1997.
- [93] K. A. De Jong, *Evolutionary Computation*. New York, USA: MIT Press, 2006.
- [94] A. Laverack. (2013, 10/03/2014). *Dusty Shop Floor Environments: When Removing the Fan is not Enough*. Available: <http://www.logicsupply.com/blog/2013/09/09/dusty-shop-floor-environments/>
- [95] (2014, 28 May). *RS Components Store*. Available: <http://za.rs-online.com/web/>
- [96] R. Faludi, *Building Wireless Sensor Networks*: O'Reilly Media Inc., 2012.
- [97] Digi. (2014, 12 August 2014). *Antenna Options for Xbee Series 2*. Available: <http://www.digi.com/support/kbase/kbaseresultdetl?id=2167>
- [98] Atmel. (2014, 10 August 2014). *Atmega328 Specifications*. Available: <http://www.atmel.com/devices/atmega328p.aspx>
- [99] (2014, 21 May). *SparkFun Electronics Shop*. Available: <https://www.sparkfun.com/>
- [100] M. Margolis, *Arduino Cookbook*, 2nd ed. Cambridge, USA: O'Reilly Media, 2011.
- [101] T. O'Hara and M. Wesselmark, "Battery Technologies: A General Overview & Focus on Lithium Ion," Intertek2009.
- [102] H. Wang, Y. Wang, and S. Wan, "A Density-Based Clustering Algorithm for Uncertain Data," presented at the International Conference on Computer Science and Electronics Engineering, Hangzhou, China, 2012.
- [103] M. Ester, H. Kriegel, J. Sander, and X. Xu, "A Density-Based Algorithm for Discovering Clusters in Large Spatial Databases with Noise," presented at the 2nd International Conference on Knowledge Discovery and Data Mining, Portland, USA, 1996.
- [104] T. Bezzera, "Performance Evaluation of Zigbee Transmission on Grass Environment," presented at the International Conference on Mobile Ubiquitous Computing, Systems, Services and Technologies, Rome, Italy, 2014.
- [105] (2014, 12/03/2014). *DART UWB*. Available: <https://www.zebra.com/us/en/products/location-solutions/dart-uw.html>
- [106] (2014, 12/03/2014). *Technology: UWB Triangulation*. Available: <http://bespoon.com/triangulation/>

15. Appendices

This section contains all data and calculations which would make the main body of the text much less readable.

15.1 Appendix A: P2N2222A Transistor Data Sheet

P2N2222A

Amplifier Transistors

NPN Silicon

Features

- Pb-Free Packages are Available*


MAXIMUM RATINGS ($T_A = 25^\circ\text{C}$ unless otherwise noted)

Characteristic	Symbol	Value	Unit
Collector - Emitter Voltage	V_{CEO}	40	Vdc
Collector - Base Voltage	V_{CBO}	75	Vdc
Emitter - Base Voltage	V_{EBO}	6.0	Vdc
Collector Current - Continuous	I_C	600	mAdc
Total Device Dissipation @ $T_A = 25^\circ\text{C}$ Derate above 25°C	P_D	625 5.0	mW mW/ $^\circ\text{C}$
Total Device Dissipation @ $T_C = 25^\circ\text{C}$ Derate above 25°C	P_D	1.5 12	W mW/ $^\circ\text{C}$
Operating and Storage Junction Temperature Range	T_J, T_{stg}	-55 to +150	$^\circ\text{C}$

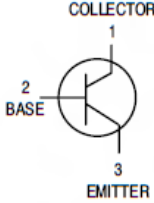
THERMAL CHARACTERISTICS


Characteristic	Symbol	Max	Unit
Thermal Resistance, Junction to Ambient	$R_{\theta JA}$	200	$^\circ\text{C}/\text{W}$
Thermal Resistance, Junction to Case	$R_{\theta JC}$	83.3	$^\circ\text{C}/\text{W}$

Stresses exceeding Maximum Ratings may damage the device. Maximum Ratings are stress ratings only. Functional operation above the Recommended Operating Conditions is not implied. Extended exposure to stresses above the Recommended Operating Conditions may affect device reliability.

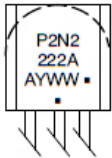


ON Semiconductor®
http://onsemi.com





**TO-92
(T0-226AA)
CASE 29-11
STYLE 17**



**MARKING
DIAGRAM**

P2N2 = Device Code
222A = Specific Device
A = Assembly Location
Y = Year
WW = Work Week
▪ = Pb-Free Package
(Note: Microdot may be in either location)

ORDERING INFORMATION

Device	Package	Shipping†
P2N2222A	TO-92	5000 Units / Bulk
P2N2222AG	TO-92 (Pb-Free)	5000 Units / Bulk
P2N2222ARL1	TO-92	2000 / Tape & Ammo
P2N2222ARL1G	TO-92 (Pb-Free)	2000 / Tape & Ammo
P2N2222AZL1	TO-92	2000 / Tape & Reel
P2N2222AZL1G	TO-92 (Pb-Free)	2000 Units / Tube

†For information on tape and reel specifications, including part orientation and tape sizes, please refer to our Tape and Reel Packaging Specification Brochure, BRD8011/D.

*For additional information on our Pb-Free strategy and soldering details, please download the ON Semiconductor Soldering and Mounting Techniques Reference Manual, SOLDERRM/D.

© Semiconductor Components Industries, LLC, 2006
March, 2006 - Rev. 3

1

Publication Order Number:
P2N2222A/D

P2N2222A

ELECTRICAL CHARACTERISTICS ($T_A = 25^\circ\text{C}$ unless otherwise noted)

Characteristic	Symbol	Min	Max	Unit
OFF CHARACTERISTICS				
Collector-Emitter Breakdown Voltage ($I_C = 10\text{ mAdc}$, $I_B = 0$)	$V_{(BR)CEO}$	40	-	Vdc
Collector-Base Breakdown Voltage ($I_C = 10\text{ }\mu\text{Adc}$, $I_E = 0$)	$V_{(BR)CBO}$	75	-	Vdc
Emitter-Base Breakdown Voltage ($I_E = 10\text{ }\mu\text{Adc}$, $I_C = 0$)	$V_{(BR)EBO}$	6.0	-	Vdc
Collector Cutoff Current ($V_{CE} = 60\text{ Vdc}$, $V_{EB(off)} = 3.0\text{ Vdc}$)	I_{CEX}	-	10	nAdc
Collector Cutoff Current ($V_{CB} = 60\text{ Vdc}$, $I_E = 0$) ($V_{CB} = 60\text{ Vdc}$, $I_E = 0$, $T_A = 150^\circ\text{C}$)	I_{CBO}	-	0.01 10	μAdc
Emitter Cutoff Current ($V_{EB} = 3.0\text{ Vdc}$, $I_C = 0$)	I_{EBO}	-	10	nAdc
Collector Cutoff Current ($V_{CE} = 10\text{ V}$)	I_{CEO}	-	10	nAdc
Base Cutoff Current ($V_{CE} = 60\text{ Vdc}$, $V_{EB(off)} = 3.0\text{ Vdc}$)	I_{BEX}	-	20	nAdc
ON CHARACTERISTICS				
DC Current Gain ($I_C = 0.1\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$) ($I_C = 1.0\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$) ($I_C = 10\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$) ($I_C = 10\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $T_A = -55^\circ\text{C}$) ($I_C = 150\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$) (Note 1) ($I_C = 150\text{ mAdc}$, $V_{CE} = 1.0\text{ Vdc}$) (Note 1) ($I_C = 500\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$) (Note 1)	h_{FE}	35 50 75 35 100 50 40	- - - - 300 - -	-
Collector-Emitter Saturation Voltage (Note 1) ($I_C = 150\text{ mAdc}$, $I_B = 15\text{ mAdc}$) ($I_C = 500\text{ mAdc}$, $I_B = 50\text{ mAdc}$)	$V_{CE(sat)}$	-	0.3 1.0	Vdc
Base-Emitter Saturation Voltage (Note 1) ($I_C = 150\text{ mAdc}$, $I_B = 15\text{ mAdc}$) ($I_C = 500\text{ mAdc}$, $I_B = 50\text{ mAdc}$)	$V_{BE(sat)}$	0.6 -	1.2 2.0	Vdc
SMALL-SIGNAL CHARACTERISTICS				
Current-Gain-Bandwidth Product (Note 2) ($I_C = 20\text{ mAdc}$, $V_{CE} = 20\text{ Vdc}$, $f = 100\text{ MHz}$)C	f_T	300	-	MHz
Output Capacitance ($V_{CB} = 10\text{ Vdc}$, $I_E = 0$, $f = 1.0\text{ MHz}$)	C_{obo}	-	8.0	pF
Input Capacitance ($V_{EB} = 0.5\text{ Vdc}$, $I_C = 0$, $f = 1.0\text{ MHz}$)	C_{ibo}	-	25	pF
Input Impedance ($I_C = 1.0\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $f = 1.0\text{ kHz}$) ($I_C = 10\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $f = 1.0\text{ kHz}$)	h_{ie}	2.0 0.25	8.0 1.25	k Ω
Voltage Feedback Ratio ($I_C = 1.0\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $f = 1.0\text{ kHz}$) ($I_C = 10\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $f = 1.0\text{ kHz}$)	h_{re}	-	8.0 4.0	$\times 10^{-4}$
Small-Signal Current Gain ($I_C = 1.0\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $f = 1.0\text{ kHz}$) ($I_C = 10\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $f = 1.0\text{ kHz}$)	h_{fe}	50 75	300 375	-
Output Admittance ($I_C = 1.0\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $f = 1.0\text{ kHz}$) ($I_C = 10\text{ mAdc}$, $V_{CE} = 10\text{ Vdc}$, $f = 1.0\text{ kHz}$)	h_{oe}	5.0 25	35 200	μMhos
Collector Base Time Constant ($I_E = 20\text{ mAdc}$, $V_{CB} = 20\text{ Vdc}$, $f = 31.8\text{ MHz}$)	$rb'C_c$	-	150	ps
Noise Figure ($I_C = 100\text{ }\mu\text{Adc}$, $V_{CE} = 10\text{ Vdc}$, $R_S = 1.0\text{ k}\Omega$, $f = 1.0\text{ kHz}$)	N_F	-	4.0	dB

1. Pulse Test: Pulse Width $\leq 300\text{ }\mu\text{s}$, Duty Cycle $\leq 2.0\%$.
2. f_T is defined as the frequency at which $|h_{fe}|$ extrapolates to unity.

