

**DESIGN AND CONSTRUCTION OF MEERCAT:
AN AUTONOMOUS INDOOR AND OUTDOOR
COURIER SERVICE ROBOT**

Peter Antoon Bosscha

207527345

In fulfilment of the degree of Master of Science in Mechatronic Engineering at the School of Mechanical Engineering, University of KwaZulu-Natal.

March 2011

Supervisor: Prof. Glen Bright

Declaration

As the candidate's Supervisor, I agree / do not agree to the submission of this thesis.

I, Peter Antoon Bosscha, declare that

- (i) The research reported in this thesis, except where otherwise indicated, is my original work.
- (ii) This thesis has not been submitted for any degree or examination at any other university.
- (iii) This thesis does not contain other persons' data, pictures, graphs or other information, unless specifically acknowledged as being sourced from other persons.
- (iv) This thesis 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, their writing has been placed inside quotation marks, and referenced.
- (v) Where I have reproduced a publication of which I am an author, co-author or editor, I have indicated in detail which part of the publication was actually written by myself alone and have fully referenced such publications.
- (vi) This thesis does not contain text, graphics or tables copied and pasted from the Internet, unless specifically acknowledged, and the source being detailed in the thesis and in the Reference sections.

Signed:

.....

Peter A Bosscha

Acknowledgements

I would like to thank Professor Glen Bright from the University of Kwa-Zulu Natal for his guidance and advice with this work.

I would like to thank the Council for Scientific and Industrial Research (CSIR) for funding this project and degree.

Thank you to my colleagues from the Mechatronics Group at the CSIR, who have at various stages been sounding boards, advisors, critics, sympathetic ears and idea generators.

Abstract

This project details the construction and development of, and experimentation with a mobile service courier robot named Meercat. This robot has been built from the ground up using parts sourced from various places. The application for this service robot is the delivery of internal mail parcels between the buildings situated on the campus of the Council for Scientific and Industrial Research (CSIR) in Pretoria. To achieve this, the robot has to be able to localise and navigate through indoor office and laboratory environments and over outdoor tarred roads which interconnect the various buildings.

Not many robots are intended for operation in both indoor and outdoor environments, and to achieve this, multiple sensing systems are implemented on the platform, where the correct selection of sensing inputs is a key aspect. Further testing and experiments will take place with algorithms for localisation and navigation. As a limited budget was available for the development of this robot, cost-effective solutions had to be found for the mechanical, sensing and computation needs.

The Mechatronics group from the Mechatronics and Micro Manufacturing (MMM) competency area at the CSIR is involved with the development of various autonomous mobile robots. The particular robot developed in this project will be an addition to the CSIR's current fleet of robots and will be used as a stepping stone for experimentation with new sensors and electronics, and the development of further positioning and navigation algorithms.

Contents

Acronyms	x
List of Figures	xi
1 Introduction.....	1
1.1 Background information and research objectives	1
1.2 Problem statement.....	3
1.3 System specifications	4
1.4 Definitions, assumptions, limitations and delineations	5
1.5 Thesis outline and chapter overviews	5
1.5.1 Literature review	5
1.5.2 Mechanical design.....	5
1.5.3 Sensors	6
1.5.4 Electronic design.....	6
1.5.5 Software design.....	6
1.5.6 Control system design and experimentation	6
1.5.7 Validation and conclusions	6
2 Literature Survey.....	7
2.1 Background	7
2.1.1 Mobile robotics history	7
2.1.2 Service robot applications	8
2.2 Classifications and terminology	10
2.3 Locomotion	11
2.4 Perception.....	13
2.4.1 Odometry	13
2.4.2 Magnetic sensors	14
2.4.3 Inertial sensing	15

2.4.4	Range finder sensors	16
2.5	Localisation	17
2.5.1	Global navigation satellite systems	19
2.6	Cognition	20
2.6.1	Autonomy levels	20
2.7	Planning and navigation algorithms	22
2.7.1	Road mapping	22
2.7.2	Cell decomposition	23
2.7.3	Potential field algorithm	23
2.7.4	Obstacle avoidance	24
2.8	Software frameworks	24
2.9	Summary	25
3	Mechanical Design	26
3.1	Mechanical design parameters	26
3.2	Concept drawings	26
3.3	Mechanical component selection	28
3.4	Vehicle iterations	30
3.5	Summary	31
4	Sensors	32
4.1	Motion sensors	32
4.1.1	Wheel encoders	32
4.1.2	Accelerometers	34
4.1.3	Gyroscopes	37
4.2	Obstacle sensors	37
4.2.1	Bumper sensors	37
4.2.2	Ultrasonic range sensors	38
4.2.3	Infrared beam sensor	39
4.2.4	Laser beam sensor	41
4.3	World position sensors	42

4.3.1	Compass	42
4.3.2	GPS	44
4.3.3	Differential GPS.....	45
4.4	Summary	47
5	Electronic Design.....	48
5.1	I/O board: Generic CPU.....	49
5.2	EVK1100 AVR32 UC3A evaluation board.....	50
5.3	Embedded personal computer	52
5.4	Radio remote control interface.....	53
5.5	3COM WiFi router.....	54
5.6	MotorController: AX2550	55
5.7	Human feedback	56
5.8	Power system	57
5.9	Summary	58
6	Software Design.....	59
6.1	Generic CPU software.....	60
6.2	AVR32 software	61
6.2.1	FreeRTOS and software framework	62
6.2.2	Control task.....	63
6.2.3	MODBUS RTU protocol task.....	63
6.3	Player software.....	64
6.4	Development and debugging.....	66
6.4.1	Communication protocol testers.....	67
6.4.2	Mapping application.....	68
6.5	Summary	69
7	Control System Design and Experimentation	70
7.1	Meercat control modes.....	70
7.2	Odometry calibration	74
7.3	Localisation and navigation experiments.....	79

7.3.1	Wall-following experiments.....	79
7.3.2	First odometry test.....	79
7.3.3	Compass testing.....	80
7.3.4	Gyroscope testing.....	82
7.3.5	Odometry and GPS test.....	84
7.3.6	Advanced Monte Carlo localisation experiment.....	85
7.3.7	Navigation experiment.....	86
7.3.8	Differential GPS navigation experiment.....	87
7.4	Summary.....	89
8	Validation and Conclusions.....	90
8.1	Validation of robot construction.....	90
8.1.1	Mechanical components.....	90
8.1.2	Sensors.....	91
8.1.3	Electronics.....	93
8.1.4	Power and drive.....	93
8.1.5	Computing and communication.....	94
8.2	Validation of localisation.....	95
8.2.1	Indoor localisation.....	95
8.2.2	Outdoor localisation.....	95
8.3	Validation of navigation.....	95
8.4	Validation of specifications.....	96
8.5	Conclusions.....	97
8.6	Summary of contributions.....	98
8.7	Recommendations.....	98
	Bibliography.....	101
	Appendix A: Meercat Electrical System.....	110
	Appendix B: How Robotics Research keeps re-inventing the Wheel.....	111
	Appendix C: Meercat Mechanical Drawings.....	112
	Appendix D: “Generic CPU Board Schematic”.....	114

Appendix E: “AVR32 CPU Board Schematic”	118
Appendix F: “Generic CPU Project Definition and Files”	133
Appendix G: “AVR32 Project Definition and Files”	134
Appendix H: “Player Driver & Client Files”	135

Acronyms

ADC	Analogue-to-digital converter
AMCL	Adaptive Monte Carlo localisation
AVR/AVR32	The manufacturer ATMEL is coy about what the acronym for the microprocessor stands for, however, the theory is that it relates to the inventors' names.
CPU	Central processing unit
CS	Chip Select
CSIR	Council for Scientific and Industrial Research
DARPA	Defence Advanced Research Projects Agency
D-GPS	Differential global positioning system
GCC	GNU compiler collection
GNU	GNUs not UNIX
GPL	GNU general public licence
GPS	Global positioning system
GSM	Global system for mobile communications
I/O	Input and output
IC	Integrated circuit
I2C	Inter-integrated circuit, a multi-master serial interface invented by Philips
IEEE	Institute of Electrical and Electronics Engineers
JPEG	Joint photographic experts group
MMM	Mechatronics and Micro Manufacturing
MSB	Most significant bit
ND	Nearness diagram
PC	Personal computer
PID	Proportional integral and derivative
PWM	Pulse width modulation
SLAM	Simultaneous localisation and mapping
SPI	Serial peripheral interface
UART	Universal asynchronous receiver and transmitter
VNC	Virtual network computing
VFH	Vector field histogram

List of Figures

Figure 1.1: Roomba vacuum cleaner	2
Figure 2.1: Robot infrastructure [32]	11
Figure 2.2: Robot odometry straight line error accumulation	14
Figure 2.3: Probability grid	18
Figure 2.4: Sub-optimal path planning	22
Figure 2.5: Fixed decomposition grid	23
Figure 3.1: Concept drawing 1	27
Figure 3.2: Concept drawing 2	27
Figure 3.3: Concept drawing 3	28
Figure 3.4: EMD PM50/63 Drive motor and torque curve	29
Figure 3.5 (a,b,c): Meercat evolution	30
Figure 4.1: Motor and wheel assembly plus encoder kit	33
Figure 4.2: WheelWatcher encoder signals	33
Figure 4.3: MXD2020 Accelerometer on Rapid Prototype board	34
Figure 4.4: Accelerometer data, raw and filtered	35
Figure 4.5: Accelerometer data, first derivative and standard deviation	36
Figure 4.6: Analogue devices: ADIS16350 combined accelerometers and gyroscopes	36
Figure 4.7: MD100 Gyroscope	37
Figure 4.8: Bumper sensors	38
Figure 4.9: MaxSonar-EZ1 ultrasonic sensor	38
Figure 4.10: Ultrasound sensors placement and beam patterns	39
Figure 4.11: Hokuyo PBS-03JN infrared beam sensor	40
Figure 4.12: Hokuyo PBS-03JN beam pattern	40
Figure 4.13: SICK LMS200 laser range finder	42
Figure 4.14: Magnetometers raw values	43
Figure 4.15: Normalised magnetometer values	44
Figure 4.16: Compass heading	44
Figure 4.17: u-blox EVK-5H GPS evaluation kit	45
Figure 4.18: Trignet stations map	46
Figure 5.1: Electronics tray details	48
Figure 5.2: Generic CPU circuit board	50
Figure 5.3: ATMEL EVK1100 board	52

Figure 5.4: Single board computer in enclosure.....	53
Figure 5.5: Custom remote control (a) and Meercat receiver (b).....	54
Figure 5.6: 3COM OfficeConnect wireless router	55
Figure 5.7: RoboteQ AX-2550 motor controller.....	56
Figure 6.1: Meercat high-level software structure	59
Figure 6.2: AVR32 software	62
Figure 6.3: Excel MODBUS spreadsheet.....	64
Figure 6.4: Player structure in Meercat.....	65
Figure 6.5: Generic CPU and MODBUS protocol testers.....	67
Figure 6.6: Map builder application	68
Figure 7.1: Meercat position2D interface and linkages.....	71
Figure 7.2: Velocity mode control diagram	72
Figure 7.3: Position mode control diagram.....	74
Figure 7.4: Differential drive definitions	75
Figure 7.5: Calibration run errors.....	77
Figure 7.6: Effect of uneven wheels.....	78
Figure 7.7: Experiment 1, Odometry.....	80
Figure 7.8: Odometry and compass data.....	81
Figure 7.9: Compass data, motor effects.....	82
Figure 7.10: Raw gyroscope data.....	83
Figure 7.11: Corrected gyroscope data.....	83
Figure 7.12: Odometry and GPS, indoors and outdoors	84
Figure 7.13: CSIR Building 14F & 14E, Source: Google Earth	85
Figure 7.14: AMCL localisation sequence.....	86
Figure 7.15: Wavefront route planning	87
Figure 7.16: Differential GPS position determination	88

1 Introduction

This chapter presents the outline of the project and provides background information, followed by the motivation for the study. It also includes a discussion of the objectives to be achieved as well as the project specifications, and finally gives some brief chapter overviews.

1.1 Background information and research objectives

Mobile robots are researched world wide on a large scale. Many universities and research organisations have one or more research areas focused on developing hardware and software for these systems. Most of the developed machines can be classified as service robots, that is, machines intended to perform a service to lighten a person's work. Service robots encompass an extensive field of study with a myriad applications, involving many academic and commercial organisations playing active roles in their development. A number of organisations have been created to track and attempt to coordinate the development of these systems, e.g. the International Federation of Robotics [1] and the IEEE Technical Committee on Service Robotics [2].

Per definition, a service robot is a robot that operates autonomously, performing services useful to the well-being of man and servicing equipment, excluding manufacturing operations. Its applications are diverse, but can be classified as follows [3] :

- Servicing people (personal safeguarding, entertainment, etc)
- Servicing equipment (maintenance, repair, cleaning etc)
- Performing an autonomous function (surveillance, transport, data acquisition, etc) and/or servicing robots that cannot be classified in the two groups above.

One of the applications of a service robot is to provide domestic aid for the elderly and disabled, serving various functions ranging from cleaning to entertainment. Due to high labour costs in developed countries, and the emergence of a greying population in need of personal assistance, especially in Europe and Japan, the service robotics market has developed into a robust field.

As service robots are in close proximity to people, safety concerns in human-machine interaction are an important factor. However, the intelligence of such a system is also crucial to

its functioning since it needs to cope with a variety of unpredictable conditions. There are alternative fun solutions, which facilitate interaction between dumber robots and people. The Tweenbots, which the creator classifies as human-dependent robots, featured in an experiment in New York's Central Park [4], showing that this interaction is possible, and indeed, the robot seems to function surprisingly well.

Leading robotics companies continue to make progress in the manufacture of intelligent and safer robots, preparing for the day when every home will have a robot, or indeed every person will have a personal assistance robot [5]. Already service robots are making significant advances in assistance with menial home tasks, e.g. the Roomba vacuum cleaner developed by the company iRobot [6], shown in [Figure 1.1](#), and related family members, which achieved sales volumes of five million units [7].



Figure 1.1: Roomba vacuum cleaner

Service courier robots also exist. Typical applications are those in industry where line-following carts carry supply parts, or in hospitals where the robot carries medication, linen and other supply needs.

Some of the difficulties that are delaying the wider acceptance and use of service robots are:

- development of specialised mechanical mechanisms for specific chores
- development of autonomous systems, sensors and intelligence to guide the service robot
- power requirements
- market fears and acceptance (of robots)

This project comprises the development of an autonomous vehicle that can be used for the delivery of parcels from one building to another within the CSIR campus. This service robot can either be an inter-building courier, from reception to reception, or could possibly eventually even be a desk-to-desk service. As the Mechatronics and Micro Manufacturing group at the

CSIR is in the habit of using animal names for their platforms, and the initial design of the vehicle showed an upright posture, the name Meercat was adopted for this platform.

The robotics industry in South Africa relies heavily on expensive overseas expertise and components. One of the objectives of this research study was to try and find low-cost solutions for the mechanical design, electronics, sensors and software required for the construction of a service courier robot. Eventually, these efforts could lead to the establishment of local robotics skills or even locally designed robotics systems for specific South African problems.

The study investigates and tests several obstacle and environment sensors, and implements navigation and localisation algorithms for the vehicle. The Meercat should be able to locate itself and navigate equally well in indoor and outdoor environments. Along the way, this study also discusses the obstacles and design steps for the development of the vehicle.

The Mechatronics and Micro Manufacturing group at the CSIR is in the process of developing several service robots with a focus on building skills and capabilities. This vehicle forms part of this initiative and will contribute to the functioning of the group. In future, the Meercat robot will be used as a research platform for further studies on sensors, navigation and localisation algorithms.

1.2 Problem statement

Robotics research and development has covered extensive work around the problem of vehicle position and movement tracking. Localisation is one of the fundamental problems in robotic vehicles since in many applications a robot needs to know its location before being able to navigate and perform its tasks.

Solutions could include the use of global positioning systems (GPS) and the addition of other devices such as odometry sensors, etc. However, in an indoor environment no reliable GPS signal is available. Therefore, in this case other sensors are needed to track the vehicle's position accurately. Beaconing systems are a possible solution to the problem around indoor navigation, and examples are the expensive Northstar system [8] or MIT's Cricket system [9]. These systems can be very effective, but the implementation of such an infrastructure on a wider scale than a laboratory is not considered to be practical for this project.

There are significant complexities involved in constructing a combination of sensors and algorithms to devise a robot capable of locating and navigating in an indoor and outdoor environment reliably without human intervention. One solution is to use massive amounts of

computing power and very expensive sensors, as can be witnessed on some of the vehicles in the DARPA races [10]. However, as this project has no access to rich computational or sensor resources, nor the required manpower, alternative low-cost inventive solutions must be found.

The research objectives for this project are therefore the design and construction of a mobile courier robot that:

- can traverse an indoor office environment as well as an outdoor campus environment, using sufficiently low-cost mechanical components, sensors and electronics, as well as power and drive systems; and is equipped with its own stand-alone computing source and can be easily interfaced remotely.
- can accurately locate itself through different localisation methods in indoor and outdoor environments, as applicable, such that it understands its position at all times.
- can accurately navigate itself in indoor and outdoor environments, be able to reach intended goals and whilst navigating, avoid expected and unexpected obstacles.

Meeting the objectives of this project will assist in the gathering of knowledge and skills on design, construction, testing and operation of a mobile service robot, developing new research areas and project activities in the Mechatronics Group at the CSIR.

1.3 System specifications

Physical dimensions: The platform must be able to easily navigate through corridors and should fit through a single doorway frame. A height restriction of 1.9 m applies, to ensure that the vehicle fits through a door opening.

Payload: The platform must be able to carry mail items and small packages. The intended payload is no more than 5 kg and physical dimensions are no more than 30 x 25 x 10 cm.

Endurance: The system must be able to operate over a full eight-hour working day without requiring recharge. This does not mean that the system is in constant motion; a 50% moving/non-moving duty cycle is envisaged.

Speed: The vehicle should be able to move at an average human walking speed of 5 km per hour at least.

Manoeuvrability: The vehicle should be able to manoeuvre in confined spaces and be able to turn on the spot.

Vehicle position accuracy: The vehicle should be able to determine its position indoors and outdoors with similar x and y errors of ± 1 m and orientation errors of less than 10° .

1.4 Definitions, assumptions, limitations and delineations

Various assumptions were made pertaining to the environment in which the Meercat vehicle would be operating:

- The indoor environment is typical CSIR offices and laboratories. These feature varying surfaces such as (plastic) tiling, carpeting and standard office door openings. No attempts will be made to travel above lower ground offices.
- The outdoor environment is the CSIR campus, featuring mostly tarred roads and occupied by pedestrians and vehicles not travelling faster than 40 km/h. Only navigation to wheelchair-friendly buildings is considered, as these usually feature ramped access.
- Gaining access to a building was deemed to be too complex for this study, and a period of manual guidance might be required to transit from outdoors to indoors.

1.5 Thesis outline and chapter overviews

1.5.1 Literature review

The reviewed literature from research studies deemed relevant to this project is discussed in this chapter. A short history is provided, followed by applications for service robots. Autonomy is discussed, followed by a review on perception, localisation and navigation techniques. The chapter concludes with a review on software methods.

1.5.2 Mechanical design

This chapter details the work performed on the mechanical construction of the Meercat mobile robot. It comprises a discussion around some initial concept drawings and material choices, and it provides a description of the final working vehicle.

1.5.3 Sensors

All the sensors implemented on the Meercat robot are classified as motion sensors, obstacle sensors and world position sensors, and discussed accordingly.

1.5.4 Electronic design

This chapter covers the design and implementation of all electronic components on the vehicle, excluding the sensors. It provides details on the computation equipment from the lowest to the highest level, the wireless interfaces, battery system, power distribution and motor controller hardware. The conclusion specifies the achievements.

1.5.5 Software design

All the elements of the Meercat's software implementation are discussed from the lowest embedded level, through to the central controller and the higher level infrastructure. Also discussed are the supporting tools and software elements that were used.

1.5.6 Control system design and experimentation

The position and velocity control algorithms that were implemented between the driver and embedded controller are discussed in detail, followed by the work performed to obtain an accurate odometry solution, as well as a number of experiments which test the robot's sensors, localisation, planning and navigation abilities in indoor and outdoor environments.

1.5.7 Validation and conclusions

In this chapter the outcomes of the development of the system are validated followed by a conclusion on the study. In addition some recommendations are made on further development of and experimentation on the Meercat project.