15.2 Appendix B: Teensy 3.1 Datasheet Excerpt

Symbol	Description	Min.	Typ. ¹	Max.	Unit	Notes
VREGIN	Input supply voltage	2.7	—	5.5	V	
I _{DDon}	Quiescent current — Run mode, load current equal zero, input supply (VREGIN) > 3.6 V	—	120	186	μA	
I _{DDstby}	Quiescent current — Standby mode, load current equal zero	—	1.1	10	μA	
I _{DDoff}	Quiescent current — Shutdown mode <ul style="list-style-type: none"> VREGIN = 5.0 V and temperature=25C Across operating voltage and temperature 	—	650	—	nA	
		—	—	4	μA	
I _{LOADrun}	Maximum load current — Run mode	—	—	120	mA	
I _{LOADstby}	Maximum load current — Standby mode	—	—	1	mA	
V _{Reg33out}	Regulator output voltage — Input supply (VREGIN) > 3.6 V <ul style="list-style-type: none"> Run mode Standby mode 	3	3.3	3.6	V	
		2.1	2.8	3.6	V	
V _{Reg33out}	Regulator output voltage — Input supply (VREGIN) < 3.6 V, pass-through mode	2.1	—	3.6	V	2
C _{OUT}	External output capacitor	1.76	2.2	8.16	μF	
ESR	External output capacitor equivalent series resistance	1	—	100	mΩ	

15.3 Appendix C: UA7812CKCT Datasheet Excerpt

Figure 2. Pin connections (top view)

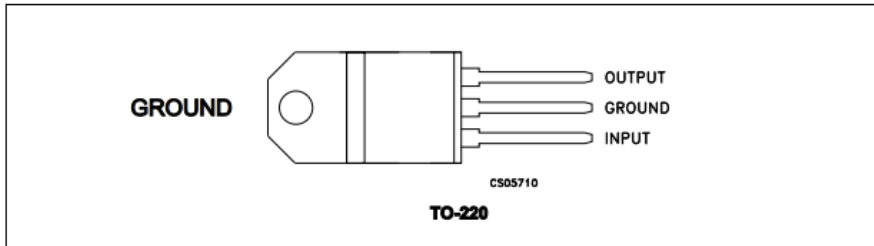


Figure 3. Schematic diagram

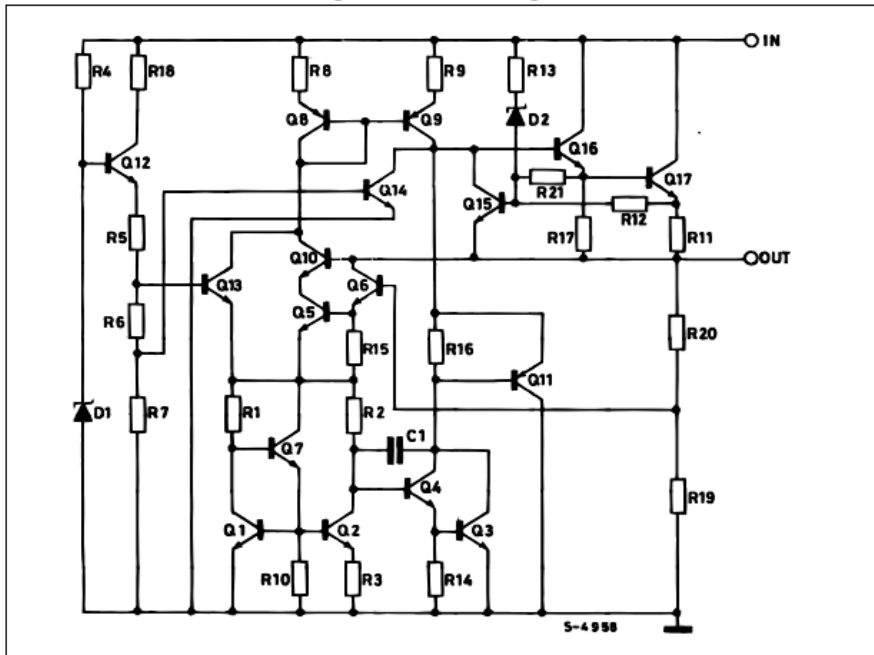


Table 2. Absolute maximum ratings

Symbol	Parameter	Value	Unit
V_I	DC input voltage	for $V_O = 5$ to $18V$	35
		for $V_O = 24V$	40
I_O	Output current	Internally limited	
P_D	Power dissipation	Internally limited	
T_{STG}	Storage temperature range	-65 to 150	$^{\circ}C$
T_{OP}	Operating junction temperature range	0 to 150	$^{\circ}C$

Table 3. Thermal data

Symbol	Parameter	TO-220	Unit
R_{thJC}	Thermal resistance junction-case	5	$^{\circ}C/W$
R_{thJA}	Thermal resistance junction-ambient	50	$^{\circ}C/W$

15.4 Appendix D: L78S05CV Datasheet Excerpt

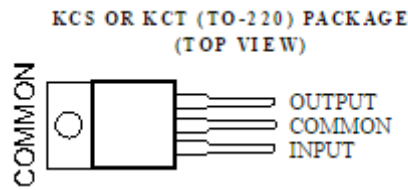
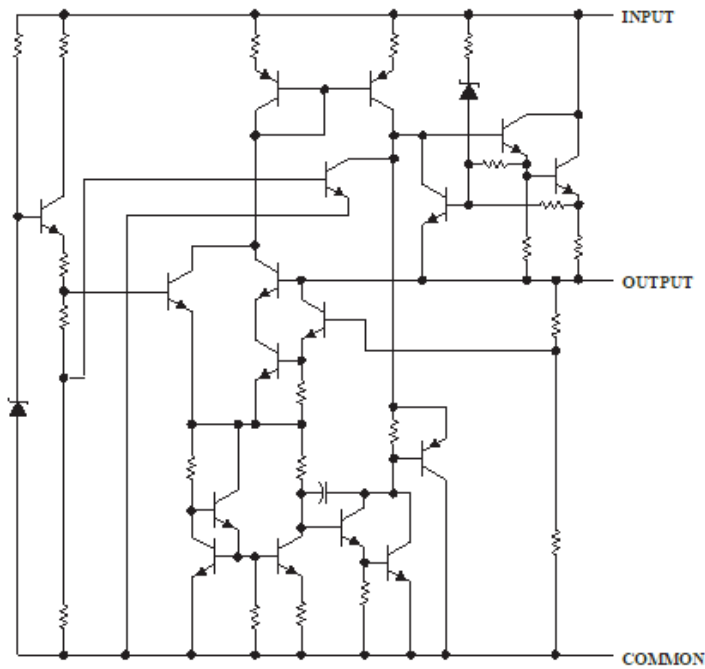


Figure 1. SCHEMATIC



over virtual junction temperature range (unless otherwise noted)

		MIN	MAX	UNIT
V _i	Input voltage		40	V
		µA7824C		
	All others		35	
T _j	Operating virtual junction temperature		150	°C
	Lead temperature		260	°C
				1.6 mm (1/16 in) from case for 10 s
T _{stg}	Storage temperature range	-65	150	°C

Package Thermal Data ⁽¹⁾

PACKAGE	BOARD	θ _{JA}	θ _{JC}	θ _{JP} ⁽²⁾
PowerFLEX (KTE) – OBSOLETE	High K, JESD 51-5	23°C/W	3°C/W	2.7°C/W
TO-220 (KCS), (KCT) (KC – OBSOLETE)	High K, JESD 51-5	19°C/W	17°C/W	3°C/W
TO-263 (KTT)	High K, JESD 51-5	25.3°C/W	18°C/W	1.94°C/W

15.5 Appendix E: Data Sheet for XBee Series 2

RSSI PWM

The XBee module features an RSSI/PWM pin (pin 6) that, if enabled, will adjust the PWM output to indicate the signal strength of the last received packet. The P0 (P-zero) command is used to enable the RSSI pulse width modulation (PWM) output on the pin. If P0 is set to 1, the RSSI/PWM pin will output a pulse width modulated signal where the frequency is adjusted based on the received signal strength of the last packet. Otherwise, for all other P0 settings, the pin can be used for general purpose IO.

When a data packet is received, if P0 is set to enable the RSSI/PWM feature, the RSSI PWM output is adjusted based on the RSSI of the last packet. The RSSI/PWM output will be enabled for a time based on the RP command. Each time an RF packet is received, the RSSI/PWM output is adjusted based on the RSSI of the new packet, and the RSSI timer is reset. If the RSSI timer expires, the RSSI/PWM pin is driven low. RP is measured in 100ms units and defaults to a value of 40 (4 seconds).

The RSSI PWM runs at 12MHz and has 2400 total counts (200us period).

RSSI (in dBm) is converted to PWM counts using the following equation:

$$\text{PWM counts} = (41 * \text{RSSI_Unsigned}) - 5928$$

IO Examples

Example 1: Configure the following IO settings on the XBee.

Configure AD1/DIO1 as a digital input with pullup resistor enabled

Configure AD2/DIO2 as an analog input

Configure DIO4 as a digital output, driving high.

To configure AD1/DIO1 as an input, issue the ATD1 command with a parameter of 3 ("ATD13"). To enable pull-up resistors on the same pin, the PR command should be issued with bit 3 set (e.g. ATPR8, ATPR1FFF, etc.).

The ATD2 command should be issued with a parameter of 2 to enable the analog input ("ATD22"). Finally, DIO4 can be set as an output, driving high by issuing the ATD4 command with a parameter value of 5 ("ATD45").

After issuing these commands, changes must be applied before the module IO pins will be updated to the new states. The AC or CN commands can be issued to apply changes (e.g. ATAC).

Example 2: Calculate the PWM counts for a packet received with an RSSI of -84dBm.

$$\text{RSSI} = -84 = 0xAC = 172 \text{ decimal (unsigned)}$$

$$\text{PWM counts} = (41 * 172) - 5928$$

$$\text{PWM counts} = 1124$$

With a total of 2400 counts, this yields an ON time of $(1124 / 2400) = 46.8\%$

Example 3: Configure the RSSI/PWM pin to operate for 2 seconds after each received RF packet.

First, ensure the RSSI/PWM functionality is enabled by reading the P0 (P-zero) command. It should be set to 1 (default).

To configure the duration of the RSSI/PWM output, set the RP command. To achieve a 2 second PWM output, set RP to 0x14 (20 decimal, or 2 seconds) and apply changes (AC command).

After applying changes, all received RF data packets should set the RSSI timer for 2 seconds.

15.6 Appendix F: C Code for Multilateration

```

#include<SoftwareSerial.h>
#include<SPI.h>
//pin assignments
int rssiPin = 9;
//constants
const int numBeacons = 6;
const int numIterations = 20;
const int pathLoss = 0;
const int refRssi = 0;
const int refRad = 0;
//variable declarations
float beaconX[numBeacons];
float beaconY[numBeacons];
float intersectX[numBeacons*2];
float usedIntersectY[numBeacons*2];
float usedIntersectX[numBeacons];
float intersectY[numBeacons];
int curRssi, curRssiSum, intersectPoint;
float radii[numBeacons];
float curPosX[2];
float curPosY[2];
float posX, posY, curRssiAve;
String instruction;
void setup()
{
  pinMode(rssiPin, INPUT);
  Serial.begin(9600);
}
void loop()
{
  instruction = Serial1.read();
  if (instruction == "Get Pos")
  {
    doMultilatCalc();
  }
}
void findRssiRadii()
{
  curRssiAve = 0;
  for (int i=0; i < numBeacons; i++) //gather RSSI Data from each beacon
  {
    beaconX[i] = 0; //recieved x value
    beaconY[i] = 0; //recieved y value

    for (int j = 0; j < numIterations; j++)
    {
      //wait until signal is received
      if(Serial.available() >= 21)
      {
        if (Serial.read() == 0x7E)
        {
          for(int i=1; i < 19; i++)
            byte disgardByte = Serial.read();
          curRssi = pulseIn(rssiPin, LOW, 200);
        }
      }
      curRssi = 0; //get rssi from signal
      curRssiSum = curRssiSum + curRssi;
    }
    curRssiAve = curRssiSum/numIterations;
    radii[i] = refRad*exp((curRssiAve - refRssi)/(10*pathLoss));
  }
}
void doMultilatCalc()
{
  for (int i=0; i < numBeacons-1; i++)
  {
    for (int j=(i+1); j < numBeacons; j++)
    {
      if (i==0 && j==1 || i==3 && j==2)
      {
        float a = beaconX[i]; //assign centroids
        float b = beaconY[i];
        float c = beaconX[j];
        float d = beaconY[j];
        float p = radii[j]; //assign radii
      }
    }
  }
}

```

```

float r = radii[i];
intersectX[intersectPoint] = (sq(a)-sq(c)+sq(p)-sq(r))/(2*(a-c));
intersectY[intersectPoint] = 0.5*(2*b -2*sqrt(((a*(sq(a)-sq(c)+sq(p)-
sq(r)))/(a-c))-((sq(sq(a)-sq(c)+sq(p)-
sq(r)))/(4*(sq(a-c))))-sq(a)+sq(r)));
intersectX[intersectPoint+1] = (sq(a)-sq(c)+sq(p)-sq(r))/(2*(a-c));
intersectY[intersectPoint+1] = 0.5*(2*b2*sqrt(((a*(sq(a)-sq(c)+sq(p)-
sq(r)))/(a-c))-((sq(sq(a)-sq(c)+sq(p)-
sq(r)))/(4*(sq(a-c))))-sq(a)+sq(r)));
intersectPoint = intersectPoint + 2;
//double iterate the intersection number (2 are added each time)
}
if (i==0 && j==3 || i==1 && j==2)
{
float a = beaconX[i]; //assign centroids
float b = beaconY[i];
float c = beaconX[j];
float d = beaconY[j];
float p = radii[j]; //assign radii
float r = radii[i];
intersectX[intersectPoint] = (1/(2*sq(b-d)))*((2*a*sq(b))-(4*a*b*d)+
(2*a*sq(d))-sqrt(-(sq(b-d))*((b-d-p-
r)*(b-d+p-r)*(b-d-p+r)*(b-d+p+r))));
intersectY[intersectPoint] = (sq(b)-sq(d)+sq(p)-sq(r))/(2*(b-d));
intersectX[intersectPoint+1] = (1/(2*sq(b-d)))*((2*a*sq(b))-(4*a*b*d)+
(2*a*sq(d))+sqrt(-(sq(b-d))*((b-d-p-
r)*(b-d+p-r)*(b-d-p+r)*(b-d+p+r))));
intersectY[intersectPoint+1] = (sq(b)-sq(d)+sq(p)-sq(r))/(2*(b-d));
intersectPoint = intersectPoint + 2;
}
}
}
weedOutNoise();
findAverage();
}
void weedOutNoise()
{
int numInter = 2*numBeacons;
sort(beaconX, numBeacons); //sorts the beacon positions to filter
sort(beaconY, numBeacons); //out unwanted intersections
int j = 0;
for (int i=0; i < numInter; i++)
//filters out so only intersections within the floorspace are used
{
if (intersectY[i] >= beaconY[0] && intersectY[i] <= beaconY[numBeacons])
{
if (intersectX[i] > beaconX[0] && intersectY[i] <= beaconX[numBeacons])
{
usedIntersectX[j] = intersectX[i];
usedIntersectY[j] = intersectY[i];
j++;
}
}
}
}
}
void findAverage()
{
float sumX = 0;
float sumY = 0;
for (int i=0; i < numBeacons; i++)
{
sumX = sumX + usedIntersectX[i];
sumY = sumY + usedIntersectY[i];
}
curPosX = sumX/numBeacons;
curPosY = sumY/numBeacons;
}

```

15.7 Appendix G: Matlab Code for Multilateration

```

clear
clc
pF = 0;
x = 5000;
Y = 5000;
x1 = 7071;
x2 = 7071;
x3 = 7071;
x4 = 7071;
off1 = -250;
off2 = 1250;
posX = zeros(1,100);
posY = zeros(1,100);
sumPosX = 0;
sumPosY = 0;
for y=1:100
r1 = (x1+off1) + ((x1+off2)-(x1+off1)).*rand();
r2 = (x2+off1) + ((x2+off2)-(x2+off1)).*rand();
r3 = (x3+off1) + ((x3+off2)-(x3+off1)).*rand();
r4 = (x4+off1) + ((x4+off2)-(x4+off1)).*rand();
acc = 600;
radii = [r1; r2; r3; r4];
beaconX = [0;10000;10000;0];
beaconY = [10000;10000;0;0];
intersectX =zeros(1,8);
intersectY =zeros(1,8);
usedIntersectX = zeros(1,4);
usedIntersectY = zeros(1,4);
    intersectPoint = 1;
    for i=1:4
        for j=1:4
            if i==1 && j==2 || i==4 && j==3
                a = beaconX(i);
                b = beaconY(i);
                c = beaconX(j);
                d = beaconY(j);
                p = radii(j);
                r = radii(i);
                intersectX(intersectPoint)=((a^2)-(c^2)+(p^2)-(r^2))/(2*(a-c));

```

```

intersectY(intersectPoint)=0.5*(2*b-2*sqrt(((a*((a^2)-(c^2)+
(p^2)-(r^2)))/(a-c))-(((a^2)-(c^2)+
(p^2)-(r^2))^2)/(4*((a-c)^2)))-
(a^2)+(r^2));
intersectX(intersectPoint+1)=((a^2)-(c^2)+(p^2)-(r^2))/(2*(a-
c));
intersectY(intersectPoint+1)=0.5*(2*b+2*sqrt(((a*((a^2)-
(c^2)+(p^2)-(r^2)))/(a-c))-
(((a^2)-(c^2)+(p^2)-(r^2))^2)
/(4*((a-c)^2)))- (a^2)+(r^2));

intersectPoint = intersectPoint + 2;
end
if i==1 && j==4 || i==2 && j==3
a = beaconX(i);
b = beaconY(i);
c = beaconX(j);
d = beaconY(j);
p = radii(j);
r = radii(i);
intersectX(intersectPoint)=(1/(2*((b-d)^2)))*((2*a*(b^2))-
(4*a*b*d)+(2*a*(d^2))-sqrt(-((b-
d)^2)*((b-d-p-r)*(b-d+p-r)*(b-d-
p+r)*(b-d+p+r))));
intersectY(intersectPoint)=((b^2)-(d^2)+(p^2)-(r^2))/(2*(b-
d));
intersectX(intersectPoint+1)=(1/(2*((b-d)^2)))*((2*a*(b^2))-
(4*a*b*d)+(2*a*(d^2))+sqrt(-((b-
d)^2)*((b-d-p-r)*(b-d+p-r)*(b-d-
p+r)*(b-d+p+r))));
intersectY(intersectPoint+1)=((b^2)-(d^2)+(p^2)-(r^2))/(2*(b-
d));

intersectPoint = intersectPoint + 2;
end
end
end
beaconX = sort(beaconX);
beaconY = sort(beaconY);
m = 1;
for i=1:8
if intersectY(i) >= beaconY(1) && intersectY(i) <= beaconY(4)
if intersectX(i) >= beaconX(1) && intersectX(i) <= beaconX(4)
usedIntersectX(m) = intersectX(i);
usedIntersectY(m) = intersectY(i);
m=m+1;
end
end
end
sumX = 0;