2 Literature Survey

This chapter will discuss the reviewed literature from research studies deemed relevant to this project. The mobile robotics research field is extensive to say the least, and the challenge therefore is to not get lost in the literature. A short history of mobile robotics is provided, followed by applications for service robots, with a focus on courier robotics applications. This is then followed by a review on important subjects such as perception, localisation, planning and navigation techniques. The chapter ends with a review on software frameworks for mobile robots.

2.1 Background

2.1.1 Mobile robotics history

The mobile robot is generally considered to be a recent invention, but in fact, one of the earliest known mobile robots was invented by Hero of Alexandria (ca. 10 – 70AD), a Greek mathematician, scientist, and inventor, whose device worked by weights hanging from wires. As shown in a recent re-creation by New Scientist, this robot can be very simply programmed to move forward, reverse and stop [11].

In more recent history, the military implemented versions of mobile robots include the Russian radio-operated Teletank of the 1930s [12], World War II's Goliath tracked mine [13] and the more advanced V1 and V2 flying bombs [14]. These latter devices were autonomous once launched and contained primitive autopilots and automated detonation systems.

The writings of science-fiction author Isaac Asimov in the 1940s and 1950s have been a great driver in inspiring researchers and developers to dream about robotic applications; in fact he was the first person to use the word robotics in one of his stories.

The development of electronics, integrated circuits and computing resources that continue to make astonishing advances has had a powerful impact on the evolution of mobile robotics. One of the earliest mobile robots that used artificial intelligence to control its actions was SHAKY, developed in 1969 at Stanford University by Nils Nilsson [15]. This robot was equipped with a visual range finder and binary tactile sensors, and had as objective to navigate through highly

structured environments such as office buildings. During the Moon race that developed in late 1960s, the Russians took the lead in mobile robotics with the first autonomous lunar rover called Lunokhod [16]. The American response came in the 1970s when the Jet Propulsion Laboratory (JPL) developed the Lunar rover [17] designed for planetary exploration. Using vision systems, a laser range finder and other sensors, this robot categorised the terrain it travelled through in terms of accessibility. In the late 1970s Hans Moravec [18] developed CART, a line-following robot, in the Artificial Intelligence Laboratory at Stanford. A multiple-angle camera system on this robot was used to measure distances to obstacles placed in the robot's path. A six-legged walking robot called Dante II was developed by Carnegie Mellon University for the exploration of rough terrains. This robot was used to explore the Mount Spurr volcano in Alaska, sampling volcanic gases. In 1997 NASA's Mars Pathfinder delivered the remotely operated Sojourner rover [19] to Mars, where the rover proceeded to send back images and other data of its travels on the distant planet. In 1998 the Rhino tour guide robot was implemented at the Deutsches Museum by the German University of Bonn. This is only one example of the various types of autonomous tour guides which were developed in that era. Rhino was able to function for six full days whilst guiding museum visitors to various exhibits. During this 47 hour test period it only experienced six collisions [20].

Since then, the rate of development of mobile robots has accelerated to a flood of models and applications, with new models and capabilities emerging almost daily. A last mention of interest is Honda's long-term ASIMO project, which at the time of writing, is the world's most advanced humanoid robot which can explore its environment by walking, stair climbing and even dancing [21].

From this short history study it is clear that robots are advancing on every level, with increasing sophistication and advanced abilities [22].

2.1.2 Service robot applications

Although it could be argued that almost all mobile robots are service robots of some kind, they are classified as service robots when they perform a certain task that makes life easier or safer for people. The applications for service robots are very diverse; a list from the IEEE technological committee on service robots [2] shows the following applications; cleaning and housekeeping, edutainment, humanoids, humanitarian demining, rehabilitation, inspection, agriculture and harvesting, lawn mowers, surveillance, medical applications, mining applications, construction, automatic refilling, guides and office, fire fighters, picking and

palletising, food industry, search and rescue. This list, although exhaustive, is probably by no means complete as many researchers are driven by the utopian dream of automation of all chores. It is also clear that service robots will enter our homes in the near future. Bill Gates famously wrote an article for *Scientific American* on this very subject in 2006 [23]. He concluded that these devices could have just as profound an impact on the way we work, communicate, learn and entertain ourselves as the PC has had over the past 30 years.

In this study, we focus on the application of service robotics in courier applications in office and inter-office operations. Courier robots find their roots in the less sophisticated automated guided vehicles (AGV), which are used to carry goods in warehouses and hospitals. The earliest use of AGVs was in industry where these robots are confined to special areas in a factory. The vehicle tends to be a marker-following robot equipped with obstacle sensors. The marker can be a basic painted line or a magnetic strip.

Warehouses are another industrial area where the transportation of goods by robotic vehicles has been implemented. As warehouses tend to be a very structured environment, the robots in these environments mainly have to contend with each other. One example of a very advanced version of such a system is currently being produced by Kiva Systems as reported on by *IEEE Spectrum* [24].

Transporting goods in hospitals is a more challenging application where the AGV concept transcends into a more intelligent mobile robot able to navigate an environment populated by unforeseen obstacles and people moving about.

An early courier robot used in hospitals was the NASA-funded Helpmate Robot, which was introduced in 1997. The original company developed its own light sensors for obstacle avoidance, which proved to be very effective. Up to 100 of these machines have been deployed in hospitals in the USA [25]. Another example of an automated courier is the TUG robot from Aethon [26], which is also used in hospital and pharmacy applications. As discussed in [27], the implementation of these systems is of obvious benefit, but it requires thorough planning and proper integration with the rest of the logistics system.

Interaction with people is an important aspect of the courier robot's operation. Acceptance of robots in the workplace has so far been achieved by either ensuring no human interaction at all or moving relatively slowly and announcing the vehicle's intentions clearly by means of lights and sounds. Another factor which determines the acceptance of robots is the size and look of the machine. A robot that is smaller in size with a personable appearance may seem less threatening. As robots become increasingly sophisticated in the future, rules will have to be

created to govern their behaviour. In official terminology, this is known as the study of Human-Robot-Interactions (HRI) which is a new field of study.

Far ahead of his time, the previously mentioned writer Isaac Asimov already introduced the “Three Laws of Robotics” in his *I, Robot* short story collection [28], as follows :

- A robot may not injure a human being or, through inaction, allow a human being to come to harm.
- A robot must obey any orders given to it by human beings, except where such orders would conflict with the First Law.
- A robot must protect its own existence, as long as such protection does not conflict with the First or Second Law.

Asimov’s book then proceeds to weave stories around these laws, proving in a playful fashion that they laws could actually function. These stories assume very intelligent and capable robots, however it will only be a matter of time before such systems will become a reality in our modern world.

Bartneck and Okada in their study on robotic interfaces, make an initial attempt to understand and classify the subject matter [29]. The paper “Social and collaborative aspects of interaction with a service robot” makes some very good recommendations on the interaction and usage of service robots in human environments [30].

A good place for further study on this subject is the “International Conference on Human-Robot Interaction” an IEEE organised event which has was initiated in 2006 [31].

2.2 Classifications and terminology

The many components of mechatronics design come together in the creation of a mobile robot, and the problems associated with this activity are diverse. The problems detailed in this study address the fundamental issues around mobile robots. As the modern field of mobile robotics has progressed, specific areas have been demarcated and a classification has been developed [32]. The development path of mobile robotics research is evident in these concepts; efforts on earlier devices were mostly concerned with locomotion but gradually developments moved up the chain.

In order for a mobile robot to move around its environment it has to have actuators to give it motive power. It then uses its sensors to gather information about its environment. The mobile

robot uses the information gathered to make an estimate on its position. It then uses algorithms to plan a path towards its goal whilst avoiding known and unknown obstacles.

Even though it is a somewhat older source, the book *Where Am I* from the University of Michigan [33] gives an excellent introduction to the subject of mobile robotics. Its title belies the book's content, which is far richer in depth than expected in that it describes sensors, locomotion systems, and navigation methods in addition to position determination algorithms.

The concepts of locomotion, perception, localisation and cognition are shown diagrammatically in Figure 2.1 and are discussed below:

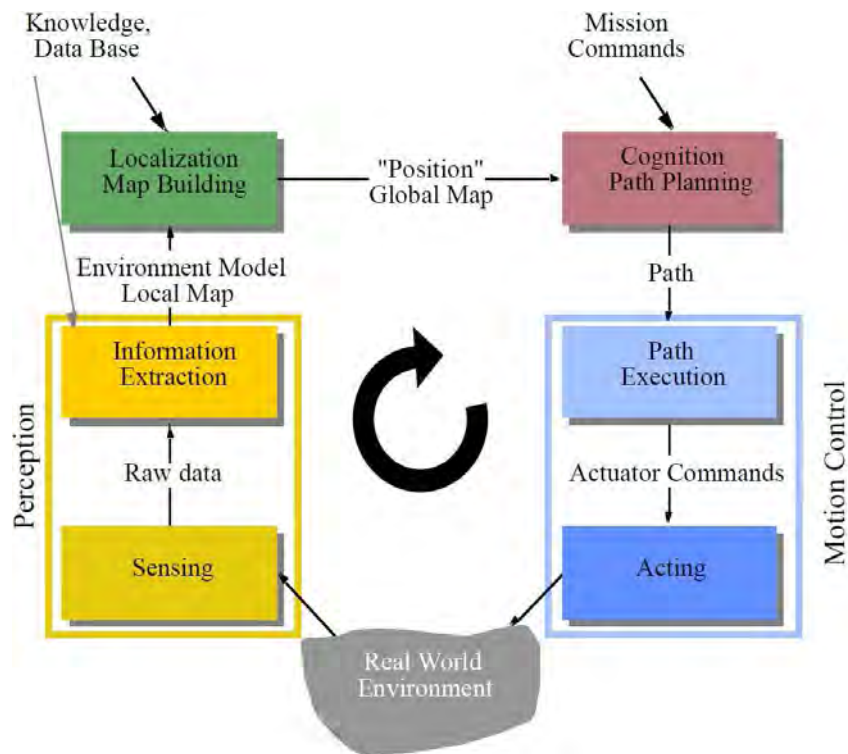


Figure 2.1: Robot infrastructure [32]

2.3 Locomotion

Locomotion in a mobile robot consists of those systems and mechanisms that enable it to move through its environment. Almost all locomotion systems that have been implemented on mobile robots find their basis in nature; these are the walking, jumping, running, sliding, swimming and flying robots. The one exception is the man-invented wheel, a highly efficient locomotion system on flat ground. The core issues of locomotion are stability, characteristics of contact and type of environment [32].

Many locomotion systems can be implemented on a courier vehicle, but one important input into that decision is the fact that the robot has to, by the nature of its mission, manoeuvre in human controlled environments. For movement in spaces created by people, such as offices, inter-office roads and pathways, the wheeled motion is often the simplest and most efficient method of locomotion.

Once wheeled locomotion is chosen, there is a wide array of choices on the actual implementation in terms of number of wheels, steering arrangements and many more options. The Cye Robot [34] implements a two-wheeled differential drive system which is dynamically stable. It achieves this by having the centre of mass beneath the robot's drive shafts. However, as a robot's design expands, this neat solution becomes a problem since the wheels need to become impractically big. The Cye mobile robot solves this problem by adding a towing cart, but it sacrifices manoeuvrability. The robot uses a motor on each wheel; this scheme is known as a differential drive system. This is an easy to implement solution, but it has disadvantages in that uneven friction and uneven wheel sizes easily make the robot travel in curved motions if not corrected.

When the robot's centre of mass is above the wheel axes, usually one or more caster wheels are added for stability. Note that caster wheels with significant loading can easily induce wheel slippage when the platform reverses direction [33]. An exception to this is the Robotic Mobility Platform 100 (RMP100) from Segway [35], a two-wheeled dynamically stable robot which is unfortunately a very expensive solution.

Synchronous drive is a popular solution as it is very well behaved from an odometry point of view, but this solution also has a problem in that the alignment of the base of the platform can drift away from the rest of the robot after time [33].

There are numerous mobile robots that use a four-wheeled solution, starting with an R/C car. The disadvantage of such a wheeled solution is the Ackerman steering, which makes the robot non-holonomic resulting in navigation problems in enclosed spaces.

To achieve holonomic solutions, individually steered and powered caster wheels have been used on the Seekur robot [36], but these are a complex and expensive solution to implement.

Other solutions for omni-directional platforms include omni-wheeled solutions [37], which are based on three or more wheels, and the so-called mecanum or Swedish wheel arrangement [38]. These solutions tend to incur significant vibrations on the platform and are usually only suited

to very flat terrain. In addition, these wheels do not provide good solutions for odometry as the slip on the wheel is highly unpredictable.

2.4 Perception

In a mobile robotic system, perception of the environment is of high importance. Perception is not only the taking of measurements from the sensors that are available to the robot, but also interpreting the results of these measurements in a meaningful manner. A wide variety of sensors are available, which can provide the robot with information about the environment through which it is moving. Sensors can be classified as active or passive, and as either sensors that measure the internal robotics' system, or sensors that measure the external environment in which the robot moves [32]. In this section some of the sensor systems relevant to the Meercat platform are explored further.

2.4.1 Odometry

Odometry is a system of sensors that measures the motion of a robot where these sensors are typically connected to its drive system. The advantage of this system is that it is usually easy and relatively low cost to implement, but the disadvantage is that errors tend to accumulate and some additional correction scheme is required. The science behind odometry is often underestimated by outsiders; a surprising amount of factors affect the accuracy of such a system. The usual assumption is that the odometry system operates perfectly and that the wheel always moves correctly without slip; unfortunately this is not always the case.

Based on the literature reviewed [39], the errors are divided as follows:

- Systematic errors
- Uncertain robot geometry
- Kinematic imperfections
- Non-Systematic errors
- Bumps or uneven floors.

Note that systematic odometry errors have by far the greater influence on the actual robot pose. A graphic representation of odometry error accumulation can be made, as shown in Figure 2.2, where odometry errors increase as the robotic platform progresses. Uncertainty about the orientation of the robot is what causes the errors in the y direction to increase faster, hence the elliptic nature of the error [32].

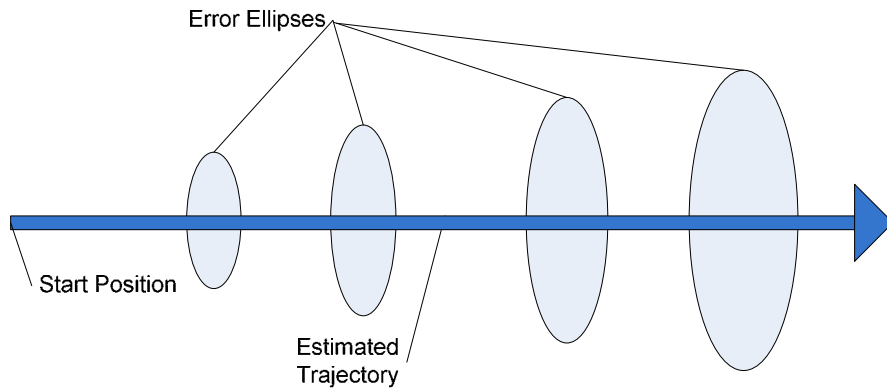


Figure 2.2: Robot odometry straight line error accumulation

The University of Michigan has performed extensive tests on the subject. UMBMark [39] proposed a calibration scheme to correct for systematic errors. It should be noted that although the UMBMark procedure dates from 1994, research into this area is still ongoing with alternatives proposed in [40] and [41].

Non-systematic odometry errors are caused by simple obstacles such as door thresholds, carpet joints or loose cables, and they introduce infrequent additional errors to odometry readings which are difficult to adjust. Gyrodometry [42] appears to be one successful attempt to correct for non-systematic errors. This scheme proposes the use of on-board gyroscopes to correct the robot's odometry when bumps are experienced.

Solutions discussed to date make use of wheel/motor information collection and associated problems. Another option that circumvents the typical errors associated with these systems is visual odometry where mono or stereo vision cameras are used to estimate the motion of the robot. Visual odometry systems have proven to be of a much higher accuracy than wheel odometry, as shown in the Mars Exploration Rovers [43], easily taking into account wheel slip and other effects of robot motion through uncertain terrain. The disadvantage of these systems is the fact that they impose a much higher processing load and the total solution can be quite costly.

2.4.2 Magnetic sensors

Although a magnetic sensor seems ideal for measuring the pose of the vehicle, a major performance drawback for these magnetic sensors is that they are sensitive to other disturbances in the Earth's magnetic field. These disturbances are introduced by other magnetic objects in close vicinity of the sensor as discussed in [32], and include metallic objects used in the construction of the vehicle as well as objects in the environment around the vehicle. A

somewhat cumbersome method to get around this problem is to recalibrate the sensor whenever it is introduced to a new environment or even before every experiment is performed.

Recently, very interesting biologically-inspired research has been done on mapping a magnetic field and using that map for robotic navigation [44], which would suggest that these are viable solutions for indoor navigation at the very least.

2.4.3 Inertial sensing

In mobile robots, inertial sensors are used to measure the accelerations and changes in orientation that a platform experiences. The sensors take the form of either basic single, dual and triple axis reporting devices or more complete Inertial Measurement Units (IMU) which contain three-axis accelerometers and gyroscopes. IMUs should not be confused with Inertial Navigation Units (INU) which are much more complex devices that could contain additional sensors. Typically, they contain a computing device that reports a corrected and calibrated heading and position.

Basic issues with these sensors are noisy performance from accelerometers, resulting in false acceleration reports and drift in gyroscopes, which means that even for a stationary sensor some turn rate is reported. Even very expensive devices exhibit these errors, therefore better methods have to be found to interpret their outputs. An extreme example of these sensors is in the military environment. Intercontinental Ballistic Missiles comprise sensors that are extremely costly, but yield a typical performance of 200 m error after having flown 13000 km [45].

The advent of low-cost accelerometers and gyroscope devices has made them accessible for mobile robots and several experiments have been made to evaluate their effectiveness. A low-cost solid-state accelerometer, combined with a Kalman filtering algorithm for mobile robot positioning, was found to be effective for short periods [46]. A more accurate, but also somewhat more expensive, fibre optic technology-based gyroscope was used for similar experiments [47]. In this paper, it was found that implementing Kalman Filtering only yielded a two-fold improvement whilst careful calibration resulted in a five-fold improvement in operation. This system is very usable over medium distances when combining both filtering and calibration; experiments using distances of over 200 m resulted in errors of less than 10 cm.

Heuristic methods described in [48] combine Kalman filtering, calibration and additional vehicle logic to obtain up to nine-fold improvement in performance of a gyroscope. A spin-off company from the University of Michigan's work performed in this arena offers a heuristic

software module that can be added to any code, claiming up to ten-fold improvement for any type of gyroscope [49].

2.4.4 Range finder sensors

Range finder sensors inform the robot of distances to objects. These sensors operate on different principles and are priced from a very low cost R140 for an ultrasonic rangefinder to a very high cost of R40 000 for a laser range finder.

Ultrasonic range sensors operating in the tens of kilohertz frequencies is a traditional and popular method of obtaining information about an obstacle without incurring much costs. The advantages of the devices are that they are low in cost and are easy to implement, but they have the disadvantage that they can be sensitive to surrounding noise [50].

These devices come in many forms and are used by many robotic platforms as range sensors to detect obstacles. The operating principle of these sensors is fairly simple: an ultrasonic pulse is transmitted (known as a “ping”) and the sensor’s circuitry records the time of flight of the signal until a return echo is received. It must be noted that the results of these sensors are not extremely accurate as the time-of-flight of sound in air is greatly affected by primarily the temperature of the medium, and secondly (but much less so), by the humidity of the medium. Industrial ultrasonic sensors compensate for these effects, but the sensors used in many of the robotics platforms do not do so. The typical ultrasonic range finder works either with separate transmitting and receiving transducers or with a single transducer both transmitting and receiving the ultrasound. The advantage of these latter devices is that they are more compact, but the disadvantage is that they exhibit a so-called “dead-zone” limiting the near detection range.

Another low-cost range-finding solution is the short-range infrared distance sensor. The application of an IR beam is to use reflected light where usually the infrared spectrum is used to avoid measurement complications with visible light. Several versions of this sensor are available, but the performance of these sensors is usually not that good. The sensors tend to be used more for obstacle detection than for ranging results [51].