```

```
sumY = 0;
for i=1:4
    sumX = sumX + usedIntersectX(i);
    sumY = sumY + usedIntersectY(i);
end
curPosX = 0;
curPosY = 0;
curPosX = sumX/4;
curPosY = sumY/4;
posX(y) = curPosX;
posY(y) = curPosY;
sumPosX = sumPosX + curPosX;
sumPosY = sumPosY + curPosY;
if (curPosX>(X-acc)) && (curPosX<(X+acc)) && (curPosY>(Y-
    acc)) && (curPosY<(Y+acc))
    pF = pF+1;
end
end
avgPosX = sumPosX/100;
avgPosY = sumPosY/100;
pF
```

15.8 Appendix H: Structure of CRC Classes

Machine Class:

```
namespace CC_Ver1
{
    public class Machine
    {
        //public variables
        public const int noise = -1; //standard
        public const int unclass = 0; // standard
        public string machineName;
        public string machineType;
        public Controller controllerType;
        public int xDim;
        public int yDim;
        //set start values
        public string xPos = "0";
        public string yPos = "0";
        public int xPosInt = 0;
        public int yPosInt = 0;
        public int hardwareConfigNum = 0;
        public int softwareConfigNum = 0;
        public int ID = 0;

        public Machine(string name, string type, string controlType, int
            sizeX, int sizeY)
        {
            machineName = name;
            controllerType = new Controller(controlType);
            machineType = type;
            xDim = sizeX;
            yDim = sizeY;
        }

        public void ConvertToInt()
        {
            float xPosF = float.Parse(xPos);
            float yPosF = float.Parse(yPos);
            xPosInt = Convert.ToInt32(xPosF);
            yPosInt = Convert.ToInt32(yPosF);
        }

        public int DistanceSquared(Machine p1, Machine p2)
        {
            int diffX = p2.xPosInt - p1.xPosInt;
            int diffY = p2.yPosInt - p1.yPosInt;
            return diffX * diffX + diffY * diffY;
        }

        public string displayMach()
        {
            string tempStr;
            tempStr = machineType + "; " + hardwareConfigNum + "; " +
                xPosInt.ToString() + "; " + yPosInt.ToString() +
                "; " + ID.ToString();
            return tempStr;
        }

        public bool sameAs(Machine mach1)
        {
            if (mach1.machineType == this.machineType &&
                mach1.hardwareConfigNum == this.hardwareConfigNum)
            {
                return true;
            }
            else
                return false;
        }
    }
}
```

Cell Class:

```

namespace CC_Ver1
{
    public class Cell
    {
        public string cellName;
        public int numMach;
        public string cellDuty;
        public Machine[] machInCell;
        public int[] progSelect;
        bool flag;
        public int layNum;

        public Cell(string newName, int num, string duty, Machine[] machines)
        {
            cellName = newName;
            numMach = num;
            cellDuty = duty;
            machInCell = machines;
        }

        public void assignProgram()
        {
            for (int i = 0; i < machInCell.Length; i++)
            {
                machInCell[i].softwareConfigNum = progSelect[i];
            }
        }

        public bool sameAs(Cell cell1)
        {
            int check = 0;

            if (this.machInCell.Length == cell1.machInCell.Length)
            {
                for (int i = 0; i < this.machInCell.Length; i++)
                {
                    for (int j = 0; j < cell1.machInCell.Length; j++)
                    {
                        if (this.machInCell[i].sameAs(cell1.machInCell[j]))
                        {
                            check++;
                        }
                    }
                }
            }
            if (check == cell1.machInCell.Length)
                flag = true;
            else
                flag = false;

            return flag;
        }
    }
}

```

Layout Class:

```
namespace CC_Ver1
{
    public class Layout
    {
        public string layoutName;
        public string layoutDuty;
        public Cell[] cellsInLayout;
        bool flag;

        public Layout(string name, string duty, Cell[] cells)
        {
            layoutName = name;
            layoutDuty = duty;
            cellsInLayout = cells;
        }

        public bool sameAs(Layout layout1)
        {
            int check = 0;
            if (this.cellsInLayout.Length ==
                layout1.cellsInLayout.Length)//checks if same num of cells
            {
                for (int i = 0; i < this.cellsInLayout.Length; i++)
                {
                    for (int j = 0; j < layout1.cellsInLayout.Length; j++)
                    {
                        if (this.cellsInLayout[i].sameAs(
                            layout1.cellsInLayout[j]))
                            check++;
                    }
                }
            }
            if (check >= layout1.cellsInLayout.Length)
                flag = true;
            else
                flag = false;

            return flag;
        }
    }
}
```


15.9 Appendix I: C# Code for DBSCAN Experimentation

```

static private List<List<Machine>> getClust(List<Machine> machines, double
EPS, int minP)
{
    if (machines == null) return null;
    List<List<Machine>> clusters = new List<List<Machine>>();
    EPS = EPS*EPS; // square EPS
    int ID = 1;
    for (int i = 0; i < machines.Count; i++)
    {
        Machine M = machines[i];
        if (M.ID == Machine.unclass)
        {
            if (ClusterExp(machines, M, ID, EPS, minP))
                ID++;
        }
    }
    //stock bubble sort algorithm for points order
    int temp = 0;
    for (int write = 0; write < machines.Count; write++)
    {
        for (int sort = 0; sort < machines.Count - 1; sort++)
        {
            if (machines[sort].ID > machines[sort + 1].ID)
            {
                temp = machines[sort + 1].ID;
                machines[sort + 1].ID = machines[sort].ID;
                machines[sort].ID = temp;
            }
        }
    }
    int maxID = machines[machines.Count - 1].ID;
    if (maxID < 1) return clusters; // list is empty, no clusters!
    for (int i = 0; i < maxID; i++)
        clusters.Add(new List<Machine>());
    foreach (Machine M in machines)
        if (M.ID > 0) clusters[M.ID - 1].Add(M);
    }
    return clusters;
}

static List<Machine> RegionSelect(List<Machine> machines, Machine M, double
EPS)

```

```

{
    Machine Ex = new Machine("temp", "temp", "temp", 10, 10);

    List<Machine> region = new List<Machine>();
    for (int i = 0; i < machines.Count; i++)
    {
        int distSquared = Ex.DistanceSquared(M, machines[i]);
        if (distSquared <= EPS)
            region.Add(machines[i]);
    }
    return region;
}

static bool ClusterExp(List<Machine> machines, Machine M, int
                        ID, double EPS, int minP)
{
    List<Machine> seed = RegionSelect(machines, M, EPS);
    if (seed.Count < minP) // no core point
    {
        M.ID = Machine.noise;
        return false;
    }
    else // all these points are reachable from Machine M
    {
        for (int i = 0; i < seed.Count; i++)
            seed[i].ID = ID;

        seed.Remove(M);
        while (seed.Count > 0)
        {
            Machine currentM = seed[0];
            List<Machine> result = RegionSelect(machines, currentM, EPS);
            if (result.Count >= minP)
            {
                for (int i = 0; i < result.Count; i++)
                {
                    Machine resultM = result[i];
                    if (resultM.ID==Machine.unclass||resultM.ID==Machine.noise)
                    {
                        if(resultM.ID==Machine.unclass)
                            seed.Add(resultM);
                        resultM.ID = ID;
                    }
                }
            }
        }
    }
}

```

```

        }
    }
    seed.Remove(currentM);
}
return true;
}
}

```

15.10 Appendix J: C# Code for CRC Inference Engine

Inference engine:

```

private void runKBS_Click(object sender, EventArgs e)
{
    flag = false;
    //needs to compare CURRENT LAYOUT the data base
    foreach (Layout someState in Program.knowledgeBase.baseOfKnowledge)
    {
        Layout currentLay = scanWindow.curLayout; //temp layout
        if (currentLay.sameAs(someState)) //find a matching layout
        {
            flag = true;
            forAssignment = someState;
        }
    }
    if (flag == true)
        statusKBS.Text = "Match Found!\nContinue\nwith Assignment.";
    else if (flag == false)
        statusKBS.Text = "Failed.\nDo NOT Continue\nwith Assignment!";
}

```

From the Layout Class, called by the KBS inference engine:

```

public bool sameAs(Layout layout1)
{
    int check = 0;
    if (this.cellsInLayout.Length == layout1.cellsInLayout.Length)
        //checks if same num of cells
    {
        for (int i = 0; i < this.cellsInLayout.Length; i++)
        {
            for (int j = 0; j < layout1.cellsInLayout.Length; j++)
            {
                //checks if each machine in cell is the same
                if(this.cellsInLayout[i].sameAs(
                    layout1.cellsInLayout[j]))
                    check++;
            }
        }
        if (check >= layout1.cellsInLayout.Length)
            flag = true;
        else
            flag = false;
    }
    return flag;
}

```

From the Cell Class, called by above:

```

public bool sameAs(Cell cell1)
{
    int check = 0;
    if (this.machInCell.Length == cell1.machInCell.Length)

```

```
{
    for (int i = 0; i < this.machInCell.Length; i++)
    {
        for (int j = 0; j < cell1.machInCell.Length; j++)
        {
            if (this.machInCell[i].sameAs(cell1.machInCell[j]))
                check++;
        }
    }
    if (check == cell1.machInCell.Length)
        flag = true;
    else
        flag = false;
    return flag;
}
```

From the Machine Class, called by above:

```
public bool sameAs(Machine mach1)
{
    if (mach1.machineType == this.machineType &&
        mach1.hardwareConfigNum == this.hardwareConfigNum)
    {
        return true;
    }
    else
        return false;
}
```

15.11 Appendix K: Raw Data for 10.3

Distance (m)	Recording #	Experiment 1		Experiment 2		Experiment 3		Experiment 4	
		Hex Read (dBm)	Dec Value (dBm)	Hex Read (dBm)	Dec Value (dBm)	Hex Read (dBm)	Dec Value (dBm)	Hex Read (dBm)	Dec Value (dBm)
1	1	2D	45	2E	46	2A	42	2E	46
	2	2D	45	2C	44	2B	43	2F	47
	3	2E	46	2F	47	2B	43	2C	44
	4	2A	42	2F	47	2C	44	2D	45
	5	2C	44	2E	46	2B	43	2D	45
	σ (Dec dBm)	1.356		1.095		0.632		1.020	
	Ave (Dec dBm)	44.4		46.0		43.0		45.4	
1.5	1	2F	47	31	49	2E	46	30	48
	2	30	48	32	50	2F	47	2F	47
	3	2D	45	34	52	2F	47	2F	47
	4	2F	47	31	49	2E	46	2E	46
	5	2F	47	30	48	2E	46	2F	47
	σ (Dec dBm)	0.980		1.356		0.490		0.632	
	Ave (Dec dBm)	46.8		49.6		46.4		47.0	
2	1	30	48	30	48	30	48	31	49
	2	31	49	31	49	30	48	33	51
	3	30	48	31	49	31	49	32	50
	4	33	51	30	48	32	50	31	49
	5	32	50	32	50	31	49	32	50
	σ (Dec dBm)	1.166		0.748		0.748		0.748	
	Ave (Dec dBm)	49.2		48.8		48.8		49.8	
2.5	1	35	53	35	53	31	49	31	49
	2	38	56	32	50	34	52	2F	47
	3	36	54	33	51	34	52	33	51
	4	36	54	32	50	33	51	33	51
	5	38	56	34	52	32	50	31	49
	σ (Dec dBm)	1.200		1.166		1.166		1.497	
	Ave (Dec dBm)	54.6		51.2		50.8		49.4	
3	1	39	57	37	55	36	54	34	52
	2	36	54	36	54	35	53	35	53
	3	39	57	34	52	34	52	33	51
	4	37	55	36	54	34	52	34	52
	5	35	53	37	55	33	51	32	50
	σ (Dec dBm)	1.600		1.095		1.020		1.020	
	Ave (Dec dBm)	55.2		54.0		52.4		51.6	
3.5	1	38	56	3A	58	32	50	3B	59
	2	37	55	3B	59	36	54	3B	59
	3	36	54	38	56	35	53	3A	58
	4	39	57	38	56	35	53	3B	59
	5	3A	58	39	57	36	54	39	57
	σ (Dec dBm)	1.414		1.166		1.470		0.800	
	Ave (Dec dBm)	56.0		57.2		52.8		58.4	
4	1	38	56	3A	58	3A	58	3C	60
	2	39	57	3C	60	3A	58	3C	60
	3	39	57	3B	59	39	57	3A	58
	4	3A	58	3A	58	39	57	3A	58
	5	3A	58	3C	60	39	59	39	57
	σ (Dec dBm)	0.748		0.894		0.748		1.200	
	Ave (Dec dBm)	57.2		59.0		57.8		58.6	

4.5	1	3A	58	3D	61	39	57	33	51
	2	3B	59	3D	61	39	57	36	54
	3	38	56	3C	60	3A	58	37	55
	4	38	56	3A	58	3B	59	38	56
	5	3B	59	3D	61	3B	59	36	54
	σ (Dec dBm)	1.356		1.166		0.894		1.673	
	Ave (Dec dBm)	57.6		60.2		58.0		54.0	
5	1	3A	58	3E	62	3C	60	3C	60
	2	3F	63	3D	61	3B	59	3B	59
	3	3D	61	3E	62	3C	60	3A	58
	4	3E	62	3D	61	3A	58	39	57
	5	3E	62	3F	63	3A	58	3B	59
	σ (Dec dBm)	1.720		0.748		0.894		1.020	
	Ave (Dec dBm)	61.2		61.8		59.0		58.6	
5.5	1	38	56	39	57	3A	58	39	57
	2	38	56	3A	58	3B	59	39	57
	3	3A	58	3A	58	39	57	38	56
	4	37	55	3B	59	39	57	3A	58
	5	37	55	3B	59	3B	59	3A	58
	σ (Dec dBm)	1.095		0.748		0.894		0.748	
	Ave (Dec dBm)	56.0		58.2		58.0		57.2	
6	1	3A	58	38	56	36	54	37	55
	2	3A	58	35	53	37	55	39	57
	3	39	57	36	54	37	55	36	54
	4	39	57	38	56	38	56	37	55
	5	39	57	38	56	3B	59	3A	58
	σ (Dec dBm)	0.490		1.265		1.720		1.470	
	Ave (Dec dBm)	57.4		55.0		55.8		55.8	
6.5	1	3C	60	3F	63	3A	58	3D	61
	2	3C	60	3E	62	3A	58	3A	58
	3	3B	59	3D	61	39	57	3A	58
	4	3B	59	3D	61	37	55	3C	60
	5	3B	59	3D	61	37	55	3A	58
	σ (Dec dBm)	0.490		0.800		1.356		1.265	
	Ave (Dec dBm)	59.4		61.6		56.6		59.0	
7	1	36	54	3F	63	38	56	36	54
	2	38	56	3F	63	37	55	38	56
	3	3A	58	3D	61	39	57	34	52
	4	37	55	3E	62	3A	58	37	55
	5	37	55	3E	62	3A	58	36	54
	σ (Dec dBm)	1.356		0.748		1.166		1.327	
	Ave (Dec dBm)	55.6		62.2		56.8		54.2	
7.5	1	39	57	41	65	39	57	3E	62
	2	3A	58	41	65	3A	58	3E	62
	3	3A	58	3F	63	3B	59	3D	61
	4	3B	59	3E	62	3A	58	3D	61
	5	39	57	3F	63	39	57	3D	61
	σ (Dec dBm)	0.748		1.200		0.748		0.490	
	Ave (Dec dBm)	57.8		63.6		57.8		61.4	
8	1	39	57	3E	62	37	55	3A	58
	2	3C	60	3D	61	36	54	39	57
	3	3A	58	3A	58	36	54	3C	60
	4	3B	59	3A	58	39	57	3C	60
	5	3A	58	3B	59	3A		3B	59
	σ (Dec dBm)	1.020		1.625		1.225		1.166	
	Ave (Dec dBm)	58.4		59.6		44.0		58.8	

8.5	1	3E	62	42	66	3C	60	39	57
	2	3E	62	41	65	3A	58	3A	58
	3	3F	63	41	65	3B	59	3A	58
	4	3E	62	3D	61	3A	58	38	56
	5	3D	61	3F	63	3B	59	39	57
	σ (Dec dBm)	0.632		1.789		0.748		0.748	
Ave (Dec dBm)	62.0		64.0		58.8		57.2		
9	1	3A	58	3D	61	3A	58	3A	58
	2	3C	60	3A	58	3C	60	3B	59
	3	3A	58	3D	61	3D	61	3B	59
	4	3B	59	3E	62	3D	61	3C	60
	5	3A	58	3C	60	3C	60	39	57
	σ (Dec dBm)	0.800		1.356		1.095		1.020	
Ave (Dec dBm)	58.6		60.4		60.0		58.6		
9.5	1	3C	60	3C	60	3D	61	3C	60
	2	3E	62	3D	61	3C	60	3C	60
	3	3D	61	3D	61	3B	59	3A	58
	4	3C	60	3E	62	3C	60	3B	59
	5	3C	60	3E	62	3C	60	3B	59
	σ (Dec dBm)	0.800		0.748		0.632		0.748	
Ave (Dec dBm)	60.6		61.2		60.0		59.2		
10	1	3F	63	3C	60	3E	62	3E	62
	2	3F	63	3E	62	3F	63	3D	61
	3	40	64	3E	62	3C	60	3C	60
	4	3D	61	3D	61	3D	61	3D	61
	5	3F	63	3E	62	3D	61	3C	60
	σ (Dec dBm)	0.980		0.800		1.020		0.748	
Ave (Dec dBm)	62.8		61.4		61.4		60.8		
10.5	1	41	65	40	64	3E	62	3F	63
	2	43	67	3F	63	3E	62	3E	62
	3	3F	63	3F	63	40	64	40	64
	4	3F	63	40	64	3E	62	41	65
	5	40	64	40	64	40	64	3F	63
	σ (Dec dBm)	1.497		0.490		0.980		1.020	
Ave (Dec dBm)	64.4		63.6		62.8		63.4		
11	1	42	66	40	64	3F	63	3F	63
	2	41	65	3F	63	41	65	3E	62
	3	43	67	41	65	3F	63	3E	62
	4	3F	63	3F	63	40	64	40	64
	5	3F	63	40	64	3F	63	3F	63
	σ (Dec dBm)	1.600		0.748		0.800		0.748	
Ave (Dec dBm)	64.8		63.8		63.6		62.8		
11.5	1	41	65	42	66	40	64	40	64
	2	43	67	40	64	42	66	3F	63
	3	41	65	41	65	3F	63	42	66
	4	43	67	42	66	40	64	42	66
	5	3F	63	40	64	40	64	42	66
	σ (Dec dBm)	1.497		0.894		0.980		1.265	
Ave (Dec dBm)	65.4		65.0		64.2		65.0		
12	1	41	65	43	67	42	66	40	64
	2	42	66	41	65	41	65	3F	63
	3	43	67	42	66	42	66	41	65
	4	42	66	41	65	41	65	42	66
	5	42	66	40	64	41	65	3F	63
	σ (Dec dBm)	0.632		1.020		0.490		1.166	
Ave (Dec dBm)	66.0		65.4		65.4		64.2		