There is a myriad of simple low-cost range-finding sensors, but the options become increasingly expensive, such as planar scanners followed by 2D and 3D scanners. Most of these technologies are based on laser reflected light. These solutions are very accurate and give very good results, but they tend to be very expensive.

A number of alternative options present themselves. Literature describes a low-cost laser scanner which has a component cost of less than \$30 [52]. This sensor was initially announced to become commercially available as a stand-alone sensor but recently the development direction seems to have been the integration of this sensor onto the NEATO XV-11 Robotic All-Floor Vacuum Cleaner [53], a competitor of the previously mentioned Roomba vacuum cleaner. Alternate cruder attempts can be found on the Internet [54].

The trend seems clear however; the technology, knowledge and skills needed to construct these types of sensors are becoming widely available, and it will only be a question of time before a low-cost scanning laser range-finder will become available.

2.5 Localisation

The process that is used to determine a robot's position is known as localisation. This process is one of the most researched subjects in mobile robotics since there are so many potential solutions [32].

The complexity of the application determines whether a robot needs to understand its position at all. For certain simple robots no localisation is needed as they rely on some simple rules to move around their environment and deal with obstacles. The disadvantage of such a system is that no specific goal can be targeted as the robot never realises where it is located [55, 6].

The next step for moving a mobile robot around in a known environment is the implementation of a rules-based system where certain goals are known and methods for reaching them are pre-programmed into the system. The methodology for this could be line-following for a modified environment or wall-following methods for a non-modified environment. Exceptions to the rules need to be built in the system to overcome certain non-standard obstacles. This is then followed by a simple landmark detection system signifying that the goal has been reached. Usually these landmarks are artificial constructs; a few examples are camera systems sensing specific shapes [56], ceiling lights detection [57] and ultrasonic beacons [9].

The disadvantages of these systems is the fact that the rules for finding the goals are fixed and any change to the environment, be it modifications to the building or relocation of the robot, means that the system rules will in all probability need to be re-programmed.

The next level up is a map-based system where a pre-defined map of the environment is programmed into the system. This is the better system to implement on a courier robot as its

application demands that a specific goal is targeted and reached, or indeed, is perceived to have been reached.

If a map is used for localisation, the precision of the map must closely match the required robot precision. This means that in larger environments, the eventual map can become a large, highly detailed representation, requiring much work if constructed manually. Note that constructing a complete map of a building is not always needed either as a map drawn according to what the robot's sensors detect should be sufficient [32].

The DERVISH robot [58] is well known for its implementation of a topological map, which is a mapping method that requires very little space. This map defines rooms and hallways as specific states, with no indication of any distances. The method is shown to work well, but it only allows a robot to travel from one area to another, with little additional capability. What is very interesting in the localisation methodology in DERVISH is the implementation of a probability matrix consisting of the likelihood of certain conditions, which the robot uses to localise itself according to the highest value.

This methodology is known as Markov localisation and can also be applied to detailed grid maps, improving the robot's location perception. The approach of this method is to build a matrix of all the possible robot poses and evaluate the probability of each potential pose. Figure 2.3 shows a graphical representation of this method where each element represents $p(l_{(x,y,\theta)} | s_1, \dots, s_n)$, which translates as the probability of pose $l_{(x,y,\theta)}$ given all sensor readings; darker elements signify higher probability.

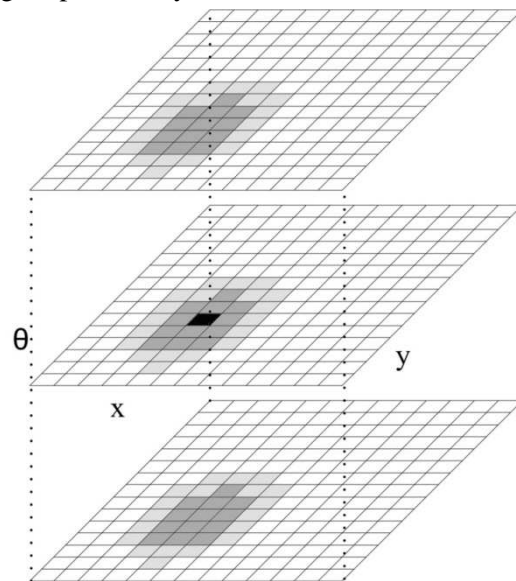


Figure 2.3: Probability grid

The advantage of this method is the ability of robot localisation even when the initial position is completely unknown. However, it is clear that using this method on larger maps would require

enormous computing resources as every potential pose needs to be evaluated. Particle filter or Monte Carlo methods are employed to optimise this process. Here only a random distribution of poses is evaluated, with higher concentrations around local maxima.

Adaptive Monte Carlo Localisation (AMCL) described in [59], uses a combination of methods, starting off with a large set of potential poses. It then proceeds with random sampling which creates local maxima. Once these local maxima are strong enough, the search area is narrowed down, lightening the computational load required for localisation. This process continues until the robot's position on the map is determined with a very high probability.

All previously described methods assume some human intervention, be it extensive in creating artificial landmarks or more limited in creation of a map, but the Holy Grail is a mobile robot that can operate fully autonomously right from the start.

The mobile robot starts from an unknown position and manoeuvres in the environment, noting specific stationary landmarks. It uses these to construct a map of the environment as perceived by the robot[60]. The constructed map is a set of landmark points, which is sufficient for a mobile robot to work with and by which it then localises itself in relation to the known landmarks. This process is known as simultaneous localisation and mapping (SLAM) and is described as the most complex problem in mobile robotics [32].

2.5.1 Global navigation satellite systems

For outdoor localisation use, the GPS is a satellite-based technology created by the USA's military developers to assist in obtaining an accurate position for Navy, Army and Airforce [61, 62]. Alternative solutions using the same principles are Glonass [63] created by the Russian military, and the European system called Galileo [64], which will only become operational in 2014. Generally, GPS is the technology of choice with widespread availability of GPS receivers.

Where initially a GPS receiver was a specialised device mainly used by military and later in professional fields such as transport, surveying and the like, the technology has become quite popular in recent times [65]. The GPS has now entered the consumer market and is appearing in numerous applications and devices. The available devices are becoming cheaper and in some cases are starting to work better, being more sensitive, integrating GSM-based assisted services for faster initial acquisition and integrating inertial sensors to assist with loss of GPS signalling.

Typical GPS accuracies under ideal situations can be better than 5 m, with manufacturers claiming 2.5m [66], but performance is greatly affected by a number of factors. The most

obvious sources of errors are obstacles such as trees and high buildings, which simply limit the number of visible satellites or introduce multi-path reflections. More basically though there are also errors in satellite orbit accuracy and tropospheric/ionospheric delays.

For a higher level of positional accuracy, the standard GPS solution can be augmented by additional correction mechanisms.

Local augmentation systems are one way of receiving these corrections. These are typically performed by specialised radio stations, which calculate the correction data based on their exactly known position and the received GPS information. This augmentation service, known as Differential GPS, is only valid within a limited radius around the station and deteriorates as distance from the reference station increases. Such stations are widely available in the USA but until recently no such service existed in the interior of South Africa. However, coastal support is available in South Africa to assist with shipping. An alternative solution to these radio stations is the use of stations which publish their data on the internet [67]. This solution works well and requires little infrastructure to construct.

Other solutions for obtaining more accurate positioning results include the Wide Area Augmentation Service (WAAS) [68], which is only available in and around the USA, or the subscription to a commercial service. The commercial services typically use an additional satellite signal, requiring very specialised receivers and processing. Examples are John Deere's precision farming Greenstar product range, with equipment being able to provide accuracy of ± 2.5 cm [69] and the OmniStar service [70].

2.6 Cognition

Cognition in a mobile robot describes those processes which allow it to interpret its situation, establish a plan and act accordingly. The next section initially describes the autonomy levels of robots, followed by some of the known planning and navigation methodologies.

2.6.1 Autonomy levels

It should be understood that there are varying levels of autonomy in a service robot. The first level is a basic tele-operated device also known as a remotely operated vehicle (ROV) which is intended to keep an operator out of harm's way. The following level is a guarded tele-operated device where some self-preservation sensors will override an operator's instructions. A well-known example of this is the Mars Pathfinder and Sojourner rover [71,19] which would only move on operator commands, whilst ensuring obstacle avoidance. Because of the time delay of

15 minutes, no real-time control was possible and the platform's self-preservation algorithms acted to ensure the platform would not get into trouble.

Basic autonomy starts playing a role in the next level robot, where a line or other identifier will be followed with possibly additional way-station identifiers. Usually these devices have obstacle detection sensors but no obstacle avoidance algorithms; they will simply wait for an obstacle to be removed. Examples of these systems can be found in devices being sold by companies such as JBT-Corporation [72], Transbotics [73] and Egemin [74].

The next level is an autonomously randomised robot, where obstacles are avoided, and this avoidance results in a randomised pattern of movement. Examples of such robots are the earlier mentioned Roomba [6] and Friendly Robotics' Lawnmower robot [55].

The last level is the fully autonomous robot, which has an idea where it is located and how to navigate to goals in position. This can be in structured or unstructured environments, where very specialised examples are the MicroMouse competition [75]. More advanced examples can be found in the PatrolBot [76] and Seekur [36].

There are no fully autonomous mobile robots. Autonomy is usually expressed in a percentage which can be 99% or higher, but there will always, at some stage, be some form of human intervention in the operation of the robot. In order to classify a mobile robot's performance, Lampe and Chatila propose methods for the measurement of autonomy in [77].

The Defence Advanced Research Projects Agency (DARPA) challenged researchers in two competitions [10] to develop autonomous vehicles capable of navigating desert and urban terrains. Shortly afterwards Google started with their driver-less car project employing members of the winning teams. Using Streetview data and additional sensors mounted on the vehicles, these autonomous vehicles have driven a total distance of 224 000 km [78]. Google claims that one of their vehicles has travelled a distance of 1600 km without human intervention, with the only accident occurring when one vehicle was rear-ended when stopped at a red light.

It is clear that currently most of the more advanced robots operate on a sliding scale of autonomy, alternating between full autonomy, full human control and stages in between.

2.7 Planning and navigation algorithms

Given an at least partial *understanding* of its environment, once a goal is set, a robot can plan for this goal to be reached. Planning has to be performed optimally so that the least amount of effort is made by the mobile robot (see Figure 2.4), but also the processing time by the algorithm has to be optimal. Numerous methods have been researched in the mobile robots community, a few of which are discussed here.



Figure 2.4: Sub-optimal path planning

Step one in the robot's planning is a basic understanding of the shape and size the robot occupies, as well as what type of motions it can make. A robot that is driven by Ackerman steering is only capable of moving through wider curves and is much less manoeuvrable than for instance a differential drive robot. A planning algorithm needs to understand these limitations and use them during the motion planning. Despite external differences, in terms of strategy, the existing planning algorithms can be classified as either road mapping, cell decomposition or potential field algorithms [32].

2.7.1 Road mapping

In the road mapping strategy, the known environment is analysed and a set of virtual roads is constructed. Once this road map exists, any movement of the robot follows these pre-defined paths. Again there are a number of methods proposed to achieve this mapping, the Voronoi diagram [79] is an interesting approach where the distance between the robot and known obstacles is maximised. Once the points on the road map have been established, the shortest route for the robot to travel between points can be established by for instance the use the A* algorithm [80], which in itself is an improvement on Dijkstra's algorithm [81]. The

disadvantage of using this method is the fact that in a robot with limited sensing the obstacles might not even be visible, potentially leading to navigation errors [32].

2.7.2 Cell decomposition

Cell decomposition is another method where the basic idea is to divide a map into free and occupied cells. Several methods exist, but the most interesting and most current method is the fixed-size grid-based representation as it ties in well with computational thinking.

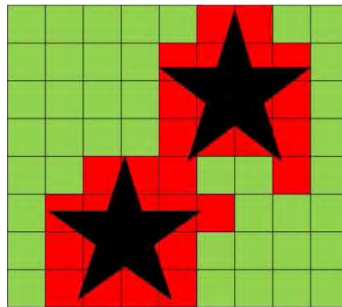


Figure 2.5: Fixed decomposition grid

One problem with this methodology, as can be seen in Figure 2.5, is the fact that this simple mapping approach results in the report of occupied cells even when the actual obstacle only partially occupies the area of the cell. These issues can be addressed by further decomposition of the cells in difficult, tighter spaces or by selecting a smaller overall cell size.

A popular method of planning through occupancy grids is a wavefront algorithm, where the grid is populated with values starting from the goal and each adjacent cell group receiving an increasing value. Travelling towards the goal from the start position then becomes a case of following decreasing numbers.

2.7.3 Potential field algorithm

The last method is the potential field algorithm where the concept of attractive and repulsive fields is used. During planning, the robot is steered away from objects as they have repulsive fields and steered toward the goal by an attractive field. The method has a number of problems in that local minima can appear in certain situations, causing the robot not to reach its target.

2.7.4 Obstacle avoidance

Once planning is completed, the robot is set in motion to reach the goal. Along the way, the robot can potentially discover obstacles around which it needs to navigate. The Bug1 and Bug2 algorithms [82] are of the earliest obstacle avoidance methods, and these attempt to travel towards the target whenever they can. Where Bug1 traverses an entire object before deciding to move on, Bug2 implements a tangential move away from an object whenever this makes sense. As the Bug algorithms have no memory function, their performance is limited and results often in non-optimal movement. The Vector Field Histogram (VFH) was developed [83] to answer some of these problems, where a small local occupancy grid is created from sensor readings. A cost function is applied to the grid and an optimal move results. Improvements to this algorithm followed in VFH+ which takes into account the vehicle kinematics, thus blocking certain routes which in VFH appeared to be open. The Nearness Diagram (ND) [84] is very similar to the VFH algorithm but improves upon it by allowing navigation in cluttered places, the referenced paper shows an impressive set of extremely close navigation tests.

2.8 Software frameworks

Mobile robotics has a long history of experimentation. Several cognitive systems have been developed, where each robotics group seemed to have developed its own standards. Some of the better known examples of these systems are the NASA/JPL developed CLARAty [85], Carnegie Mellon University's CARMEN [86] and the University of California robotics research laboratory developed Player, Stage and Gazebo software suite [87, 88]. The Player project is a GPL [89] licensed open source effort, and is often quoted as the most used robot interface in research [90]. To illustrate the popularity of the Player project, Vaughan and Gerky showed [91] that their software was downloaded over 42 000 times by January 2006. The Player project is actually a combination of three elements:

- Player: a robotic interface
- Stage: a 2D simulator for single or multiple robots
- Gazebo: a 3D simulator for single or multiple robots

The commonly accepted terminology for these framework systems is Robotic Development Environments (RDE), a comparative study on these systems was recently published [92], scoring the RDEs in terms of features, usability, and impact. The authors make the point that in their mind the choice of one RDE over another will in future be based on more than just development support, but increasingly also on available features for longer term operation of a robotic system.

The common denominator for all these frameworks is the fact that the RDE emphasises reuse of code. To develop robotic systems, one has to integrate various technologies and technical disciplines, and the complexities of these systems and their applications are continuously growing in scope. The point is made in [85] that the task of development of robotic software has become too complex and too costly to allow for re-creation of new sets of code. Therefore all RDEs attempted to implement a separation of algorithms, robotic middleware and low-level drivers, which allows for the reuse of code sections across different software and hardware platforms.

A new development that deserves mention is the recent development of the open source Robotic Operating system (ROS) [93] being developed at Willow Garage [94]. The ROS development effort is a continuation of the efforts at the Stanford Artificial Intelligence Laboratory where the Stanford AI Robot (STAIR) was developed [95]. ROS is becoming the new standard of RDEs with many international robotics groups contributing to this software, in addition Willow Garage has developed a number of impressive robots some of which were donated to international research groups.

2.9 Summary

This chapter comprises a literature review of all the necessary components required for the creation of a modern autonomous mobile robot. Some of the ideas from the literature on locomotion, perception, planning and navigation have been used in Meercat system's design and construction. For the software development framework, the Player project seemed a logical choice because of its open source free nature, its popularity, supported devices and number of available planning and navigation algorithms.

3 Mechanical Design

This chapter details the mechanical design and construction of the Meercat mobile robot. The focus of this design was always to ensure that mechanical components could be easily obtained, preferably from local suppliers and at a reasonable cost. A great deal of time was spent initially on the creation of a self-balancing robotic vehicle. The idea for this was based on the developments of David Anderson's nBot balancing robot [96] and Dean Kamen's Segway personal transporter [97]. However, a dynamic balancing solution has a number of practical considerations which affect its functionality in the use of a mobile service robot. One problem is that the platform is constantly in motion in an attempt to balance itself. This effort has a power cost; battery power is constantly consumed during the balancing action, and when the batteries run empty the platform falls over. Another problem is that the motion of the platform influences the sensor behaviour. For example, whilst the sensor might report no obstacles in its field of measurement, this might be as a result of the tilting of the platform rather than an absence of obstacles.

3.1 Mechanical design parameters

The mechanical design parameters can be summarised as follows in order of priority:

- low-cost design
- simple implementation
- modular design, easy adaptation
- robust, as the platform has to work indoors and outdoors.

3.2 Concept drawings

Some concept drawings of the platform were made using Solidworks [98] CAD software, and these were used for design inputs and discussion purposes with colleagues at the CSIR. Concept drawing 1, shown in Figure 3.1, depicts a two-wheeled design, where the idea is that if the platform falls over the damage would be minimal due to its rounded shape.

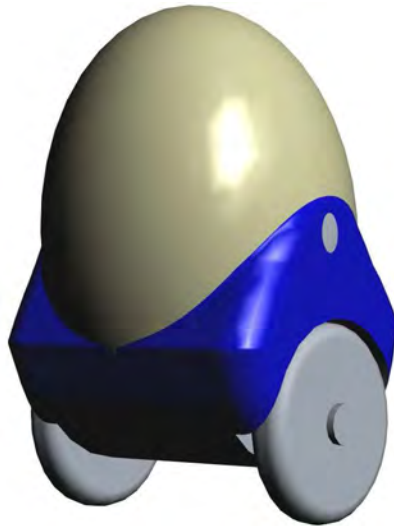


Figure 3.1: Concept drawing 1

A similar concept was explored in concept drawing 2, shown in Figure 3.2, still using two wheels only for motive power, but with the difference that the triangular part of the enclosure houses an actuator which would assist in self-righting the platform when it had fallen over. A transparent top would house some of the sensors and an alert beacon, protected by the enclosure.

Concept drawing 3, shown in Figure 3.3, shows a concept of an adaptable multi-wheeled mechanism, where the wheels could be used for stabilisation of the platform in certain operational modes, but also for self-righting or even angling of the sensors. This latter design generated a lot of interest as the design could in principle also solve problems such as traversing steps in pavements or even stairs, and further research was done into options to implement this design using linear actuators. Unfortunately, no proper low-cost solutions could be found, neither in electrical nor in pneumatic solutions.



Figure 3.2: Concept drawing 2

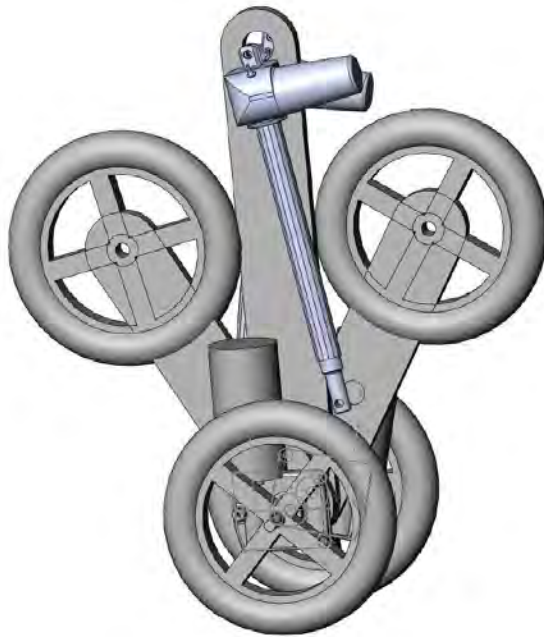


Figure 3.3: Concept drawing 3

The concept drawings were useful in that they generated debate and some design inputs, but in the end, the implementation of the platform was never going to look professionally finished and commercialised. One of the reasons for this is because the design and implementation effort was significant and the “look” of the vehicle was always of secondary consideration. Furthermore, the availability of funding and time was problematic.

3.3 Mechanical component selection

From the concept drawings a final design was made after sourcing the relevant components. Aluminium was chosen as a building material, as it is easy to machine but at the same time has robust and lightweight properties and is locally available. A modular solution was found in the use of aluminium extrusions for the creation of a multiple tiered platform. The thinking there was to divide the functionality into sections; motor and drive electronics in the bottom section, one or two electronics and sensor bays, a battery section above that and finally a parcel bay as the top section.

Electrical motors, hard rubber wheels and sealed lead acid batteries were sourced from a local supplier of wheelchairs and wheelchair components, who in turn obtains these

parts from overseas suppliers. An advantage of this sourcing method is that the project benefited from the higher purchase quantities of the supplier. From a mechanical design point of view, the design is not optimal as the available power of the motors, as shown in Figure 3.4, is more than is required for the vehicle. In a wheelchair configuration these motors together with the selected wheels are claimed by the supplier to be capable of moving a person of 120 kg at up to 10 km/h. Nevertheless, having the spare capacity is useful, and further expansion of the platform is possible without reconfiguration of the mechanical system.

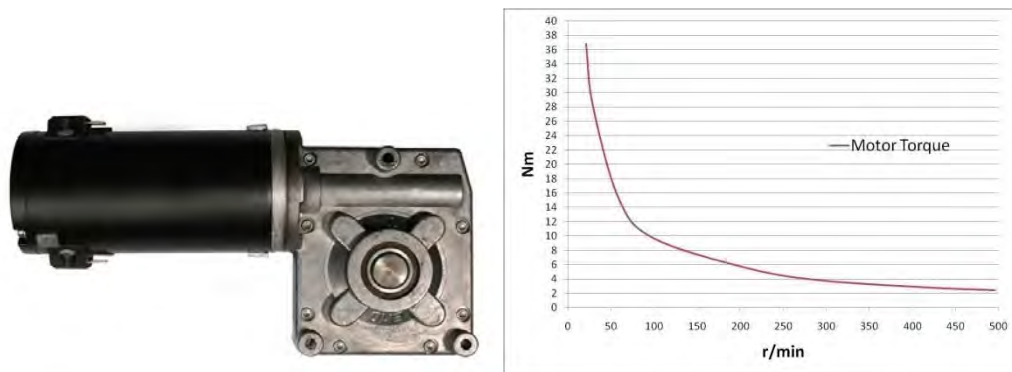


Figure 3.4: EMD PM50/63 Drive motor and torque curve

Given a wheel radius (r) of 15 cm, the weight of the robot at 40 kg, a maximum road incline of 15° (θ) and an assumed rolling resistance coefficient of 0.01 (C_{rr}), some verification calculations were done in (3.1) – (3.3). Matching these calculations to the torque curve in Figure 3.4, it is clear that the motors will be able to propel the robot on the maximum inclines at well over 5 km/h which is more than sufficient for the vehicle’s needs.

$$T = F_{normal} r + F_{Friction} r \quad (3.1)$$

$$T = m g \sin(\theta) r + C_{rr} m g \cos(\theta) r \quad (3.2)$$

$$T = 15.23 + 3.79 = 19.02 \text{ N m} \quad (3.3)$$

This means that even small bumps in the road and the occasional “sleeping policeman” should not pose a problem to the robot.

3.4 Vehicle iterations

As work on the vehicle progressed, the mechanical construction went through a number of iterations, as shown in Figure 3.5 (a,b,c). , where picture (a) shows the first iteration where the vehicle did not have any support wheels, picture (b) shows an intermediate iteration with support wheels on both sides of the vehicle and picture (c) shows the final model vehicle. This was done by re-thinking the platform needs in terms of sensor placement and observed mechanical action in the testing environment. The very first iteration of the vehicle used bushes between the vehicle chassis and the drive-shafts of the motors, but this proved to be a non-optimal solution. The bushes that were planted caused an uneven load, causing the vehicle, especially at low speeds, to significantly veer off course. The bushes were replaced with wheel bearings and these greatly improved the vehicle's performance.

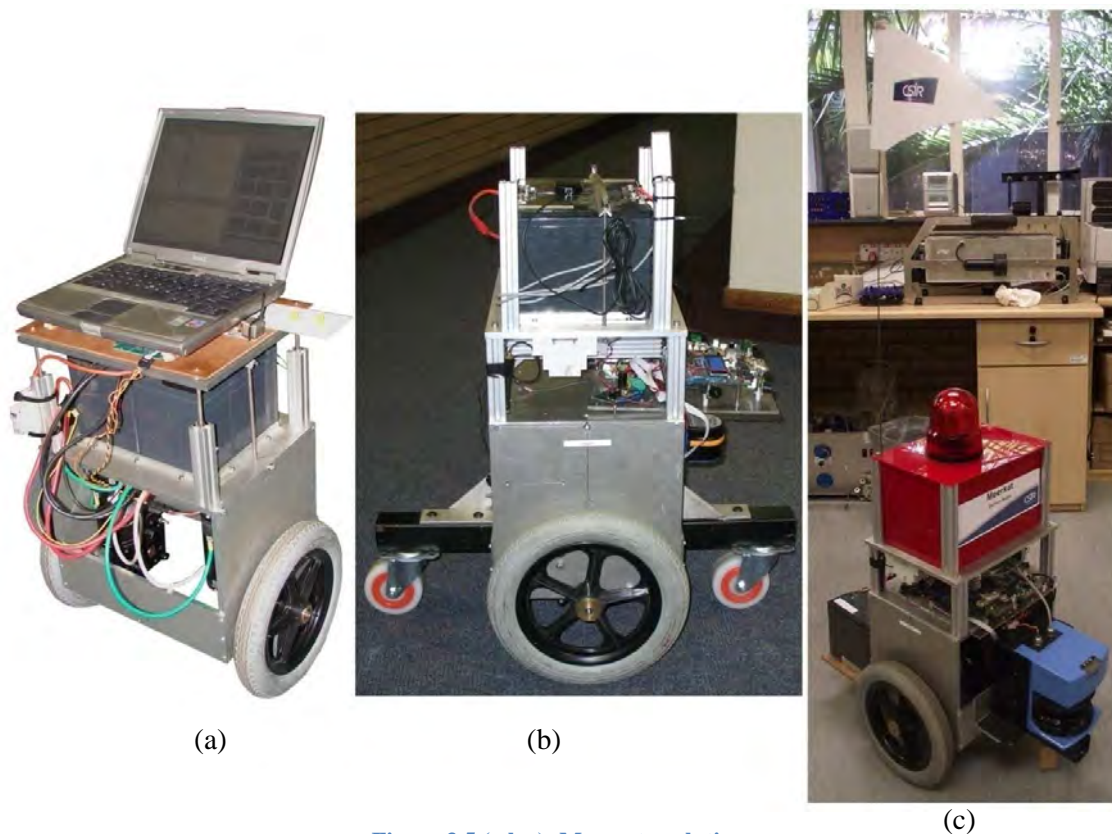


Figure 3.5 (a,b,c): Meercat evolution

Although not a key focus for this project, the choice of support or castor wheels was an important factor affecting the motion of the platform. Various forms of support wheels were sourced from supplier RS Components [99] and tested. The design has alternated from small castor wheels, followed by ball casters. It ended up with two larger diameter castor wheels and then only a single castor wheel. The ball casters functioned well in indoor environments, but were simply not practical outdoors as they caused substantial vibration for the entire robot. The

single caster wheel was made possible by moving the batteries from the top section of the vehicle towards the side of the vehicle changing the weight distribution of the platform. Appendix C shows the mechanical drawings of the current model of the Meercat mobile robot.

3.5 Summary

The mechanical design process described in this chapter started with the concept drawings. Then a final vehicle shape was chosen; and component selection and construction of the vehicle was discussed. Some verification on the vehicle's abilities was performed.

It is realised that a multi-modal vehicle which can alternate between self-balancing during faster motions and three-wheeled (or more) dynamically stable motion during slower periods of motion is the better solution to be implemented, but mechanical and electrical complexity prohibited that idea.

The mechanical development of the Meercat platform did not comprise a formal design method. Rather the platform has evolved to the stage where it now is a very useful and sturdy platform which is able to manoeuvre in a variety of environments.

4 Sensors

This chapter deals with the selection and performance of the sensors used on the Meercat platform. The large number of different sensing technologies that have been fitted allows for much experimentation with a number of alternate algorithms. Over the development period, several types of sensors have been implemented, with the intention of measuring the vehicle's actions as well as its environment and position. Certain sensors were upgraded from time to time using either different types of sensors or similar sensors with improved specifications. Usually the reason for upgrading was because an improvement could be made in the system's performance. The combination of sensors through sensor fusion allows for better position, speed and heading determination.

For the purposes of this discussion, the sensors are divided into motion sensors, obstacle sensors and world position sensors. These sensors are discussed in the following sections of this chapter, followed by individual conclusions on the use of these sensors. In most cases, a drawing or picture is used for clarification. Note that in-depth data can be found in the data pack. The chapter ends with a conclusion on the implemented sensors.

4.1 Motion sensors

The motion sensors on this platform comprise the wheel encoders, accelerometers and gyroscopes. With these sensors, it is possible to determine the vehicle's motion as well as relative position, acceleration and turn rates.

4.1.1 Wheel encoders

An optical encoder disk was attached to the shaft of each of the Meercat's motors before the gearbox, which means that actual motor revolutions are measured. Each motor assembly consists of a motor plus a 1:25 ratio gearbox through which the platform's wheels are driven. The tyre on the wheel is made of solid rubber with a very small compression factor. In this way, it is possible to measure the revolutions of the motors and derive the actual movement of the platform through some mechanical relationships. At higher levels of the system a calibration

factor is introduced to generate an accurate odometry result. Figure 4.1 shows the schematic drawing of the connected wheel encoder and a detailed picture of the wheel encoder kit.

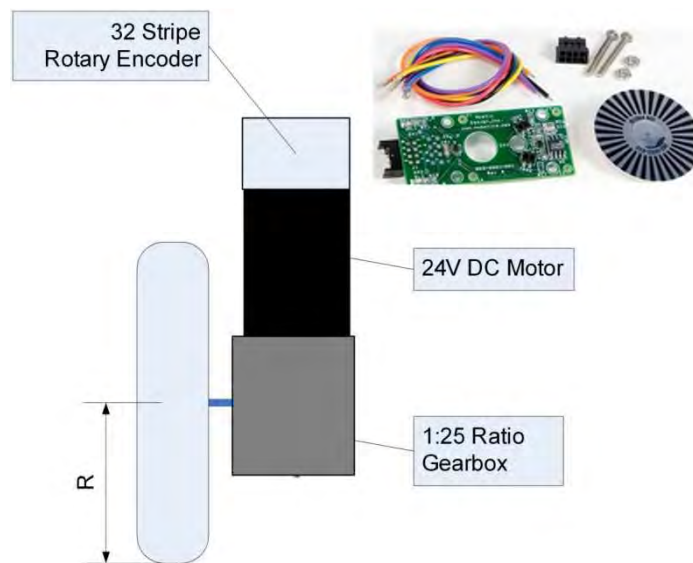


Figure 4.1: Motor and wheel assembly plus encoder kit

The encoders used are from a company called NUBOTICS, type WW2 WheelWatcher Incremental Quadrature Encoder System [100]. The encoder runs on a power supply of 5 V and generates 5 V signals; the behaviour of these signals is shown in Figure 4.2.

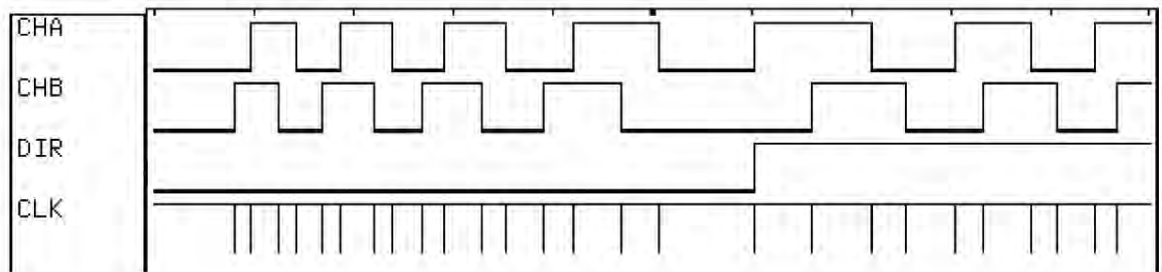


Figure 4.2: WheelWatcher encoder signals

In Figure 4.2 the CHA and CHB signals are the typical signals generated out of a standard quadrature encoder where the phase difference between CHA and CHB indicates the rotational direction, and the DIR and CLK signals are generated by a decoding process which uses CHA and CHB as input signals. The CLK signal signifies a change in the state of the encoder and the DIR signal indicates in which direction the wheel is turning.

All these signals are available on the NUBOTICS encoder system, but it was decided to use the CLK and DIR signals for ease of processing. This means that for each transition, a clock signal is generated and a positive or negative count is made, depending on the polarity of the DIR signal. A disadvantage of this choice is that, in terms of processor load, there is a fairly high

interrupt rate when the motors are both operating at high speed, as both encoders will generate their own interrupts. However, the advantages are that the counting process is much simpler than when using the CHA and CHB signals and each encoder state is recorded, thus giving a higher position resolution. When Meercat travels at a speed of 1 km/h, a frequency of approximately 930 Hz is generated. As there are two encoders connected to the embedded controller, it was decided to limit the speed of the vehicle to no more than 5 km/h even though the platform is capable of at least twice that speed. At this speed the embedded processor receives interrupts at an average interval of 107 μ s which is an acceptable rate for the high-speed processor.

4.1.2 Accelerometers

A number of different accelerometers were used on the platform, which required different interface methods. The accelerometers on the platform were originally intended to assist with the balancing of the vehicle, but when this idea was discarded they were kept to serve as augmentation of the vehicle's odometry at a later stage. The initial version that was used is

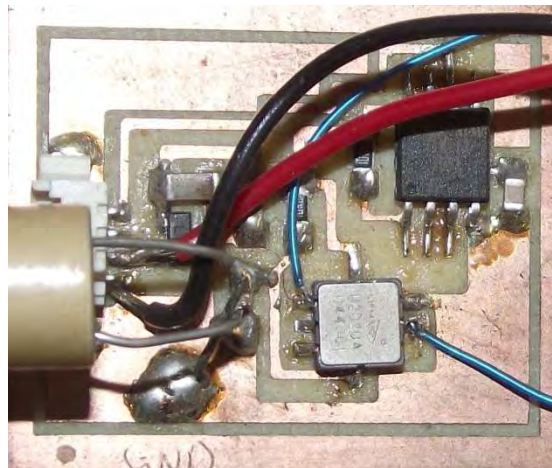


Figure 4.3: MXD2020 Accelerometer on Rapid Prototype board

The MDX2020 device is a two-axis accelerometer which uses a unique patented methodology of measuring the distribution of trapped heated air by means of thermal sensors, the datasheet for this device can be found in the “Sensors” section of the accompanying CD. The device works well but needed amplification before the signal could be measured by the microprocessor's analogue input. A rapid-prototype board was constructed, and is shown in Figure 4.3 , containing the sensor and some analogue circuitry, and this was placed on the platform. Measurements from the accelerometer system during balancing experiments on the vehicle show that the output of this device is very noisy, as the theory and specifications predicted. Figure 4.4 shows the raw data captured from the accelerometer in blue and the red

trace shows the values after a simple low pass filter was applied. Of interest were the periods of rest in the platform so that gyroscope drift could be handled. To achieve this, the filtered data needed to be processed further to obtain the slope of the data, where a zero slope would indicate a period of rest. In this instance, a colleague, Piet Terblanche, contributed by discovering a paper that was published in 1964 on smoothing and differentiation of data using computationally optimised methods [101]. At the time of publication, this paper was relevant as computing power was still at a premium. The paper was useful for this project as the data needed to be processed in real-time without too much processing power overhead. Integer calculations were used throughout since the use of floating point numbers on a non-floating point processor would have negated the gains made by the algorithms.

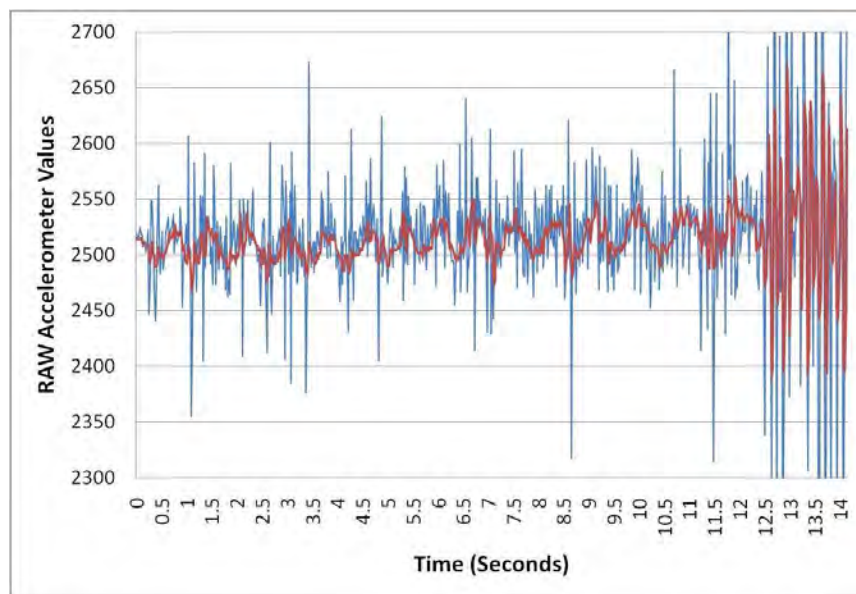


Figure 4.4: Accelerometer data, raw and filtered

After the slope of the data was computed by applying a first derivative algorithm, the standard deviation on that data was computed. Applying a minimum threshold to the standard deviation data then provided a numeric method for determining whether the system was at rest. Figure 4.5 shows the first derivative in blue and standard deviation data in red.

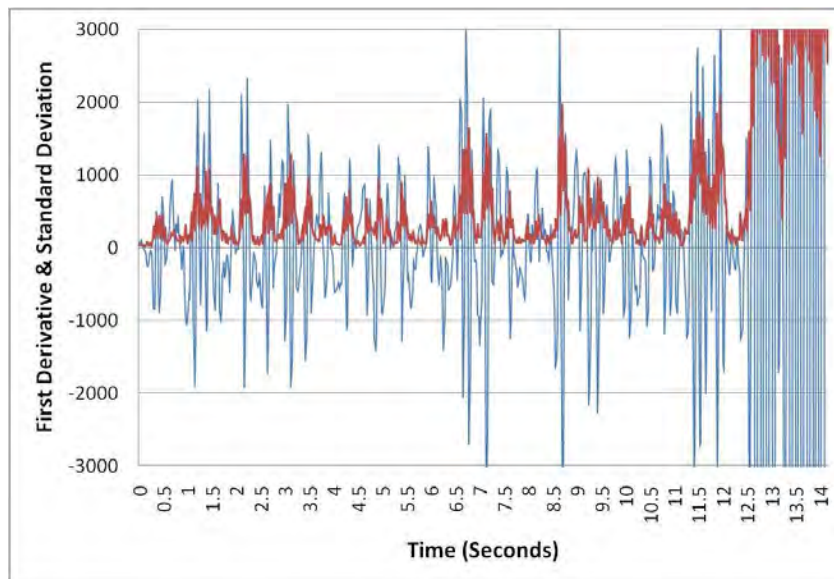


Figure 4.5: Accelerometer data, first derivative and standard deviation

As time progressed new technology became available and a new device, replacing the MD100, was implemented on the platform. The ADIS16350 iSensor device, shown in Figure 4.6, was provided by the manufacturer Analog Devices.



Figure 4.6: Analogue devices: ADIS16350 combined accelerometers and gyroscopes

This device is much more accurate, and is a MEMS technology-based integrated three-axis gyroscope and three-axis accelerometer sensor. It has a fully integrated set of electronics that interfaces to the sensors and gives access to all data via an SPI interface. This interface makes it much simpler and easier to integrate onto a platform, as the device does not require any additional specialised interface electronics. The ADIS16350 performs power management, auto calibration, temperature sensing, which can be used for further calibration, and many more integrated functions, which make it a very versatile device to use.