15.12 Appendix L: Calculations for 10.3

Conversion from Natural Logarithmic ($\ln(\cdot)$) values given by Excel to the standard log-base-10 form ($\log_{10}(\cdot)$) shown in the literature:

$$p_j = p_0 + 10(n)\log\left(\frac{r_j}{r_0}\right) \quad (\text{given in the literature})$$

$$y = 7.42\ln(x) + 45.15 \quad (\text{generated from Excel})$$

$$\text{now, } \ln(x) = \frac{\log(x)}{\log(e)}$$

$$\Rightarrow y = 7.42 \frac{\log(x)}{\log(e)} + 45.15$$

$$= 7.42 \frac{\log(x)}{0.4343} + 45.15$$

$$= 17.08\log(x) + 45.15$$

$$= 45.15 + 10(1.71)\log\left(\frac{x}{1}\right) \quad (\text{to match the literature form})$$

$$\Rightarrow p_0 = 45.15; n = 1.71; r_j = x$$

15.13 Appendix M: Raw Data for 10.4

Distance (m)	Recording #	Experiment 1 PulseIn Value (ms)	Experiment 2 PulseIn Value (ms)	Experiment 3 PulseIn Value (ms)
1	1	2	5	4
	2	2	4	4
	3	4	7	2
	4	2	2	2
	5	2	2	2
	6	2	2	4
	7	4	2	2
	8	2	2	4
	9	2	2	2
	10	2	4	2
	σ		0,80	1,66
Ave		2,4	3,2	2,8
1,5	1	2	5	4
	2	4	6	2
	3	6	4	6
	4	4	5	7
	5	7	9	4
	6	7	5	6
	7	4	5	7
	8	7	7	4
	9	2	7	7
	10	4	6	2
	σ		1,85	1,37
Ave		4,7	5,9	4,9
2	1	12	4	7
	2	7	7	6
	3	7	12	12
	4	7	9	14
	5	15	7	6
	6	15	7	15
	7	7	8	7
	8	9	12	9
	9	7	9	12
	10	10	9	9
	σ		3,14	2,29
Ave		9,6	8,4	9,7
2,5	1	20	12	22
	2	20	11	18
	3	20	12	15
	4	25	15	18
	5	23	15	23
	6	23	16	21
	7	23	12	20
	8	20	9	18
	9	23	10	16
	10	18	14	23
	σ		2,06	2,20
Ave		21,5	12,6	19,4
3	1	27	22	23
	2	23	23	24
	3	23	18	16
	4	28	16	18
	5	14	12	24
	6	20	16	18
	7	28	12	26
	8	30	16	26
	9	23	16	22
	10	28	21	25
	σ		4,59	3,63
Ave		24,4	17,2	22,2

3,5	1	33	26	36
	2	28	24	24
	3	33	24	36
	4	28	24	28
	5	36	21	24
	6	28	28	34
	7	28	28	28
	8	30	26	31
	9	28	25	21
	10	32	23	36
	σ		2,76	2,07
Ave		30,4	24,9	29,8
4	1	33	32	24
	2	36	28	28
	3	36	28	36
	4	38	26	36
	5	31	31	34
	6	31	28	29
	7	32	30	24
	8	33	28	38
	9	36	28	34
	10	38	34	28
	σ		2,58	2,28
Ave		34,4	29,3	31,1
4,5	1	41	38	32
	2	33	32	42
	3	33	35	42
	4	41	33	36
	5	41	33	41
	6	44	38	36
	7	33	31	38
	8	38	28	40
	9	44	28	28
	10	46	31	32
	σ		4,67	3,35
Ave		39,4	32,7	36,7
5	1	40	45	38
	2	44	36	38
	3	44	41	45
	4	49	38	44
	5	36	34	41
	6	36	40	38
	7	41	36	36
	8	44	38	36
	9	38	36	41
	10	46	38	44
	σ		4,12	2,99
Ave		41,8	38,2	40,1
5,5	1	48	58	42
	2	46	48	46
	3	52	48	44
	4	54	42	51
	5	50	41	47
	6	59	42	54
	7	51	42	48
	8	51	47	48
	9	50	47	52
	10	46	49	44
	σ		3,66	4,84
Ave		50,7	46,4	47,6
6	1	51	51	47
	2	51	48	52
	3	57	52	52
	4	44	45	48
	5	54	51	54
	6	51	58	58
	7	52	48	52
	8	51	44	42
	9	51	46	44
	10	49	47	52
	σ		3,14	3,92
Ave		51,1	49,0	50,1

6,5	1	54	53	48
	2	59	48	54
	3	48	56	58
	4	54	54	47
	5	62	49	48
	6	51	52	56
	7	60	49	56
	8	57	52	51
	9	51	52	84
	10	52	47	48
	σ	4,31	2,71	10,39
Ave	54,8	51,2	55,0	
7	1	60	63	58
	2	65	52	48
	3	65	44	62
	4	67	51	66
	5	52	58	68
	6	48	61	58
	7	59	58	54
	8	65	51	52
	9	65	61	57
	10	59	58	56
	σ	5,97	5,66	5,80
Ave	60,5	55,7	57,9	
7,5	1	68	60	64
	2	59	58	72
	3	72	57	58
	4	54	61	72
	5	75	56	64
	6	75	54	57
	7	52	50	67
	8	62	58	68
	9	50	64	52
	10	67	61	54
	σ	8,92	3,78	6,84
Ave	63,4	57,9	62,8	
8	1	80	54	72
	2	62	62	69
	3	54	65	71
	4	62	61	59
	5	67	54	62
	6	75	58	68
	7	62	58	68
	8	67	58	64
	9	75	62	70
	10	71	64	58
	σ	7,45	3,64	4,76
Ave	67,5	59,6	66,1	
8,5	1	67	62	67
	2	59	68	74
	3	70	68	81
	4	68	62	68
	5	80	58	74
	6	75	68	67
	7	67	58	64
	8	81	64	68
	9	70	69	71
	10	83	65	64
	σ	7,20	3,92	5,02
Ave	72,7	64,2	69,8	
9	1	80	69	76
	2	86	69	80
	3	85	72	84
	4	88	65	72
	5	85	71	80
	6	78	72	85
	7	76	63	68
	8	85	64	76
	9	80	58	84
	10	80	68	81
	σ	3,77	4,30	5,28
Ave	82,3	67,1	78,6	

9,5	1	86	81	78
	2	88	76	84
	3	82	72	91
	4	78	68	94
	5	90	73	79
	6	79	72	72
	7	76	68	82
	8	85	69	98
	9	86	72	74
	10	81	78	81
	σ	4,37	4,09	8,11
Ave	83,1	72,9	83,3	
10	1	92	70	93
	2	101	75	81
	3	85	80	78
	4	82	72	76
	5	100	67	101
	6	110	78	94
	7	90	81	98
	8	94	72	96
	9	78	78	91
	10	76	78	81
	σ	10,31	4,41	8,58
Ave	90,8	75,1	88,9	
10,5	1	110	82	94
	2	101	90	102
	3	86	78	101
	4	84	78	95
	5	89	90	86
	6	106	86	84
	7	108	86	92
	8	92	84	81
	9	79	73	94
	10	91	73	101
	σ	10,34	5,98	7,00
Ave	94,6	82,0	93,0	
11	1	110	89	88
	2	86	85	101
	3	94	84	94
	4	98	78	106
	5	106	76	110
	6	108	76	92
	7	109	90	101
	8	89	84	95
	9	99	86	88
	10	88	84	106
	σ	8,75	4,73	7,42
Ave	98,7	83,2	98,1	
11,5	1	108	87	109
	2	110	92	112
	3	94	94	95
	4	96	101	97
	5	116	86	110
	6	88	84	88
	7	94	90	92
	8	96	84	98
	9	109	86	110
	10	118	87	101
	σ	9,94	5,05	8,11
Ave	102,9	89,1	101,2	
12	1	121	84	104
	2	108	87	98
	3	107	93	108
	4	95	102	113
	5	109	102	92
	6	98	86	110
	7	120	85	98
	8	108	87	106
	9	109	91	112
	10	112	92	96
	σ	7,72	6,24	6,93
Ave	108,7	90,9	103,7	

15.14 Appendix N: Raw Data for 10.8

3500; 5119; 0; 12835; 32046
3600; 9219; 0; 14219; 26562
3700; 14474; 0; 14300; 21226
3800; 20766; 0; 13031; 16203
3900; 27329; 0; 10876; 11795
4000; 33500; 0; 8435; 8065
4100; 38469; 0; 6188; 5343
4200; 42211; 0; 4411; 3378
4300; 45003; 0; 3031; 1966
4400; 47140; 0; 1879; 981
4500; 48521; 0; 1088; 391
4600; 49348; 0; 580; 72
4700; 49718; 0; 276; 6
4800; 49906; 0; 94; 0
4900; 49962; 0; 38; 0
5000; 49983; 0; 17; 0
5100; 49998; 0; 2; 0
5200; 50000; 0; 0; 0
5300; 50000; 0; 0; 0
5400; 50000; 0; 0; 0
5500; 50000; 0; 0; 0
5600; 50000; 0; 0; 0
5700; 50000; 0; 0; 0
5800; 50000; 0; 0; 0
5900; 49996; 0; 4; 0
6000; 49982; 0; 18; 0

Layout 1: Random

3500; 4441; 0; 12504; 33055
3600; 7903; 0; 14168; 27929
3700; 12667; 0; 14534; 22799
3800; 18051; 0; 13863; 18086
3900; 23805; 0; 12300; 13895
4000; 29381; 0; 10243; 10376
4100; 34227; 0; 8138; 7635
4200; 38411; 0; 6117; 5472
4300; 41799; 0; 4387; 3814
4400; 44233; 0; 3109; 2658
4500; 46049; 0; 2142; 1809
4600; 47387; 0; 1414; 1199
4700; 48277; 0; 911; 812
4800; 48909; 0; 582; 509
4900; 49303; 0; 361; 336
5000; 49573; 0; 210; 217
5100; 49753; 0; 115; 132
5200; 49846; 0; 71; 83
5300; 49909; 0; 41; 50
5400; 49940; 0; 27; 33
5500; 49956; 0; 22; 22
5600; 49969; 0; 23; 8
5700; 49964; 0; 31; 5
5800; 49948; 0; 51; 1
5900; 49905; 0; 94; 1
6000; 49855; 0; 145; 0

Layout 1: Gaussian

3500; 35647; 0; 2302; 12051
3600; 41754; 0; 908; 7338
3700; 45816; 0; 109; 4075
3800; 47799; 0; 8; 2193
3900; 48954; 0; 0; 1046
4000; 49586; 0; 0; 414
4100; 49885; 0; 0; 115
4200; 49988; 0; 0; 12
4300; 50000; 0; 0; 0
4400; 49962; 0; 38; 0
4500; 49717; 0; 283; 0
4600; 49031; 0; 969; 0
4700; 47923; 0; 2077; 0
4800; 46096; 0; 3904; 0
4900; 43639; 0; 6361; 0
5000; 40467; 0; 9533; 0
5100; 36612; 0; 13388; 0
5200; 32325; 0; 17675; 0
5300; 27553; 0; 22447; 0
5400; 22582; 0; 27418; 0
5500; 17541; 0; 32459; 0
5600; 12764; 0; 37236; 0
5700; 8633; 0; 41367; 0
5800; 5410; 0; 44590; 0
5900; 3066; 0; 46934; 0
6000; 1613; 0; 48387; 0

Layout 2: Random

3500; 34353; 0; 3098; 12549
3600; 39616; 0; 2084; 8300
3700; 43574; 0; 1281; 5145
3800; 46201; 0; 774; 3025
3900; 47822; 0; 456; 1722
4000; 48790; 0; 277; 933
4100; 49299; 1; 212; 488
4200; 49562; 0; 206; 232
4300; 49570; 0; 332; 98
4400; 49367; 0; 590; 43
4500; 48987; 0; 998; 15
4600; 48315; 0; 1678; 7
4700; 47262; 0; 2735; 3
4800; 45781; 0; 4218; 1
4900; 43727; 0; 6273; 0
5000; 40964; 0; 9036; 0
5100; 37415; 0; 12585; 0
5200; 33126; 0; 16874; 0
5300; 28064; 0; 21936; 0
5400; 22931; 0; 27069; 0
5500; 17760; 0; 32240; 0
5600; 12793; 0; 37207; 0
5700; 8608; 0; 41392; 0
5800; 5302; 0; 44698; 0
5900; 2980; 0; 47020; 0
6000; 1547; 0; 48453; 0

Layout 2: Gaussian

3500; 10727; 0; 11252; 28021	3500; 9897; 0; 11532; 28571
3600; 15859; 0; 10481; 23660	3600; 14912; 0; 10773; 24315
3700; 21382; 0; 8881; 19737	3700; 20352; 0; 9318; 20330
3800; 26953; 0; 7010; 16037	3800; 25709; 0; 7731; 16560
3900; 32132; 0; 5208; 12660	3900; 30624; 1; 5899; 13476
4000; 36711; 0; 3553; 9736	4000; 34851; 1; 4491; 10657
4100; 40599; 0; 2212; 7189	4100; 38501; 1; 3217; 8281
4200; 43602; 0; 1299; 5099	4200; 41559; 1; 2209; 6231
4300; 45964; 0; 715; 3321	4300; 43804; 2; 1530; 4664
4400; 47696; 0; 317; 1987	4400; 45575; 5; 1034; 3386
4500; 48843; 0; 93; 1064	4500; 46850; 7; 704; 2439
4600; 49504; 0; 18; 478	4600; 47780; 4; 517; 1699
4700; 49794; 0; 1; 205	4700; 48391; 3; 429; 1177
4800; 49912; 0; 0; 88	4800; 48795; 3; 407; 795
4900; 49972; 0; 0; 28	4900; 48967; 2; 490; 541
5000; 49945; 0; 48; 7	5000; 48940; 1; 681; 378
5100; 49743; 0; 256; 1	5100; 48824; 0; 938; 238
5200; 49366; 0; 634; 0	5200; 48558; 0; 1296; 146
5300; 48815; 0; 1185; 0	5300; 48141; 0; 1767; 92
5400; 48004; 0; 1996; 0	5400; 47497; 0; 2447; 56
5500; 47063; 0; 2937; 0	5500; 46702; 0; 3266; 32
5600; 45918; 0; 4082; 0	5600; 45645; 0; 4339; 16
5700; 44576; 0; 5424; 0	5700; 44393; 0; 5598; 9
5800; 43026; 0; 6974; 0	5800; 42879; 0; 7113; 8
5900; 41248; 0; 8752; 0	5900; 41225; 0; 8768; 7
6000; 39303; 0; 10697; 0	6000; 39181; 0; 10814; 5

Layout 3: Random

Layout 3: Gaussian

3500;	17558;	0;	4933;	27509
3600;	23784;	2;	4153;	22061
3700;	29786;	2;	3199;	17013
3800;	35470;	4;	2189;	12337
3900;	40312;	16;	1288;	8384
4000;	43974;	20;	618;	5388
4100;	46453;	27;	362;	3158
4200;	47726;	32;	444;	1798
4300;	48326;	35;	735;	904
4400;	48444;	25;	1153;	378
4500;	48076;	9;	1786;	129
4600;	47211;	1;	2756;	32
4700;	45820;	1;	4176;	3
4800;	43984;	0;	6016;	0
4900;	41693;	0;	8307;	0
5000;	38934;	0;	11066;	0
5100;	35556;	0;	14444;	0
5200;	31892;	0;	18108;	0
5300;	27857;	0;	22143;	0
5400;	23842;	0;	26158;	0
5500;	19819;	0;	30181;	0
5600;	15980;	0;	34020;	0
5700;	12373;	0;	37627;	0
5800;	9221;	0;	40779;	0
5900;	6521;	0;	43479;	0
6000;	4422;	0;	45578;	0

Layout 4: Random

3500;	16110;	7;	6423;	27460
3600;	21980;	13;	5827;	22180
3700;	27515;	26;	5070;	17389
3800;	32507;	46;	4139;	13308
3900;	36829;	61;	3164;	9946
4000;	40382;	75;	2406;	7137
4100;	43091;	109;	1849;	4951
4200;	44955;	125;	1550;	3370
4300;	46170;	132;	1518;	2180
4400;	46742;	131;	1724;	1403
4500;	46772;	122;	2252;	854
4600;	46259;	98;	3136;	507
4700;	45274;	68;	4377;	281
4800;	43718;	57;	6068;	157
4900;	41725;	41;	8152;	82
5000;	39174;	22;	10752;	52
5100;	36122;	16;	13838;	24
5200;	32516;	9;	17461;	14
5300;	28604;	7;	21383;	6
5400;	24342;	6;	25648;	4
5500;	20002;	4;	29993;	1
5600;	15841;	3;	34156;	0
5700;	12091;	1;	37908;	0
5800;	8695;	0;	41305;	0
5900;	5952;	0;	44048;	0
6000;	3893;	0;	46107;	0

Layout 4: Gaussian

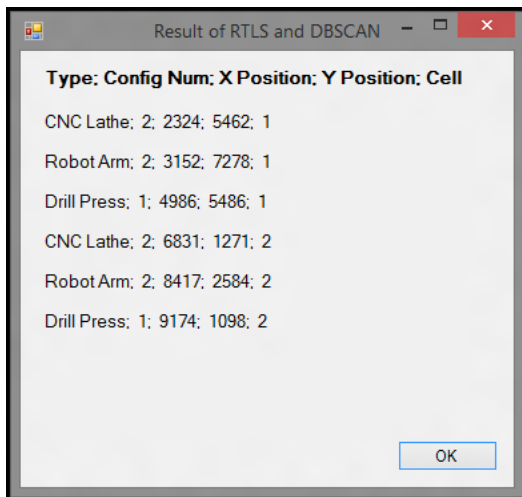
15.15 Appendix O: Knowledge Base for KBS Test

Type	Hardware Config	Cell No.	Layout No.		
two	1	1	1		
six	2				
nine	5				
twelve	1	2			
six	6				
six	6				
six	4	3			
one	8				
one	1				
five	3	4			
seven	5				
nine	4				
one	2	1	2		
four	3				
nine	1				
eleven	5				
six	1			2	
one	3				
six	4				
one	5				
four	6				
five	4				
six	1			3	
seven	1				
nine	2				
eight	6				
two	2	1	3		
three	4				
nine	2				
ten	8				
twelve	9				
nine	4			2	
ten	6				
eleven	3				
twelve	7				
two	1				
six	2				
nine	5			1	4
twelve	2				
six	3				
six	4				
six	5				
four	6	2			
one	4				
five	1				
seven	1				
nine	2				
eight	6		3		
two	1				
three	2				
four	5				
nine	3				
ten	4				
one	6	1	5		
three	8				
five	4				
eight	1				
one	6			2	
three	8				
five	4				
eight	1				
one	6				3
three	8				
five	4				
eight	1				
twelve	2	1	6		
nine	5				
ten	3				
eleven	1				
twelve	2			2	
five	1				
six	2				
twelve	2				
nine	5				
ten	3				
eleven	1			2	
twelve	2				
five	1				
six	2				
four	8	1	7		
six	6				
seven	6				
nine	5				
eleven	4				
ten	9			2	
twelve	6				
eight	2				
four	4			3	
two	1				
three	2				
four	8			4	
six	6				
four	6				
one	5				
eight	5	1	8		
two	6				
three	2			2	
nine	1				
nine	2				
twelve	2			3	
nine	3				
ten	6				
eleven	4			4	
twelve	2				
five	5				
six	9				
nine	3				
one	4				
three	2				
five	6	1	9		
seven	5				
nine	1				
one	5				
eleven	1			2	
twelve	4				
eight	3				
four	2			3	
two	5				
one	4				
six	4			4	
twelve	2				
five	2				
one	4				

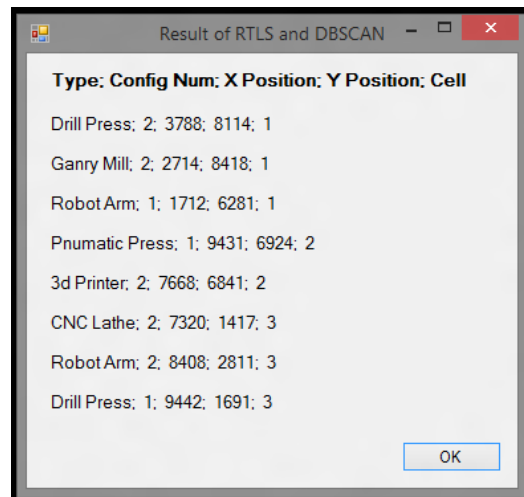
15.16 Appendix P: Knowledge Base for System Test