4.1.3 Gyroscopes

The initial implementation of a gyroscope was the MD100 device from the company Gyration. This is an older technology device which was intended to be used in pointing devices such as computer mice. As the application note did mention, robotics applications and the device were available, and a rapid prototype board was constructed, shown in Figure 4.7, which provided regulated power to the sensor and amplified the weak output for further processing. The output of this device proved to be functional but the drift for this device is very high. The datasheet mentioned a rate of $7.2^\circ/\text{min}$, which in practice was even worse.

Although useful for short-duration measurement of individual turns, longer duration use of this device created too many problems, and the arrival of the ADIS16350 sensor, as discussed in paragraph 4.1.2, was a welcome upgrade to the MD100 sensor.

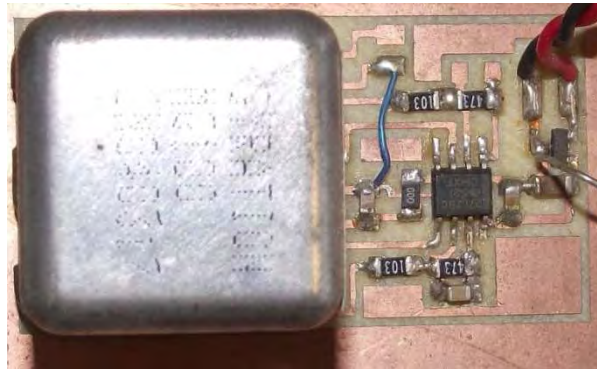


Figure 4.7: MD100 Gyroscope

4.2 Obstacle sensors

In the case of obstacle sensors, the platform has bumper sensors, ultrasonic range sensors, an infrared beam sensor and a laser beam sensor. With these sensors it is possible to detect known landmarks, as well as unexpected obstacles.

4.2.1 Bumper sensors

The bumper sensors are used as a last resort to stop the platform from driving into obstacles. They are usually simple mechanical constructs which actuate an electrical switch, thus signifying contact with an obstacle. The ideal bumper sensor configuration not only informs the robot that a collision has taken place but also in which area of the robot the collision has taken place. The use of a coil of resistance wire, which is shorted during a collision against a metal (grounded) back-plate was a very good idea that operated well on the Friendly Robotics

Lawnmower [55]. The measured resistance indicates where the robot is actually touching an object.

On the Meercat, as shown in Figure 4.8, two bumper sensors are implemented, which are actuated by whiskers which emerge from the vehicle to cover the near area in front of the vehicle. This system works well enough as the robot is most likely to bump into obstacles in this area.

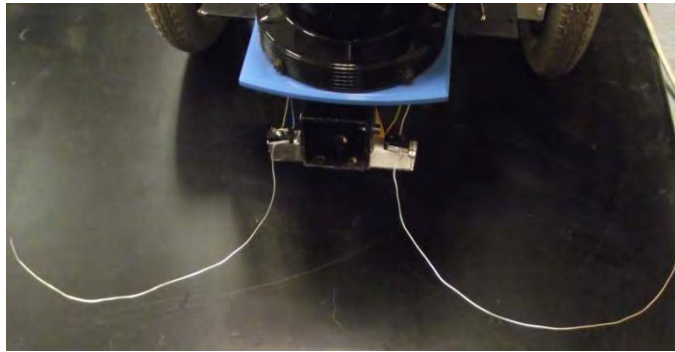


Figure 4.8: Bumper sensors

4.2.2 Ultrasonic range sensors

A single transducer sensor, type MaxSonar-EZ1 from the company Maxbotix [102], was chosen for Meercat because of its low cost and small form factor, as shown in Figure 4.9.



Figure 4.9: MaxSonar-EZ1 ultrasonic sensor

The device features a claimed beam pattern, which was useful for the Meercat, with a maximum range of more than 6 m with the aforementioned “deadzone” of only 15 cm. In literature [51] the beam width is mentioned as a disadvantage, but for the Meercat application it is sufficient to understand in which zone an obstacle resides.

The sensor can be interfaced either by serial communications or by an analogue voltage signal. In the case of Meercat, the analogue signal is measured, where each increase in centimetre distance is represented by a 3.8 mV signal increase.

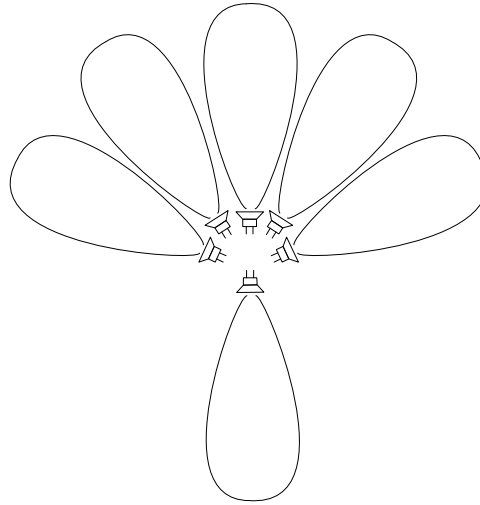


Figure 4.10: Ultrasound sensors placement and beam patterns

Figure 4.10 shows the arrangement of the sensors on the Meercat (not to scale), where the front side receives the most attention, as this is usually the travel direction of the platform. As multiple sensors are used, some form of synchronisation is required to ensure that multiple “pings” are not interfering with each other. The beam pattern shown is only an idealised form, as it is quite possible for the “pings” from one transducer to be received by other transducers. The synchronisation is implemented by wiring the devices’ Start-Conversion / End-Conversion signals together, thus forming a chain of devices. As the analogue outputs of the devices remain the same until a new target is found, the interfacing with the microprocessor was easily achieved by interfacing each device with an individual analogue input channel. A successful wall-following experiment was performed using these sensors, as discussed in paragraph **Error! Reference source not found.**

4.2.3 Infrared beam sensor

An option for a more accurate obstacle sensor is the Hokuyo PBS-03JN as shown in Figure 4.11. The device is well known in industry, with characterisation and comparison tests having been performed by third parties [103]. The datasheet for this device can be found in the “Sensors / IR Range / Datasheets” section of the accompanying CD.



Figure 4.11: Hokuyo PBS-03JN infrared beam sensor

This is a short-range device, which will produce reflection data within a radius of 3 m and output this data with a claimed accuracy of ± 10 cm in ideal situations. Figure 4.12 shows the beam pattern of the device with some indicatory dimensions. According to the manufacturer, factors that affect the accuracy are vibration of the moving platform, light interference (sunlight) and the reflectivity of the target.

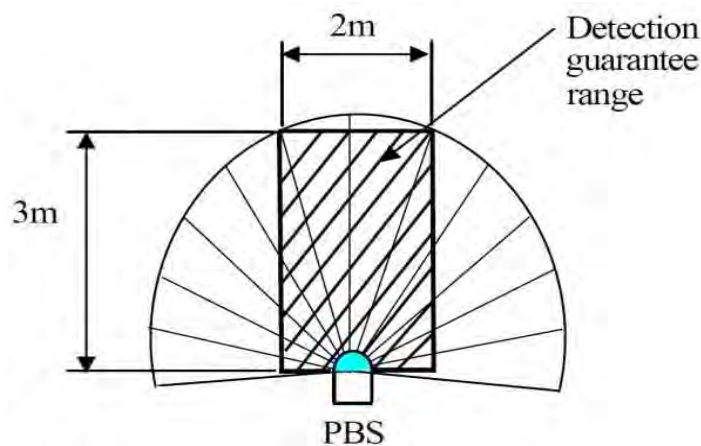


Figure 4.12: Hokuyo PBS-03JN beam pattern

Even though the manufacturer warns against using the sensor in outdoor environments, the cost of the device is low enough to make it an interesting sensor to use because of a potentially more accurate and faster output than the ultrasonic sensors. This device seemed suitable for the Meercat application in at least indoor navigation, and some experiments were performed.

The device features a serial and a digital interface, where the digital interface is a series of signals that indicates the presence of an object in a certain zone of the beam pattern. The serial interface was used on the Meercat, and the data is represented in 121 so-called “bins”, with each bin representing the data for a 1.8° field-of-view, making the total field-of-view nearly 218° .

During wall-following experiments, the sensor proved to be far from ideal. The usual problems with glass doors are known and accepted, but unforeseen effects were also experienced. A round plastic pot plant managed to confuse the sensor completely with a “hole” appearing in the reflection data, causing the robot to drive in the direction of the pot plant. These experiences indicated that the sensor is possibly suited for certain well-controlled environments, but not for the Meercat platform. The sensor was subsequently removed to be replaced by the laser scanner mentioned in paragraph 4.2.4.

4.2.4 Laser beam sensor

The Mechatronics group is fortunate enough to have a sick LMS-200 laser range finder on a long- term loan from the University of Pretoria, as shown in Figure 4.13, which made it possible to motivate for its use on the Meercat platform. These devices are very expensive, with a typical cost of around R40K, and do not actually fit within the low-cost design target. However, the performance of the sensor was interesting enough for it to be used in the application for a number of experiments as it is also envisaged that the cost of these types of devices will decrease in the near future.

The performance of this sensor is configurable in software. An operational range of 8 m results in a 0.5° step resolution with a distance resolution of $10 \text{ mm}/\pm 15 \text{ mm}$, while selecting the full 80 m range results in only 1° step resolution, where the manufacturer is not very clear on the measured distance accuracy. Further data can be found in the data-pack in the LMS200 reference manual.

The range-finder interfaces via either an RS-232 interface or a very high-speed RS485 interface. For the Meercat project the RS-232 interface was chosen, resulting in scanning data being available once every 600 ms, which worked well enough during experiments. The device itself scans much faster than this, but that data is only available on the high-speed port.

Similarly, as with the previously mentioned Infrared beam sensor, the manufacturer claims this specific sensor to be for indoor use only, but interestingly enough the outdoor experiments showed no noticeable problems for the sensor, even in full midday sun. The sensor proved to be very well behaved and sturdy. It even survived an unexpected crash when it was driven through a plaster wall.



Figure 4.13: SICK LMS200 laser range finder

In conclusion, this sensor lived up to its promise of fast and accurate data and is deemed to be a very good addition to the Meercat platform. The solution is not low cost as required, however it is only a matter of time before it becomes available at a lower cost. Refer to the discussion at the end of paragraph 2.4.4 for additional details.

4.3 World position sensors

An indoor beaconing system was not implemented on the system as the area to be covered was too wide. A compass sensor was implemented, which could potentially work for that environment., Initially, a standard GPS sensor was used for position determination in an outdoor environment, but at a later stage a very low-cost differential GPS sensor was also used.

4.3.1 Compass

A magnetic sensor, type number HMC6343, from the company Honeywell was implemented on the Meercat robot. This sensor is able to detect magnetic fields along all three axes: X, Y and Z, and reports the orientation of the sensor in relation to the detected magnetic field.

Interfacing to this sensor is achieved via a standard I2C interface, making it easy to obtain the sensor data without the use of specialised analogue electronics. Two methods are available for the calibration of the sensor, the first method uses an on-board calibration procedure, which the IC provides, and the second is a manual calibration method.

The IC's calibration method uses a "Start Calibration" command followed by rotating the sensor along all its axes, followed by an "End Calibration" command. This method expects a calibration along the Z-axis, but this is not a viable option considering the weight of the Meercat vehicle.

Fortunately, besides providing header information, the IC also provides access to its primary magnetometer sensors. Each magnetometer reports the sensed field strength in micro Teslas (μT) and a resultant total vector can then be computed. A calibration scheme is now used where the sensor's X and Y magnetometer data is taken, normalised and interpreted into a heading. The raw data obtained from the calibration process is shown in Figure 4.14 where the vehicle was slowly rotated around its axis and the data was sampled at an interval of 5 Hz.

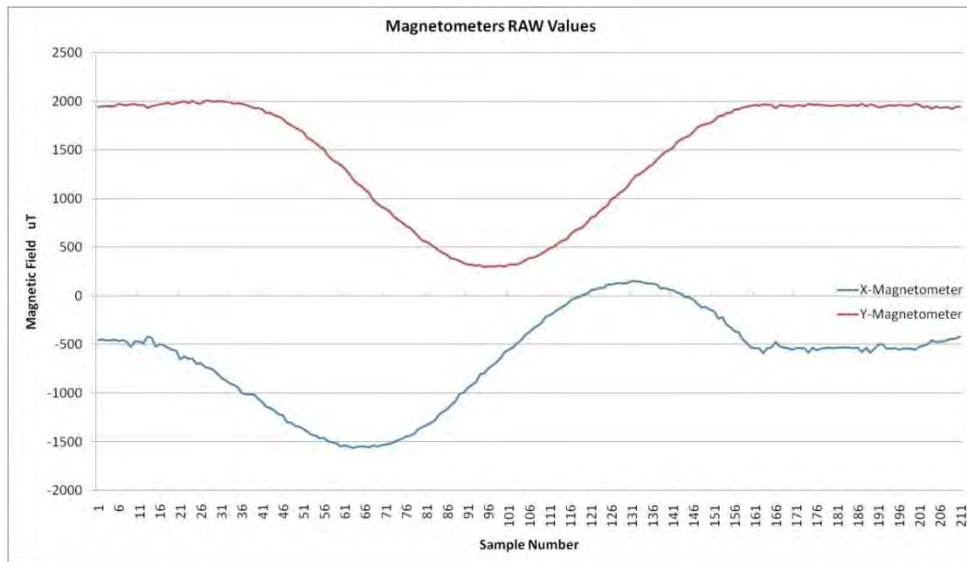


Figure 4.14: Magnetometers raw values

As these are raw values from uncalibrated magnetometer sensors, the values must be normalised to each other so that they can be used in further calculations. This is done by finding the extreme values in both sensors and adjusting the readings such that the data fits symmetrically around the zero-axis as shown in Figure 4.15.

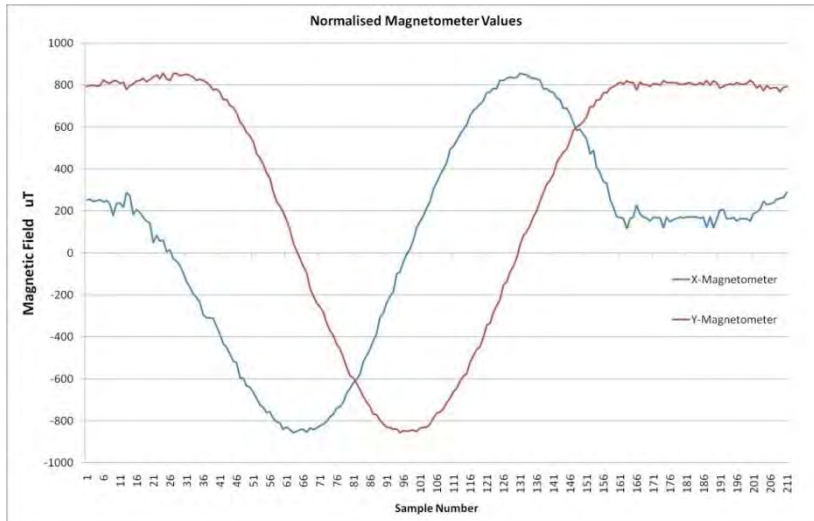


Figure 4.15: Normalised magnetometer values

After further calculations, the final heading was computed and is shown in Figure 4.16. The data is shown as radians ranging from $-\pi$ to $+\pi$ as this is the platform orientation standard used in the higher levels in the software.

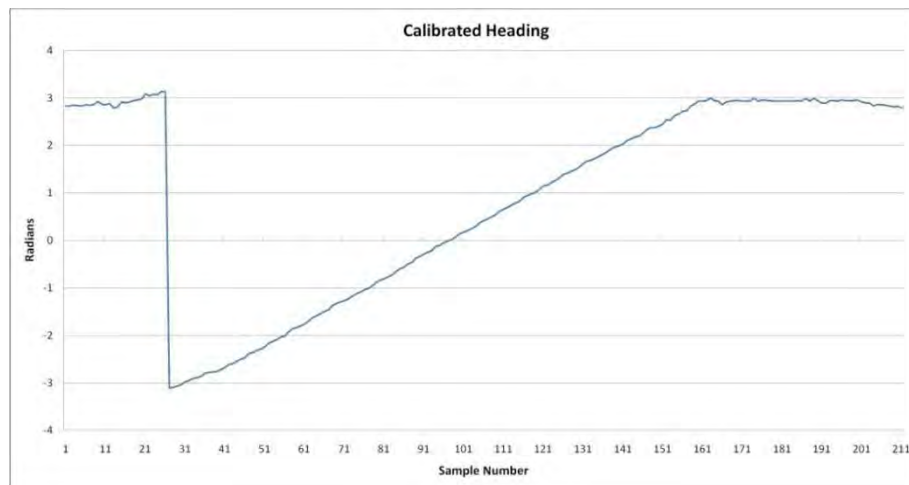


Figure 4.16: Compass heading

The Z-axis data only plays a role when the vehicle should experience tilt and this is not the case in an indoor environment where the tests took place.

4.3.2 GPS

The EVK-5H evaluation kit was chosen from the manufacturer u-blox for the GPS sensor, as shown in Figure 4.17. The kit is equipped with a USB link for direct interface to a computer. The alternative RS-232 interface was used for the Meercat as the GPS was interfaced to the AVR32 embedded platform.



Figure 4.17: u-blox EVK-5H GPS evaluation kit

The kit comes with an active GPS antenna, which means that the weak GPS signal is additionally amplified before it enters the receiver. The manufacturer claims that its SuperSense feature allows for indoor GPS as the module can receive signals down to -161 dBm. However, based on the tests performed, this seemed to be a little optimistic as acquisition of satellite signals in the office's corridor was never successful.

By moving close to a window or open door, a satellite lock was obtained and subsequently the module was very successful at maintaining at least a time signal indoors, although position information was not reliable. Generally, the solution worked well in outdoor situations, with a very good positional accuracy. As expected, some problems were experienced when GPS signals were obstructed by trees and buildings. The application of a metal base plate underneath the GPS antenna made a considerable difference to the sensitivity.

4.3.3 Differential GPS

Traditionally, a specialised receiver is required for differential or augmented GPS, which handles both the GPS signals and the augmentation service's signal. Several routes were explored, but the solutions were all very expensive. The lowest quoted price started at R70K, which is clearly not a feasible solution for a low-cost service robot.

Two solutions were subsequently considered:

- RTKLIB by the Tokyo University of Marine Science and Technology [104].
- The Trignet service by the South African Department of Land Affairs [105].

RTKLIB is an open source software package, which can be used to calculate precise positions using a variety of Global Navigation Satellite Systems (GNSS) sources. In a recent paper [106] the authors of this software describe an implementation of a full working real-time positioning

system using RTKLIB. Trignet is a service recently launched by the South African Department of Land Affairs. This service provides data which is compatible with the RTKLIB software package.

The crux of any augmentation service is that it needs to be physically close to the point of interest, as the atmospheric corrections are only relevant within a limited range. As shown in **Error! Reference source not found.** the Trignet stations are spread over the entire country, with a higher density around Cape Town, Durban and Gauteng. One of the Trignet reference stations is located in the Pretoria suburb of Silverton, only a few kilometres from the CSIR, making the available correction data highly relevant. With an internet connection through a standard GPRS mobile phone, information was obtained from the reference network.

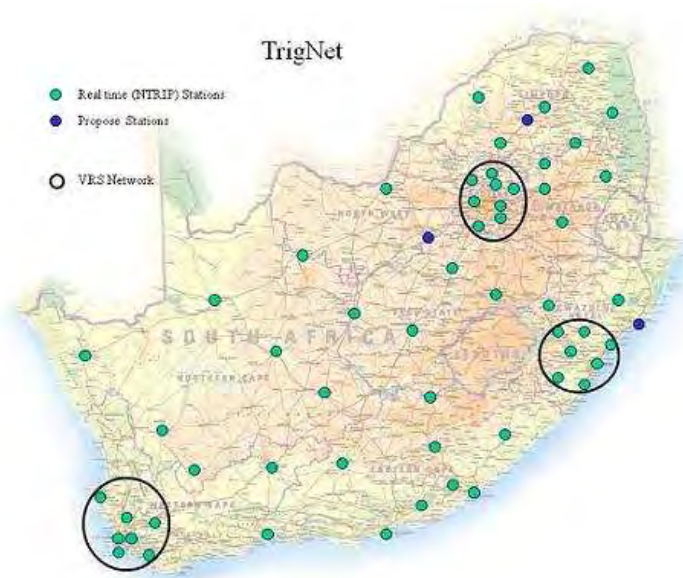


Figure 4.18: Trignet stations map

Unfortunately, the EVL-5H GPS module that was used for the GPS experiments was not suitable for the envisaged solution as it does not generate the correct raw data stream. A new evaluation kit, UBlox LEA 4L, was purchased at a cost of less than R3 000, making this the total cost of the implemented solution as the other elements were free.

With this solution, according to the Division of Geomatics at the University of Cape Town, it is generally possible to obtain RMS accuracies of 35cm in a horizontal position and 1 m accuracy in height [107].

4.4 Summary

A number of sensors performing a wide variety of sensing were implemented and evaluated on the Meercat mobile robot. During the evolution of the robot some sensors were discarded and relevant reasons were given for upgrading or discarding them. For certain sensors, some basic processing and calibration results have been shown. The Meercat platform now uses a wide variety of sensors, which enable it to sense both in indoor and outdoor environments.

5 Electronic Design

This chapter focuses on the design and integration of all the electronic systems implemented on the Meercat. Although the chapter on sensors also deals with electronic components and their interfaces, this chapter deals with all non-sensor electronics, which comprises a significant amount of hardware.

Almost all of the electronics is mounted in a single bay on one level of the Meercat platform, and is easily accessible. All the hardware is mounted on a slide-out perspex section as shown in detail in Figure 5.1. The electronics tray contains the Generic CPU board and the EVK1100 AVR32 UC3A evaluation board, details of which are described in this chapter, as well as the Ultrasonic range sensors, GPS unit and Compass which were described in Chapter 4.

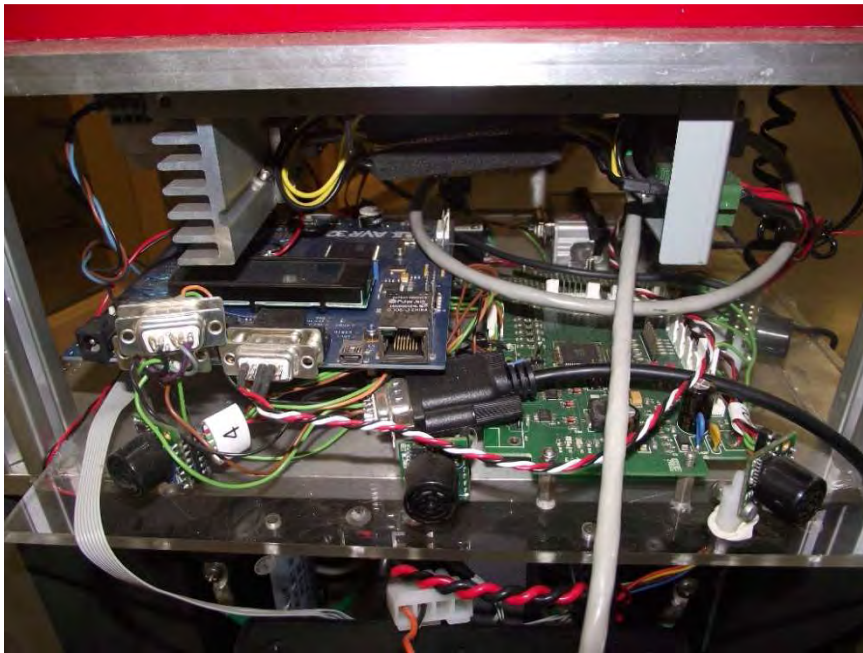


Figure 5.1: Electronics tray details

The chapter starts with the lowest level electronics, which is a fairly simple input/output circuit board, through a higher level controller based on a 32-bit processor leading up to the on-board industrial PC. This is followed by a discussion of the radio interfaces, both remote control and Wifi. The last few sections focus on the battery system, power distribution and motor controller hardware, and provide a conclusion on achievements.

For reference purposes, the entire system and the interconnections are shown in Appendix A: Meercat Electrical System. Electrical schematics and datasheets of the individual items can be found in the data-pack.

5.1 I/O board: Generic CPU

An in-house printed circuit board called Generic CPU was designed for this project to handle the interfacing with basic sensors. The board was developed during this project as a generic solution for robotics and other projects in the Mechatronics group, and has been used in many different projects.

Although other commercially available solutions can also be found, the reasons for creating an in-house design were:

- high purchasing and shipping costs of imported solutions.
- limited functionality of available solutions, requiring additional hardware components to achieve the required solution.

The Generic CPU circuit board was used on Meercat, shown in partially assembled form in Figure 5.2, for basic interfacing of some of the sensor hardware, specifically the ultrasonic sensors, bumper sensors and the remote radio controlled interface. It can be seen as an extension in input / output functionality of the AVR32 board, as discussed later.

The board has an on-board switch-mode regulator which is capable of a wide input voltage range, which makes the design forgiving and reliable. The CPU on the board is an ATMEL [108] supplied processor called the AVR MEGA128 executing at 11.0592 MHz, so chosen to attain correct baud rates for the UARTs. This processor is a good choice to use for various jobs, as it integrates a sizable amount of Flash (128Kbyte), E2PROM and SRAM memory together with a varied set of peripherals such as timers, ADC and PWM, asynchronous serial ports, SPI and I2C interfaces. A full schematic of this board can be found in Appendix D.

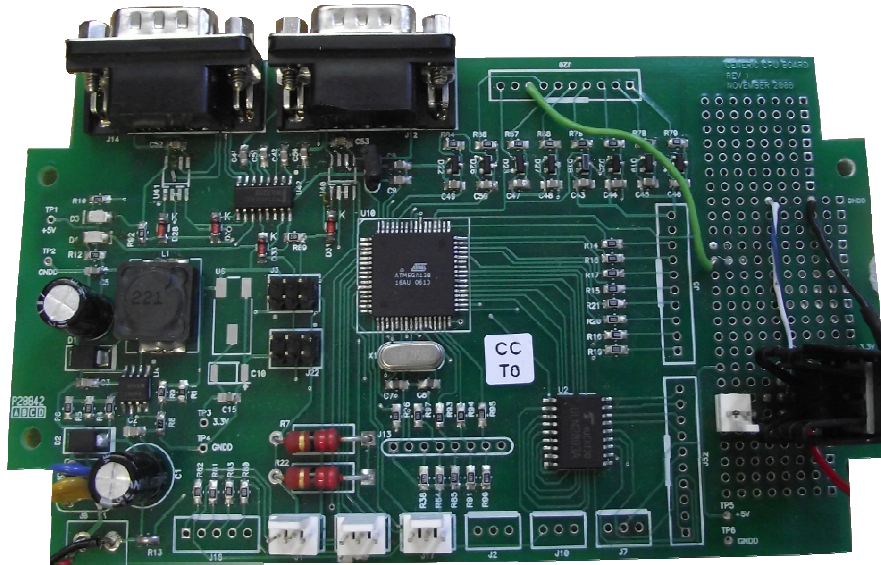


Figure 5.2: Generic CPU circuit board

The design philosophy for this board is to make available as many useful inputs and outputs as possible. A list of its capabilities include:

- Power supply capable of powering external 5V hardware
- Two serial ports, each of which accept either RS232 or RS485 connections
- I2C and SPI ports
- Eight analogue-to-digital inputs, 10 bit resolution
- Eight digital inputs for logic level signals
- Eight digital outputs, capable of driving relays
- Six PWM outputs, wired to interface directly to servos
- Breadboard area for quick functionality expansion
- Optional liquid crystal display connection.

5.2 EVK1100 AVR32 UC3A evaluation board

As the Meercat has numerous sensors on-board, and several processing tasks have to take place to coordinate all the sensor data, it was clear that the Generic CPU mentioned in paragraph 5.1 would not be enough for all the interface and processing needs. It was decided to employ a more powerful processor to meet the needs of the application, and a search ensued for a suitable solution. This decision was also partially fuelled by curiosity and the need to move on from simpler 8-bit processors to more powerful solutions so as to learn more about these systems.

There is a definite world-wide trend to employ 32-bit microcontrollers in embedded applications, mainly because it has become easier to do so. The processors are becoming more like the 8-bit processors in that they integrate memory and peripherals in a single IC. In

addition, they are provided at lower cost. In numerous cases, the spare horsepower means that development can be done using high-level programming languages and software frameworks without having to worry too much about whether a design is fast enough.

Some research was done with the available options in an attempt to find a solution that would work well, not only for the Meercat but also for future needs. Research was done on the STM32 processor by manufacturer STMicroelectronics, the Stellaris from Texas Instruments (previously Luminary Micro) and Microchip's PIC32 family. It becomes difficult to choose between solutions, especially since the ARM architecture seems to be a very popular solution with many developers. Ultimately, the AVR32 was chosen and the reasons for this are as follows:

- A free software framework is made available
- There is a free and very powerful development IDE
- A number of good debugging tools are included
- The manufacturer is very quick to respond to on hardware or software problems, typically a one- day turn-around (first-hand experience)
- A very good relationship was forged with the local supplier of the ATMEL product, who provided free samples, quick responses and the like.

The latter was a deciding factor since the ATMEL South African office resided on the CSIR campus for a while, which made it easy to walk in to get help, samples or simply discuss various aspects. A solution was found in ATMEL's EVK1100 board, shown in Figure 5.3, which is an evaluation board for the AVR32 UC3A series of processors. The processor itself, called the AVR32-UC3A, has significant memory resources in 512 Kbyte of flash memory and 64 Kbyte of SRAM memory, and includes numerous peripherals such as Ethernet, USB, USARTS, PWM, ADC, and much more. The interesting fact is that this processor is sold at a lower cost than the 8-bit processor on the Generic CPU Board. The processor is set to execute at its maximum speed of 66 MHz, and most of the I/O capability on the EVK1100 board was used.



Figure 5.3: ATMEL EVK1100 board

The EVK board was chosen over a custom developed product as this would have cost a significant amount of time and money, and since the only application at that stage was on the Meercat platform. A full schematic of this board can be found in Appendix E of this thesis document. The choice of working on the cutting edge did cause some problems during the development phase. The EVK1100 board was fitted with an engineering sample processor which had a number of errors, resulting in strange occurrences whilst testing the software. Numerous problems were experienced during software testing and implementation, which were eventually solved when the processor was replaced by a later revision.

The interfacing to all sensors and peripherals is not a very neat solution, although an attempt was made to make the system more accessible by interfacing everything through pluggable connectors. However, the end solution is not very satisfactory. Nevertheless, once the initial problems were solved, the development and stability on this system worked very well.

5.3 Embedded personal computer

Higher levels of processing are needed when the Meercat platform has to navigate autonomously, and for this reason a single board computer (SBC) was fitted onto the platform. The design philosophy here was to leave the interfacing of the sensors and real-time control of other hardware to the embedded processing platforms and perform the higher level processing and decision-making on the embedded PC.

The platform is shown in Figure 5.4 in a sturdy industrial enclosure, and its form-factor is a good fit for positioning it in the Meercat's electronics bay.



Figure 5.4: Single board computer in enclosure

The advantage of an SBC is that it is capable of running from a DC power source, and this specific model is able to operate from 9 to 36 V DC input. The SBC is connected to the 24 V power rail, as sufficient power is available from that source. To save power, passive cooling is implemented where a heat-pipe connects the processor to a heat-sink on the outside of the enclosure. As the platform is subject to vibrations whilst driving, rather than using a mechanical magnetic hard-drive, a solid state hard-drive was chosen to ensure reliable operation. The disadvantage of this solution is a limited amount of space compared to other drive solutions. Only 4 GByte is available for the operating system and the application software, but this proved to be sufficient.

The system operates using a low-power 1 GHz VIA Luke processor and 1 GByte of memory. To communicate with the SBC, its Ethernet port is interfaced to the WiFi router.

5.4 Radio remote control interface

As an experimental autonomous platform has the potential to become an out of control vehicle, it was decided to implement as many safety measures on the platform as possible. One of these safety features is implemented through the use of a radio remote control interface. The Meercat platform implements a multi-level control strategy via a remote control device. Whilst experiments in navigation take place on the platform, the thinking is that the robot is initially controlled via a human operator and that autonomous operation is only selected from the remote console. Autonomous operation is stopped as soon as either the operator de-selects this mode, the radio is switched off or the radio link is lost. This makes for a very safe operational environment as any mishaps are easily managed.

Initially a standard R/C interface was used, but this caused erratic behaviour when the radio's signal strength became weaker. The typical servo signals were interpreted on the General CPU

board, but these became jittery as soon as the vehicle was removed from the transmitter at a distance greater than 30 m, possibly caused by the quality of the R/C Radio as this as a very basic device. Attempts by the software to filter and manage this situation were not successful.

Due to these problems, it was decided to implement an improved solution using available components. A project in the Mechatronics group known as Smartfactory already integrates a high-powered but small and power-efficient radio module called AC4868 supplied by the company Aerocomm. This radio module implements a wireless networking solution using the free 868MHZ band, and has a claimed range of 15 km (using a specialised test rig). This solution was adapted for use as a remote control, the result of which can be seen in Figure 5.5 on the left-hand side. The software, written for an AVR processor, fits on a small interface board which currently caters for 6 input channels transmitted to the receiver. More channels can easily and simply be changed should additional capabilities be required. As the radio traffic is now performed via a communication protocol and is checksum protected, invalid communication is a thing of the past.

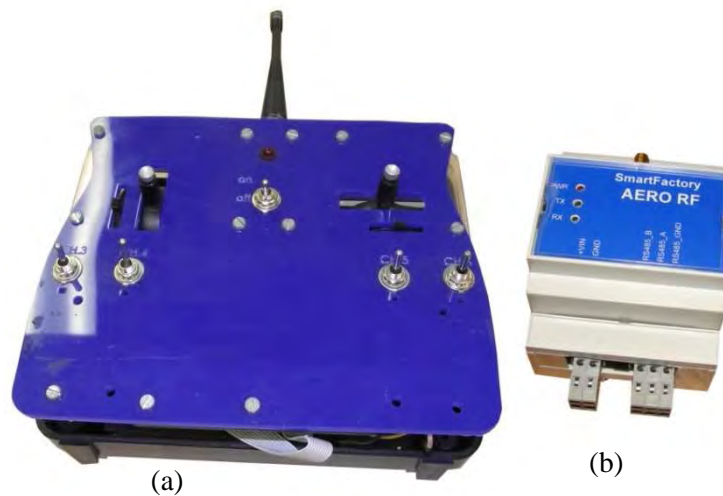


Figure 5.5: Custom remote control (a) and Meercat receiver (b)

The receiver module is a standard Smartfactory RF module powered by 24 V and is shown in Figure 5.5 on the right-hand side. It re-transmits any received radio data out of an isolated RS-485 port to the Generic CPU board (Paragraph 5.1).

5.5 3COM WiFi router

For ease of interfacing with a moving platform whilst performing autonomous operations, the best method is to use the industry standard high-bandwidth WiFi interface. The platform

implements 802.11b which caters for a maximum 54M bits/s bandwidth, which is more than enough for the demands on passing commands and data.

Some major problems were experienced initially when attempting to use a USB WiFi dongle. This solution seemed simple, elegant and low-power, but the Ubuntu operating system showed numerous problems. The idea was to create an ad hoc WiFi link between the Meercat's embedded computer and a standard laptop, but despite multiple days of intense attempts to make this work properly, it was never successful. Besides basic problems such as recognising the WiFi dongle and connecting on a network, another major problem turned out to be re-establishing a connection after it was lost.

In the end, it was decided to implement a WiFi router as this is the better architecture to handle issues such as connection fail and resume; these are all handled transparently and reliably via the router.



Figure 5.6: 3COM OfficeConnect wireless router

The 3COM Officeconnect router, shown in Figure 5.6, was used because they were available in our offices at the time and the power requirements were compatible with the available sources on the platform.

5.6 MotorController: AX2550

The motors used on the Meercat are powerful and require high amperage to activate. Because of their mechanical construction they generate high amperage when braking. To ensure reliable operation of the vehicle, a high-capacity standard motor controller, type AX2550, was chosen from the company RoboteQ [109]. This controller, shown in Figure 5.7, is capable of driving two motors at the same time with up to 120 A of current. It is interfaced via a standard RS-232 interface, from which it receives control commands from the AVR32 controller.

Although this particular controller has the ability to perform closed-loop control of position or velocity, in the Meercat system this is not implemented as control is preferred at a higher level of operation.



Figure 5.7: RoboteQ AX-2550 motor controller

Another benefit of choosing this motor controller is the fact that it implements regenerative operation, charging the batteries when the vehicle is slowed down. No other braking system is implemented on the robot as the electronic braking solution is sufficient for both dynamic situations as well as static situations. The combination of the gearbox and the electronic brake applied by the controller is sufficient to keep the vehicle in place even on steep inclines.

This controller has been a reliable driver of the motors, and no problems were experienced with it. In the case of Meercat, the controller drives the motors using a 24Volt supply, and operates from this same 24V supply. Using a serial port interface, the controller interprets motion control commands from a control program. These commands are translated into a proportional PWM value which is applied to the internal FET's in an H-Bridge configuration which controls the motor direction and speed.

5.7 Human feedback

Although human feedback was not a major focus of this research project, the platform was equipped with some systems to make people aware of the progression of an autonomous robot. A flashing light and a flag were positioned on top of the vehicle. This was a useful feature outdoors to make other road users aware of the moving system. Another human feedback mechanism is the two USB coupled speakers through which the system can announce its intentions using a voice interface or alert sounds.

The free Festival [110] text-to-speech program was used for voice generation. Some Afrikaans phrases were programmed with some difficulty, which was the source of great hilarity. These were used during the filming of an episode of Kyknet's reality show "Die Foon". Playing the "Ice-Cream-Truck" tune whilst the robot was navigating also attracted much attention.

5.8 Power system

As the Meerkat is a mobile untethered system, some form of power needs to be available on the platform. To power the motors, an embedded PC and the laser scanner, two standard 12 V / 36 Ah batteries are switched in series to achieve 24 V. A separate system, using a lower capacity 12 V / 7 Ah battery, powers the two embedded controller boards and the various smaller sensors. The main reason for implementing separate electrical systems was to ensure that the electronics would not be affected by electrical noise from the motor and motor controller.

Normal double-pole double-throw circuit breakers are used to isolate the entire system from power whilst the batteries are charged or when the system does not need to be active.

At a later stage, the robot can be made more autonomous by adding a docking and charging station. Initial plans for such a station were abandoned as it was found that the lifespan of the batteries was sufficient for a day's operation. Charging therefore is typically done manually on an overnight basis using external battery chargers, which connect to two charging power connectors mounted on the robot.

Item	Current (Amperes)	Usage Frequency (%)
12 V System	0.33	100
24 V System at rest	1.6	30
24 V System moving slowly (2-3 km/h)	2.5	50
24 V System moving fast (5 km/h)	3.5	20

Table 5.1: Meercat power usage

Power usage measurements were taken on the system, as shown in Table 5.1, to confirm the operational time between recharges. On the 12 V system rail, for a current of 0.33 Ah, the power requirements are easily met by the 12 V/ 7 Ah battery. On the 24 V system a differentiation was made between vehicle at rest, navigating slowly and moving faster. The table also adds an optimistic estimation on vehicle usage, and this then gives a result of $1.6\text{Ah} * 0.3 + 2.5\text{Ah} * 0.5 + 3.5\text{Ah} * 0.2 = 2.45\text{Ah}$ for the total average consumption. As the battery pack is rated at 36 Ah the 8 hour target is easily met, with the pack discharging to approximately 45% capacity, which is acceptable. Note that the calculated usage pattern is an extreme usage assumption, and it is envisaged that the average use of the Meercat robot will be less than this.

In addition, the regenerative nature of the motor drive should also increase the range of the system. Tests showed that under heavy braking, reverse currents of up to 2 A were pushed back into the batteries.

5.9 Summary

This chapter showed and discussed the electronic design steps taken in implementing the Meercat robot. A complex system of hardware is implemented on the Meercat robot and a system of interconnections tie the electronics together such that the whole set is able to operate the platform. The interfaces between all hardware were discussed in detail in the chapter. It was discussed how power was distributed to all hardware and converted to the correct voltage levels according to hardware requirements.

Some calculations were made on power usage and the outcomes show that the electronic design meets the requirements that were set at the beginning of the project.

6 Software Design

This chapter discusses the software that was used and developed for the Meercat robot. It is clear that software forms the biggest effort in the development of a mobile robot, and structuring this software in a proper and logical form is very important. Each of these levels was created with different tools and development environments, making the whole of the robot a diverse environment.