Machine Name	Machine Type	Software Architecture	X Dim	Y Dim	HW Config	SW Config	X Pos	Y Pos	Num in Cell	Cell No.	Layout No.	Cell Duty	Cell Product	Layout Product		
UP Printer 1	3d Printer	PC	400	400	2	8	4000	8000	1	1	6	Print & Assemble	Prod 5	Prod 5 & 6		
FANUC Arm 1	Robot Arm	FANUC	3000	3000	2	4	2750	8000	2							
UP Printer 2	3d Printer	PC	400	400	1	2	1500	6500	3							
Kress Drill 1	Drill Press	PLC	400	600	3	4	7400	2350	1	2	6	Mill & Assemble	Prod 6		Prod 5 & 6	
UKZN Mill 1	Gantry Mill	Mach 3	1000	850	3	4	9400	4250	2							
FANUC Arm 1	Robot Arm	FANUC	3000	3000	1	5	9400	2650	3	1	2	Drill & Mill	Prod 2			Prod 1 & 2
Festo Press1	Pnumatic Press	PLC	500	350	3	1	11400	2350	4							
Kress Drill 1	Drill Press	PLC	400	600	2	5	4000	8000	1							
UKZN Mill 1	Gantry Mill	Mach 3	1000	850	2	4	2750	8000	2	2	2	Print & Press	Prod 1	Prod 1 & 2		
FANUC Arm 1	Robot Arm	FANUC	3000	3000	1	1	1500	6500	3							
Festo Press 1	Pnumatic Press	PLC	500	350	1	5	9500	7000	1	3	2	Machining Cell 1	Prod 1		Prod 1 & 2	
UP Printer	3d Printer	PC	400	400	2	3	7500	7000	2							
EMCO Lathe 2	CNC Lathe	Mach 3	1200	500	2	4	7000	1500	1							
FANUC Arm 2	Robot Arm	FANUC	3000	3000	2	6	8300	3000	2	1	5	Machining Cell 3	Prod 9			Prod 9 & 10
Kress Drill 2	Drill Press	PLC	400	600	1	2	9600	1500	3							
EMCO Lathe 1	CNC Lathe	Mach 3	1200	500	2	6	8600	8000	1							
FANUC Arm 1	Robot Arm	FANUC	3000	3000	3	5	9300	2700	1	3	5	Assemble	Prod 10	Prod 9 & 10		
Festo Press 1	Pnumatic Press	PLC	500	350	4	4	10500	3600	2							
Festo Press 2	Pnumatic Press	PLC	500	350	4	4	1800	8700	1	4	5	Print & Press	Prod 10		Prod 9 & 10	
UP Printer	3d Printer	PC	400	400	2	5	3500	8700	2							
UKZN Engraver	Gantry Engraver	Mach 3	600	400	3	5	1900	4900	1	1	7	Machine, Print & Assemble	Prod 7			Prod 7
EMCO Lathe 1	CNC Lathe	EMCO	1200	500	3	2	3600	4900	2							
UP Printer 1	3d Printer	PC	400	400	4	3	5500	4900	3							
FANUC Arm 1	Robot Arm	FANUC	3000	3000	2	6	2900	3700	4	2	3	Machining Cell 2	Prod 3	Prod 3		
EMCO Lathe 1	CNC Lathe	Mach 3	1200	500	3	4	2000	5500	1							
FANUC Arm 1	Robot Arm	FANUC	3000	3000	3	7	3300	7000	2	1	7	Machine, Print & Assemble	Prod 7		Prod 7	
Kress Drill 1	Drill Press	PLC	400	600	1	3	4600	5500	3							
EMCO Lathe 2	CNC Lathe	Mach 3	1200	500	3	4	7000	1500	1							
FANUC Arm 2	Robot Arm	FANUC	3000	3000	3	7	8700	3000	2	2	3	Machining Cell 2	Prod 3			Prod 3
Kress Drill 2	Drill Press	PLC	400	600	1	3	10600	1500	3							
UP Printer 1	3d Printer	PC	400	400	3	2	1350	6400	1	1	7	Machine, Print & Assemble	Prod 7	Prod 7		
UKZN Mill 1	Gantry Mill	Mach 3	1000	850	2	3	3350	8300	2							
FANUC Arm 1	Robot Arm	FANUC	3000	3000	2	5	3350	6700	3							
Festo Press 1	Pnumatic Press	PLC	500	350	1	5	5350	6400	4	2	7	Machine, Print & Assemble	Prod 7		Prod 7	
Up Printer 2	3d Printer	PC	400	400	3	2	7400	2350	1							
UKZN Mill 2	Gantry Mill	Mach 3	1000	850	2	3	9400	4250	2	1	1	Machining Cell 1	Prod 1			Prod 1
FANUC Arm 2	Robot Arm	FANUC	3000	3000	2	6	8300	3000	2							
Kress Drill 2	Drill Press	PLC	400	600	1	2	4600	5500	3							
EMCO Lathe 2	CNC Lathe	Mach 3	1200	500	2	4	7000	1500	1	2	1	Machining Cell 1	Prod 1	Prod 1		
FANUC Arm 1	Robot Arm	FANUC	3000	3000	2	6	3300	7000	2							
UP Printer 1	3d Printer	PC	400	400	2	8	4000	8000	1	1	9	Print & Assemble	Prod 5		Prod 5 & 8	
FANUC Arm 1	Robot Arm	FANUC	3000	3000	2	4	2750	8000	2							
UP Printer 2	3d Printer	PC	400	400	1	2	1500	6500	3							
EMCO Lathe 1	CNC Lathe	Mach 3	1200	500	3	3	4000	1500	1	2	9	Lathe, Drill & Weld	Prod 8			Prod 5 & 8
FANUC Arm 2	Robot Arm	FANUC	3000	3000	2	4	6000	2600	2							
EMCO Lathe 2	CNC Lathe	Mach 3	1200	500	3	2	7000	1500	3	1	8	Lathe, Drill & Weld	Prod 8	Prod 8		
Kress Drill 2	Drill Press	PLC	400	600	1	3	8500	2000	4							
Mitsubishi	Spot Welder	PLC	700	1200	3	3	9000	2000	5							
EMCO Lathe 2	CNC Lathe	Mach 3	1200	500	1	3	2000	6500	1	1	4	Print & Engrave	Prod 4		Prod 3 & 4	
UKZN Engraver	Gantry Engraver	Mach 3	600	400	1	2	4000	6500	2							
EMCO Lathe 1	CNC Lathe	EMCO	1200	500	1	6	10000	7500	1	2	4	Print & Mill	Prod 3			Prod 3 & 4
UP Printer 1	3d Printer	PC	400	400	1	3	8000	7500	2							
FANUC Arm 1	Robot Arm	FANUC	3000	3000	3	3	6000	9000	3	3	4	Machining Cell 2	Prod 3	Prod 3 & 4		
EMCO Lathe 2	CNC Lathe	Mach 3	1200	500	3	4	7000	1500	1							
FANUC Arm 2	Robot Arm	FANUC	3000	3000	3	7	8300	3000	2							
Kress Drill 2	Drill Press	PLC	400	600	1	3	9600	1500	3	1	8	Lathe, Drill & Weld	Prod 8		Prod 8	
EMCO Lathe 1	CNC Lathe	Mach 3	1200	500	3	3	4000	1500	1							
FANUC Arm 2	Robot Arm	FANUC	3000	3000	2	4	6000	2600	2							
EMCO Lathe 2	CNC Lathe	Mach 3	1200	500	3	2	7000	1500	3	1	8	Lathe, Drill & Weld	Prod 8	Prod 8		
Kress Drill 2	Drill Press	PLC	400	600	1	3	8500	2000	4							
Mitsubishi	Spot Welder	PLC	700	1200	3	1	9000	2000	5							

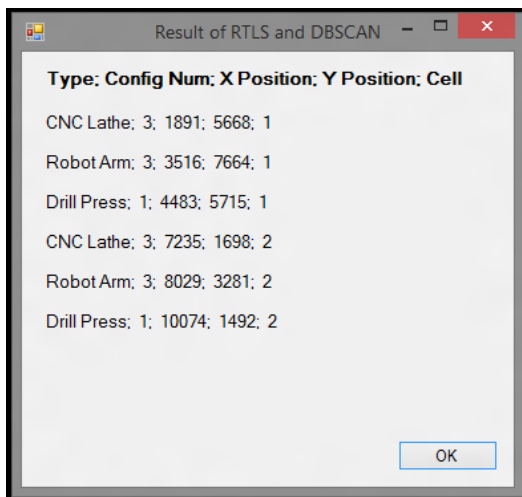
15.17 Appendix Q: Factory Floor States for 11.6



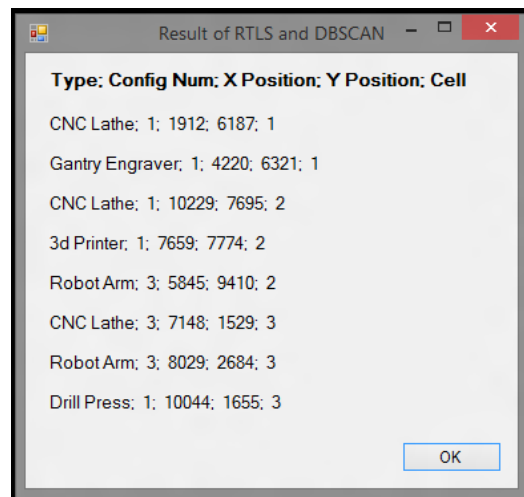
Result: Factory Floor State 1



Result: Factory Floor State 2



Result: Factory Floor State 3



Result: Factory Floor State 4