The Meercat robot consists of several levels of software, shown in a basic diagram in Figure 6.1. On the lowest level there is the software which resides in some of the sensors and the sensor interface software on the Generic CPU board. On a next level the central AVR32 controller operates this layer, and can be regarded as the central coordinator of the robot. This application is the core of the entire Meercat system as it determines whether and how the robot senses and moves. The highest level of processing is performed on the embedded PC, where a combination of a client, several algorithms and the player robotic server attempts to coordinate the entire system to make it achieve its goals.

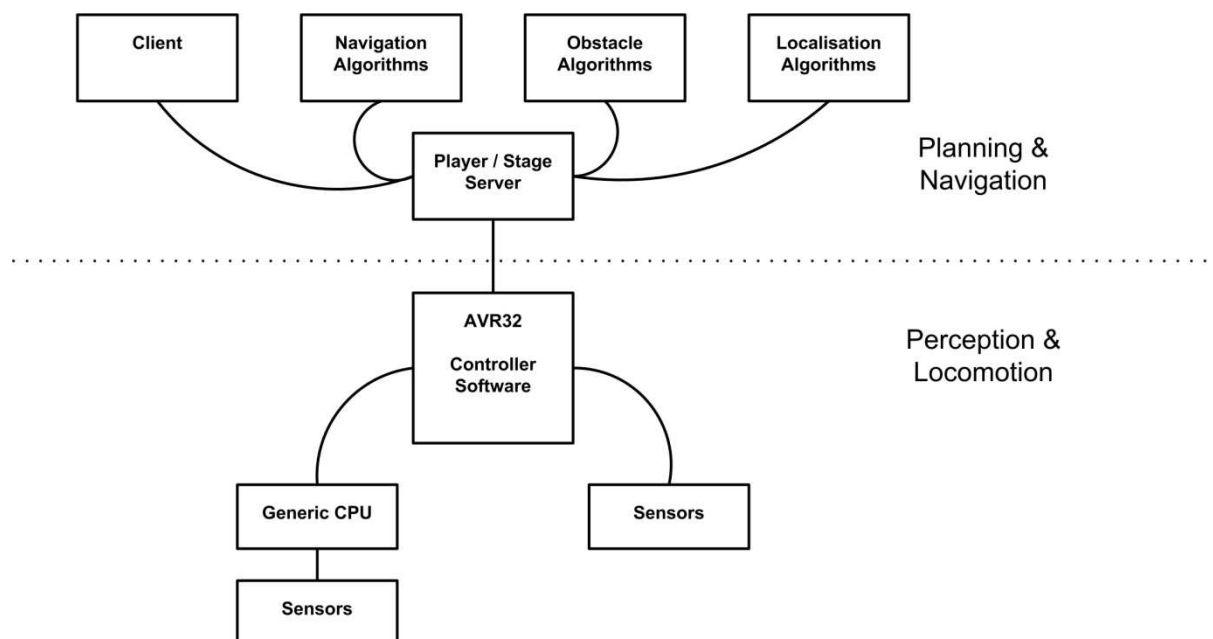


Figure 6.1: Meercat high-level software structure

6.1 Generic CPU software

This software was created using the ‘C’ programming language and the WinAVR [111] tool-chain. It is a set of open source software development tools for the AVR microprocessor, which includes the GNU GCC compiler. The microprocessor on the Generic CPU board is programmed using a programmer tool known as the ATMEL ISP MKII, which is a programming device linked to the development machine via USB.

The structure of the software implementation is known as “Cooperative Multitasking”, which is a complex name for a big while-forever-loop within which various processes are executed. The only rule for each process is that it should not take up too much time and it must release the processor whenever possible. This typically results in software that is written in state machine type structures where each process performs a little step and exits until the next step can be done. Interrupts handle system events in the background and decision-making occurs in the foreground. The software that was written for this specific project implements functions such as interrupt-based serial port drivers, analogue-to-digital conversion routines, hardware timer based functions, communication protocol and other features. The files that were created for the software on the Generic CPU are described in detail in Appendix F of this document.

In order to pass the sensor data from this board to the next controller, a simple serial line interface protocol (SLIP) [112] was initially used with an added longitudinal redundancy check byte to ensure correct reception of the data. The Generic CPU simply transmitted the sensor data on a very regular 10 Hz repeat basis, and invalid received packets of data could be discarded until the next valid packet was received. This protocol was changed to a polled master-slave protocol which enabled additional capabilities on the Generic CPU, where the AVR32 CPU acts as the master device. The new implementation of the protocol is now defined as follows, where the master device sends:

GXXYY<Cr>

in which:

- G is the message start identifier
- XX is the hexadecimal value of the outputs.
- YY is the hexadecimal representation of the XOR-CRC value of the message.
- <Cr> is the Carriage Return character

The slave device then responds to this message with:

```
P111122223333444455556666R1R2ZZYY<Cr>
```

in which:

- P is the message start identifier
- 1111 etc. is the hexadecimal value representation of the analogue channels (1-6) in 4 digits
- R1 is radio channel 1, hexadecimal value in 10 μ s resolution
- R2 is radio channel 2, hexadecimal value in 10 μ s resolution
- ZZ is the hexadecimal representation of the logic inputs
- YY is the hexadecimal representation of the XOR-CRC value of the message.

Any found CRC errors in the master message causes the slave to ignore the polling message. Similarly any CRC errors in the slave message causes the master to ignore the reported data.

6.2 AVR32 software

The processor is supported by a significant amount of standard software drivers, professional integrated development environment (IDE), debugging tools and also a real-time operating system. The AVR32 Studio IDE is based on Eclipse and is the best programming environment the author has ever witnessed, with numerous well thought out features supporting the developer. All of these software tools are provided for free, and are actively being maintained and developed.

In this section the software that was written for the AVR32 processor on the EVK1100 board is discussed, an overview diagram is shown in Figure 6.2 and some of the relevant individual items are discussed in the sub-sections.

The Meercat AVR32 application collates the data from all sensors to control the motors based on inputs from either a remote control or from the embedded PC. The structure of the Meercat application on the AVR32 is divided up into eight tasks, where each task performs a unique function as follows:

- The myFlash task is a very simple blinking LED task, showing that the system is actually running. This was very useful in the beginning phases of the project when the stability of the system was not that good.

- The Report task provides good feedback whilst the system is running, while the LCD and keyboard combination shows several screens of information. In this way, the screen of interest can be selected whilst testing certain functionalities of the system.
- The Measure task interfaces to the accelerometer and gyroscopes, collects this data and provides filtering functions.
- The Generic CPU task handles the communications with the generic CPU board and passes this on into the system.
- The GPS task interfaces to the GPS unit via a serial port and to the compass via an I2C interface; the data is collected, interpreted and stored in the relevant structures.

The following sections discuss in some additional detail the Control and MODBUS tasks.

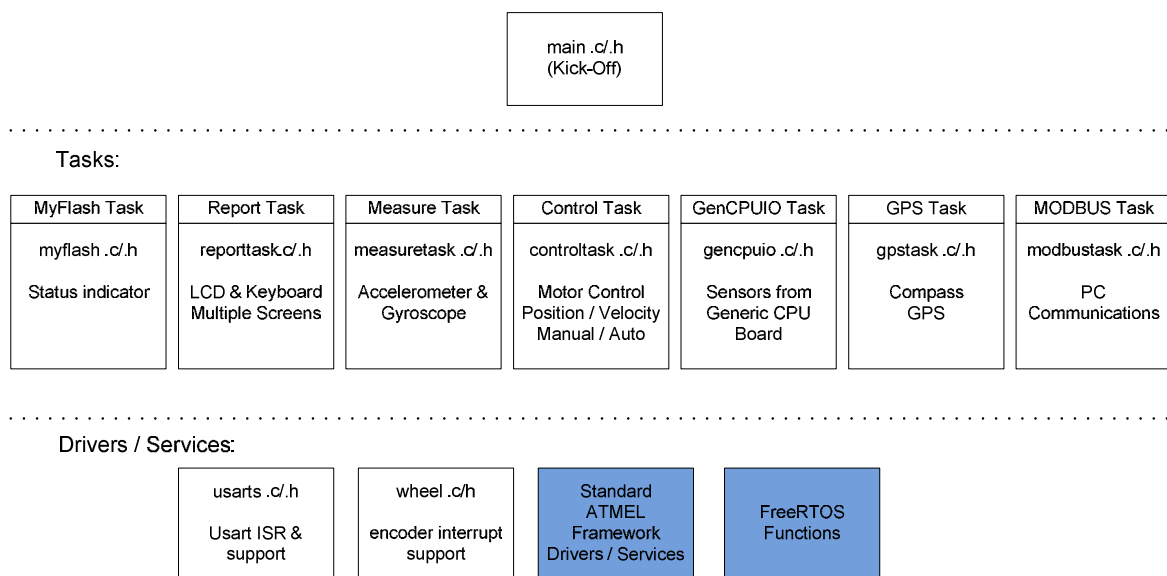


Figure 6.2: AVR32 software

Further details on these software files can be found in Appendix G of this document.

6.2.1 FreeRTOS and software framework

For more complex software projects, the use of an operating system can be of great assistance. It gives the programmer a base to work from, allowing for a concentrated effort at solving the key issues, rather than focusing on making the entire system work. Because the operating system works with tasks, the software developer is virtually forced to compartmentalise the code in individual functional blocks. This then allows for easy code re-use, which is important when trying to optimise development across multiple developers and projects.

The FreeRTOS realtime operating system [113] was ported to the AVR32 processor by ATMEL engineers and is provided for free.

A standardised driver framework is provided by ATMEL. When a project is initiated, the developer can simply make choices as to which drivers are required and these are automatically added to the project. Although all of these features are convenient, the initial impression is one of an overwhelming overhead on the project. However, this becomes less of an issue considering that the processor has enough speed to cope with software structures that are potentially not quite optimal. The USART drivers did not cater for the required functionality of background operation and a custom interrupt based version was implemented.

6.2.2 Control task

The control task is the biggest entity in the Meercat project on the AVR32 controller. It implements an extensive state machine which monitors the state of the radio interface and acts on the commands received via the serial interface from the embedded PC.

Three significant states are possible:

- Robot is in dormant state
- Robot is in remote radio control mode
- Robot operates autonomously.

Part of the control task implements two independent PID controllers, which can be used to control the left and right wheel speeds, taking input from the wheel encoders and managing the motor controller.

6.2.3 MODBUS RTU protocol task

In contrast to the Generic CPU, communications between the embedded PC and the EVK1100 have to be bi-directional, as data has to flow both ways. Rather than developing a completely new protocol with custom commands, a better solution was deemed to be the implementation of an industry standard protocol known as MODBUS RTU [114]. This protocol is an open standard and is widely used and supported in industrial environments.

The protocol implements a set of registers which can be read and/or written by a host computer. This principle makes it easy to expand or change the available data without having to add extra commands or other features - the protocol always remains the same. Once the protocol is implemented correctly in the software, a map of registers then allows the passing of data. As all

the sensor data and commands in the project are contained in single structure, the mapping against register addresses was easily achieved.

To assist with this, a handy Excel spreadsheet was developed where names of variables are entered in a sequential list. Figure 6.3 shows a snapshot of this table. The spreadsheet has built-in macros, written in Visual Basic, which can then convert the table entries into a 'C' language file, where it is directly imported into the EVK1100 software and compiled. In this way, development can occur rapidly, new data/variables can easily be added to the table and the host only has to query for the extra available register spaces.

Modbus Address	Hex Modbus Address	Bytes	Description	R/W	Var Type	Memory Type	Address	Detail
40001	0000	1	Sonar 1	R	Short	RAM	vars.GenCPU.sonars[0]	in inches
40002	0001	1	Sonar 2	R	Short	RAM	vars.GenCPU.sonars[1]	
40003	0002	1	Sonar 3	R	Short	RAM	vars.GenCPU.sonars[2]	
40004	0003	1	Sonar 4	R	Short	RAM	vars.GenCPU.sonars[3]	
40005	0004	1	Sonar 5	R	Short	RAM	vars.GenCPU.sonars[4]	
40006	0005	1	Sonar 6	R	Short	RAM	vars.GenCPU.sonars[5]	
40007	0006	1	Digital Outputs	R	Byte	RAM	vars.GenCPU.outputs	
40008	0007	1	Digital Inputs	R	Byte	RAM	vars.GenCPU.inputs	Bumpers
40009	0008	1	Radio Status	R	Byte	RAM	vars.GenCPU.radiostatus	
40010	0009	1	Radio Channel 1	R	Byte	RAM	vars.GenCPU.radiochannels[0]	
40011	000A	1	Radio Channel 2	R	Byte	RAM	vars.GenCPU.radiochannels[1]	
40012	000B	1	Radio Channel 3	R	Byte	RAM	vars.GenCPU.radiochannels[2]	
40013	000C	1	Radio Channel 4	R	Byte	RAM	vars.GenCPU.radiochannels[3]	
40014	000D	1	Radio Channel 5	R	Byte	RAM	vars.GenCPU.radiochannels[4]	
40015	000E	1	Radio Channel 6	R	Byte	RAM	vars.GenCPU.radiochannels[5]	
40016	000F	8	GPS time	R	Double	RAM	vars.GPS.utc	GPS Stuff
40017	0010		+2					
40018	0011		+4					
40019	0012		+6					
40020	0013	1	Hours	R	Byte	RAM	vars.GPS.hh	
40021	0014	1	Minutes	R	Byte	RAM	vars.GPS.mm	
40022	0015	1	Seconds	R	Byte	RAM	vars.GPS.ss	
40023	0016	8	Lattitude	R	Double	RAM	vars.GPS.lattitude	
40024	0017		+2					
40025	0018		+4					
40026	0019		+6					
40027	001A	1	Northing	R	Byte	RAM	vars.GPS.northing	
40028	001B	8	Longitude	R	Double	RAM	vars.GPS.longitude	
40029	001C		+2					
40030	001D		+4					
40031	001E		+6					

Figure 6.3: Excel MODBUS spreadsheet

6.3 Player software

The player project [115] was chosen as a framework to tie together all the sensors and control mechanisms on Meercat. Player implements a server that loads drivers as per a pre-defined script, and provides a mechanism for clients to interface to these drivers via a socket interface.

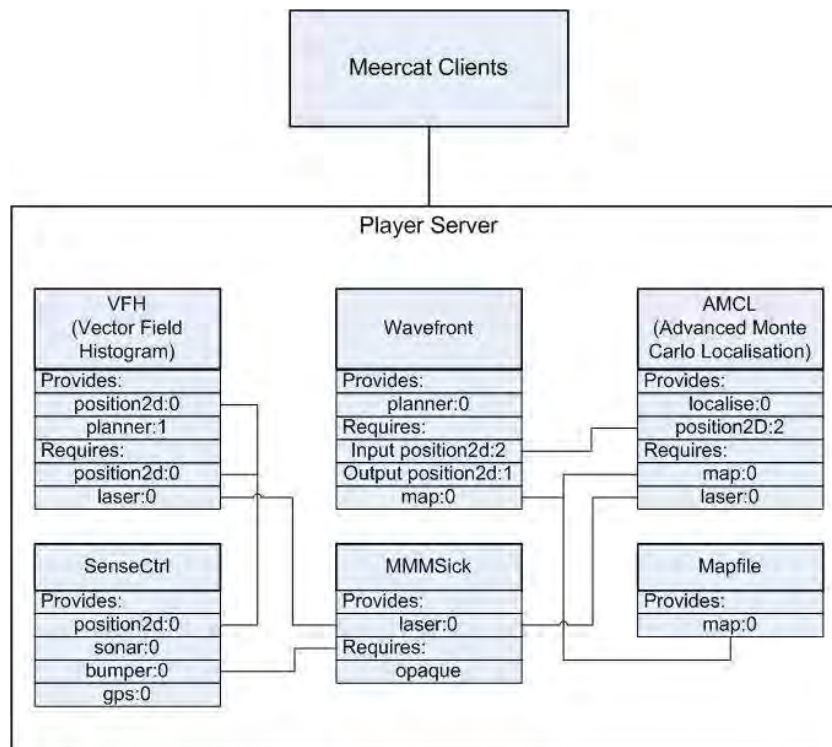


Figure 6.4: Player structure in Meercat

Structurally, the player framework defines interfaces, drivers and devices to handle a robotic system. An interface is a specification of how to interact with a robot sensor, actuator or algorithm. A driver is a piece of software that understands how to talk to a sensor such as a laser range finder or odometry sensor. A driver hides the complexity of accessing these sensors and is never accessed directly, rather it makes its information available via the definition of an addressable device.

As can be seen in Figure 6.4, the player available drivers (mapfile, wavefront, vfh and amcl) were combined with two custom-developed drivers. The first driver, called SenseCtrl, was developed to provide an interface to the Meercat hardware. The second driver, called MMMSick, provides an interface to the SICK laser range finder. This was developed [116] for general use in the Mechatronics group because the standard player laser driver never seemed to work correctly.

The SenseCtrl driver that was developed as an interface to Meercat's sensors and actuators implements the MODBUS protocol to communicate to the AVR32 processor. The commands that arrive from the higher levels are passed on to the relevant command registers on the MODBUS map, and the data is collected from other registers and presented on the relevant interfaces that the driver presents to the rest of the player system.

Initially the SenseCtrl driver's commands and data interfaces were tested from the custom written client software, which was a human operated command interface. This methodology showed that the principle of the system was working well. However, when autonomy was tested by using the planner drivers, it turned out that the system did not work that well. This was caused by the fact that the player drivers were sending data at a very high tempo and the initial implementation of the driver could not cope with that. After some changes in the SenseCtrl driver to the method of command handling and especially the communications protocol priorities, this issue was resolved.

Standard player utilities such as "playerv" and "playernav", which behave like client software, were used from a remote computer during some of the experiments to analyse and visualise the functioning of the system.

Appendix H of this document shows the file and directory structure used for the Player project software work.

6.4 Development and debugging

This paragraph provides details on the methodology of software development and the related tools that were used.

The player project aims for compliance with the portable operating system interface for Unix (POSIX), which means that it will operate on most UNIX based operating systems. For the embedded PC, the Linux based distribution Ubuntu was chosen because of its ease of use and wide availability of installation disks and images, but most importantly, the easily available South-Africa based code repositories. This last factor is important when updating the operating system or downloading utilities as the local bandwidth availability is much better and faster than the international links.

Because of Meercat's mobility, development and testing on the system had to be performed remotely via a wireless network interface. Virtual network computing (VNC) is a method of accessing a remote desktop as if it were present on one's own machine. Initially, this methodology seemed to be a promising approach to operate the mobile robot, but unfortunately the load it posed on computing and communications resources was too high, slowing the rest of the robot's operation. A better and more reliable solution was found with the secure copy (SCP) methodology, which uses the secure shell (SSH) protocol. Files were edited from a Windows client (WINSOCP) and further access into the system for compilation and operation was achieved via the PuTTY Telnet client.

To test the performance of the system, several software routines were written to evaluate and assist with operations. Because of personal preferences, the Windows Operating system together with the Borland C++ Builder development suite was used in creating this support software. In the following paragraphs some screen captures are shown and short descriptions are given of these utilities.

To connect to the Meercat remotely using the relevant player visualisation utilities, another UNIX based system was required. As limited computers were available and centralisation of code and resources was important for operational and backup purposes, a copy of SUN's Virtual Box software was installed on the Windows based machine. Virtual Box installs an additional computer on one's own computer, and another copy of Ubuntu was installed. The entire solution proved to be very reliable in operation.

6.4.1 Communication protocol testers

To test and verify the communication protocols that are used in the system, two Windows based programs were created. The protocol from the Generic CPU is, as discussed, a very simple dump of binary data. However, it was deemed useful to visualise this data and ensure that the functionality of the software and protocol was correct. To this end, a protocol test program was written, a screen capture of which can be seen in Figure 6.5 on the left-hand side. The program was very useful when initial tests were done visualising the response of the ultrasonic sensor.

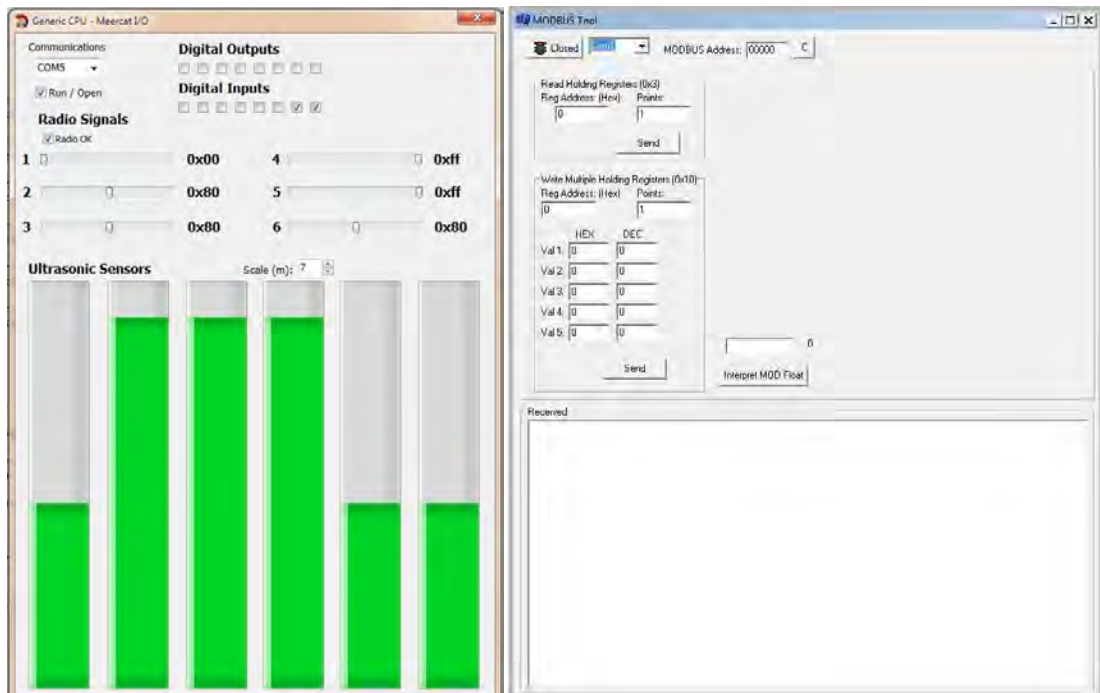


Figure 6.5: Generic CPU and MODBUS protocol testers

To test and verify the MODBUS protocol between the AVR32 processor and the embedded PC, another program was written to verify the functionality of the protocol's commands and checksum routines. A screen capture of the MODBUS protocol testing program is shown on the right-hand side of Figure 6.5. Once the communication protocol was tested and found reliable, the key communication routines developed under Windows were then ported and re-used in the development of the Meercat platform driver under Linux.

6.4.2 Mapping application

To visualise and post-process the positional data captured during the Meercat's experimental runs, an application called Map Builder was created. This utility allows for the overlay of data from diverse sources such as GPS, odometry and reference laser measurements on top of a loadable custom map. The map is referenced and scaled by means of GPS coordinates, so that the GPS logged data is properly drawn over the map. The laser and odometry logged data is read straight out of their respective logging files and this data is referenced to the map via custom defined origination points and scaling factors. Data can be represented using different shaped markers and colours, and the resultant displayed data can then be exported to a Joint Photographic Experts Group (JPEG) graphics file for further viewing or inclusion into reports.

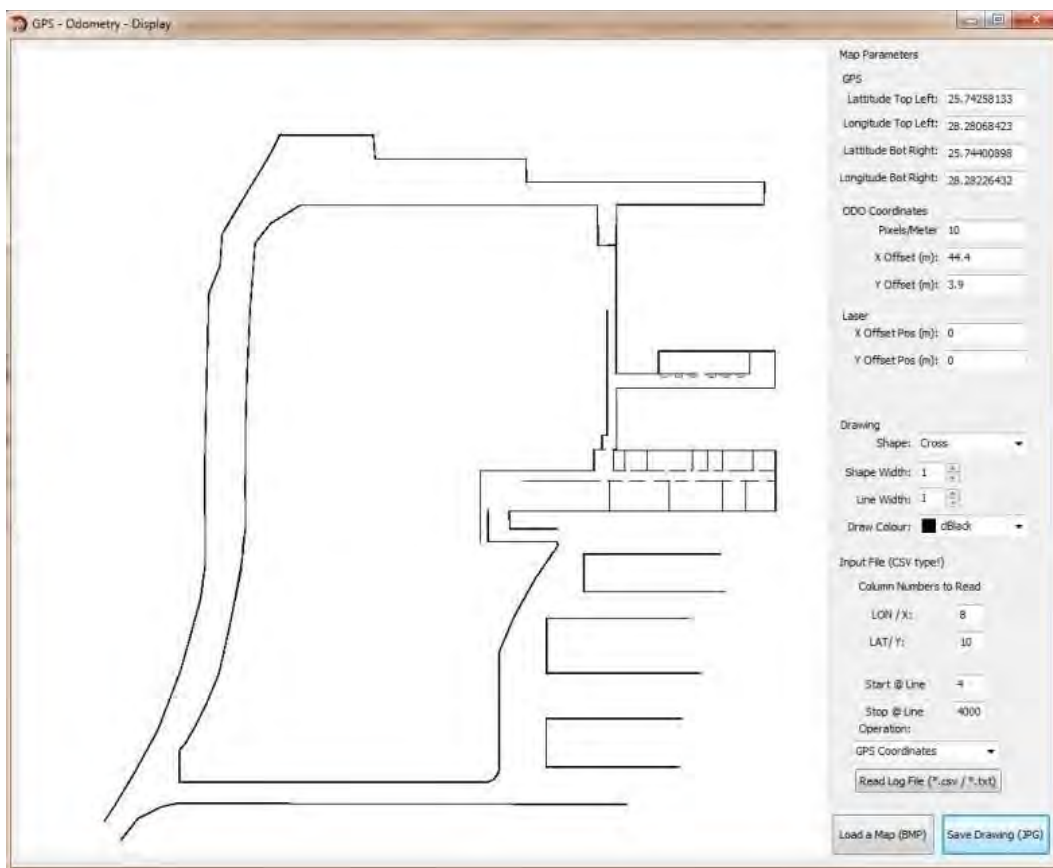


Figure 6.6: Map builder application

The captures that are shown in some of the experiments in Chapter 7 were created from the logged data from those experiments visualised by the Map Builder program.

6.5 Summary

The software systems used during the development and operation on various hardware platforms were analysed and discussed in this chapter. The developed software for the robot as well as the supporting software was included in the discussion. Although the player environment in particular was a steep learning curve, the development is deemed successful as a complete working and interacting system has been achieved. The extensive code sections can be found on the accompanying data pack.

7 Control System Design and Experimentation

This chapter discusses the control system design, some early calibration issues and indicates the experiments that were performed to determine the functionality of the system.

Some thought was given to the question of whether to adhere to traffic rules regarding outdoor navigation on a semi-public road on the CSIR campus, or to find other solutions. Implementing all traffic rules into the project was considered to be too complex at this stage, therefore a cooperative strategy was chosen for the Meercat. The Meercat platform is equipped with a visible light and flag so that road users are warned that something special is moving along their route. The platform's speed is limited to walking pace and this tends to prevent any major problems. This is a far from optimum solution, but one that enabled the experimentation with the sensors and navigation algorithms.

The possibility exists that the planned route is completely congested at which stage the robot will try to circumvent the obstacle(s) which might lead to the robot indefinitely trying to reach its target. This situation is acceptable during the experimental phase but if the platform is being put to actual day to day use then this factor will need to be addressed. The most simple solution to this problem would be to report back to a human operator that the platform is completely stuck and needs assistance.

Environment factors such as wind and rain will definitely affect the current platform, but again these factors were not deemed important during the experimental phase and were avoided by testing during dry and calm weather. Further engineering is possible to make the platform more weather proof, although there are limitations for making the sensors completely impervious to rain whilst still performing their correct functions. The solution then would be to stop operating whilst weather conditions are bad.

7.1 Meercat control modes

A standard player/stage position2D interface is implemented on the Meercat driver. The diagram shown in Figure 7.1 helps to explain the total functionality of the implemented system.

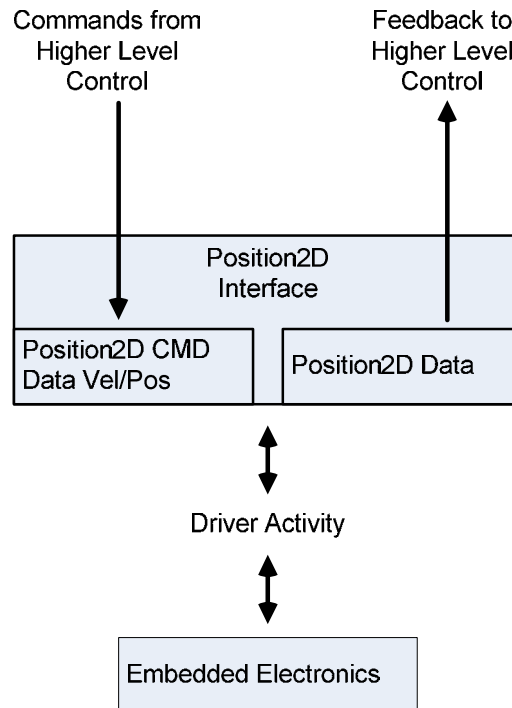


Figure 7.1: Meercat position2D interface and linkages

The position2D interface's data, known as pose information, is defined as an X (px), Y (py) position and platform orientation value Theta (pa). In the position structure, the data is interpreted as an x and y position in metres and theta as a platform rotation in radians. In the velocity structure, the x and y parameters are velocity values in metres/seconds and theta is a platform rotational speed in radians/seconds.

Most of the algorithms that are provided on player/stage, algorithms that perform obstacle avoidance, routing and planning, use velocity control mode. As a result, most of the position2D drivers on other robotic vehicles implement velocity control mode only.

On Meercat, two separate control modes are implemented: velocity control mode and position control mode. The position control mode was added in order to be able to perform scripted position control commands. The position control mode was added in order to be able to perform scripted position control commands. This control mode is used mainly for calibration of the odometry of the vehicle, but also so that a fixed movement pattern can be run from a known waypoint. Initial attempts at open loop control in this mode proved impossible as the left and right side of the vehicle reacted differently to motion commands resulting in the vehicle veering of course within a few meters of travel. The implementation of a PID control loop which takes feedback from the wheel sensors into account such that both wheels operate correctly at the required rate solved this problem. As an example, one of the experiments that was done using the Meercat simply ran an indoor movement on a fixed pattern without any obstacle avoidance,

and for this a scripted move was used. The UMBmark calibration procedure in paragraph 7.2 was also performed using this method.

The driver switches between velocity and position control mode depending on the type of control messages it receives from the higher level software.

The behaviour of the driver and embedded controller in velocity control mode is shown in Figure 7.2.

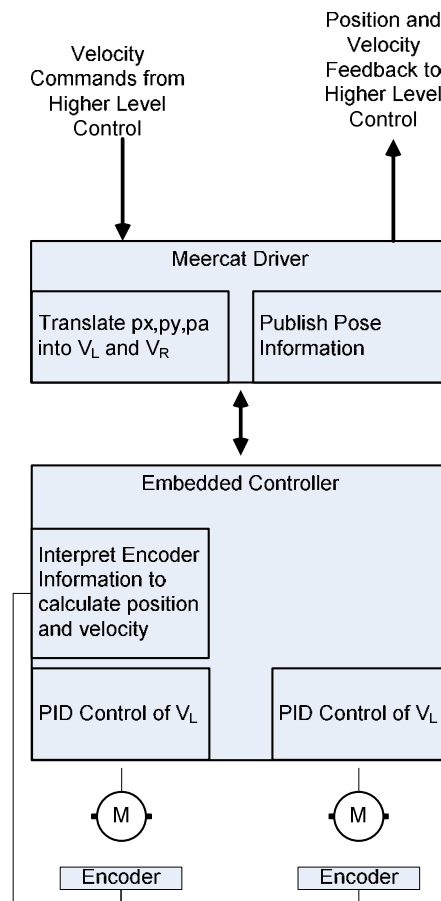


Figure 7.2: Velocity mode control diagram

An x (V_x) and y (V_y) speed is received in metres per second from higher level algorithms, as well as a rotational speed (V_a) in radians per second. This needs to be translated into a left and right wheel speed by the driver as shown in 7.3 to 7.5 below, which is then passed on to the embedded controller.

Two speed components are calculated: the first component as a result of the requested angular speed and the second component as a result of the requested x and y speed. Based on the formula used in the UMBMark literature where b is platform width between wheels:

$$pa = \frac{(Vl_{rot} - Vr_{rot})}{2b} \quad (7.1)$$

Note that rotation is around the axis of the platform, which means that:

$$Vl_{rot} = -Vr_{rot} \quad (7.2)$$

Therefore:

$$Vl_{rot} = \frac{Va}{b} \quad (7.3)$$

And:

$$Vr_{rot} = -\frac{Va}{b} \quad (7.4)$$

The speed component due to the requested V_x and V_y speeds:

$$Vl_{speed} = Vr_{speed} = \sqrt{(V_x^2 + V_y^2)} \quad (7.5)$$

The speed component and rotation component are then added together for each wheel and passed on to the hardware controller. A ramping function limits the positive and negative acceleration of the vehicle to ensure a smooth, non jerking behaviour.

In velocity mode, the embedded AVR32 controller implements separate PID controllers for the left and right wheel motor, and these control the motors on the difference between requested velocity and measured velocity. The functioning of the PID controllers was very apparent as without this control the robot tends to wander off course. With the PID velocity control, a straight line can be achieved. The parameters for the P,I and D values are stored in the player driver configuration file, and are sent to the embedded controller when the driver starts up.

The odometry calculations are performed on the embedded controller and the results are passed back to the driver, which publishes this information on its position2D interface.

In position control mode, the scheme works differently. Here the embedded controller follows a state machine procedure to achieve the requested position. The driver inspects the state of the embedded controller so that it knows when new position commands can be passed on.

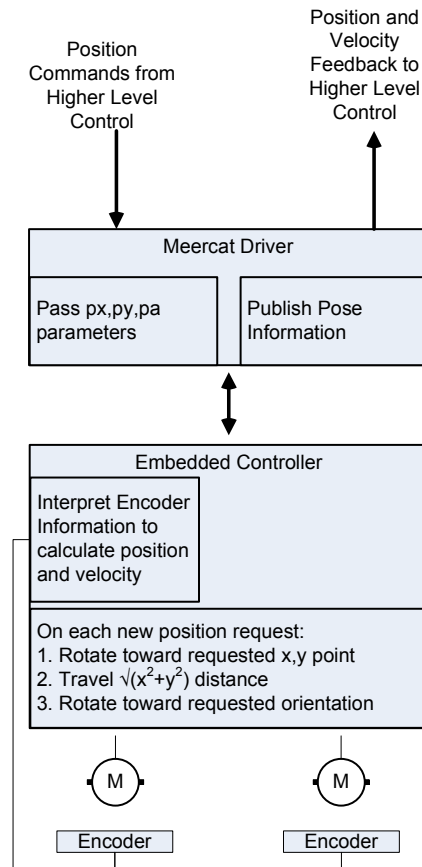


Figure 7.3: Position mode control diagram

A similar handshaking system is used between higher level software and the driver, where a client program can inspect the motor state of the position2D interface. Motor stalled means that the move has been completed. In addition, the driver will simply not accept any new position commands until the platform has either achieved the last requested position or a time-out has taken place. A position move starts with a rotation of the platform towards the intended target, followed by a high-speed movement, and then a lower speed movement. The low-speed move is intended as a means to approach the target slowly and avoid overshoot. The high and low speeds during this movement can also be controlled by setting the velocity part of the position2d interface. Some special non-standard trickery is used here as follows: px defines the high speed, py the low speed, and pa is a special control code which determines the threshold at which to change from high speed to low speed.

7.2 Odometry calibration

This section discusses the work that was performed to achieve a good working odometry subsystem. The odometry measurements and calculations form the basis of dead-reckoning navigation in shorter sections of travel. For Meercat, the UMBmark calibration procedure

developed by the University of Michigan was followed [39]. This calibration procedure addresses the systematic errors such as inaccurately known wheel diameters and errors in wheel base assumption, but the procedure does not address non-systematic errors such as rough surfaces created by bumps and stones.

The measurement of odometry starts with the quadrature optical encoders which, as previously described, are connected on the motor shaft. Since the encoder has a 32 stripe optical disk, this means that each full rotation generates a total of $32 \times 4 = 128$ counts. Through the gearbox, a total of 25 motor revolutions is required to achieve a full wheel rotation.

The wheel circumference was measured by winding a cord around the wheels, thus giving a good initial approximation of the actual wheel diameter. Based on this, the radius of the wheel was calculated. Using this method, it was found that for a single wheel rotation a distance of 958 mm is travelled for a total of 3 200 encoder counts, thus giving a very good resolution of 0.299 mm per count. The wheel base was measured with a tape measure between the centres of the wheels and was found to be 430 mm.

In Figure 7.4 commonly accepted definitions of the conventions are shown for a differential drive system, where b is the base of the platform, D the diameter of the wheels, and θ (theta) the angle to the world coordinate system along which the platform is travelling.

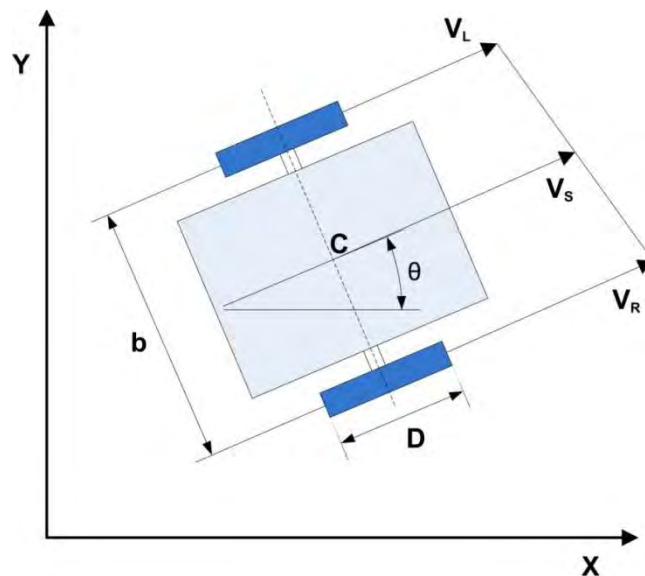


Figure 7.4: Differential drive definitions

Since the wheel is made out of rubber, a certain compression will take place as the wheel travels. Furthermore, the wheelbase cannot be accurately determined by a simple measurement.

In the following procedure we will correct for these two important factors, where E_d which is the ratio between the two wheel diameters and E_b which is the error in the wheel base.

In formula form:

$$E_d = \frac{D_R}{D_L} \text{ and } E_b = \frac{b_{actual}}{b_{nominal}} \quad (7.6)$$

The first step in translating the encoder pulses into odometry is done by introducing a conversion factor C_m , which translates wheel encoder pulses into a linear displacement.

$$C_m = \frac{\pi D}{n C_e} \quad (7.7)$$

In (7.7) n is the gear ratio between motor and wheel, and C_e is the number of pulses per full encoder rotation. By dividing the movement into small increments, for every instance the total distance travelled for each wheel then becomes:

$$\Delta U_{L/R,i} = C_m N_{L/R,i} \quad (7.8)$$

The movement for the robot's centre point ΔU_i is an average of the left and right movements:

$$\Delta U_i = \frac{(\Delta U_{L,i} + \Delta U_{R,i})}{2} \quad (7.9)$$

For the orientation angle the following relationships can be defined:

$$\Delta \theta_i = \frac{(\Delta U_{L,i} - \Delta U_{R,i})}{b} \quad (7.10)$$

To calculate the current x , y and θ for the robot, the following can then be used:

$$\theta_i = \theta_{i-1} + \Delta \theta_i \quad (7.11)$$

$$x_i = x_{i-1} + \Delta U_i \cos \theta_i \quad (7.12)$$

$$y_i = y_{i-1} + \Delta U_i \sin \theta_i \quad (7.13)$$

For the calibration procedure, the robot was made to travel over a square measuring 5 x 5 m where five counter-clockwise runs and five clockwise runs were made. At the start of each run,

the robot's odometry readings were reset and the end results were recorded. As a verification method, the actual position of the robot was measured by a stationary SICK laser rangefinder controlled by custom written software [117] intended to measure the robot's position and to log the data of each run.

The results of these calibration runs can be seen in the graphs in Figure 7.5.

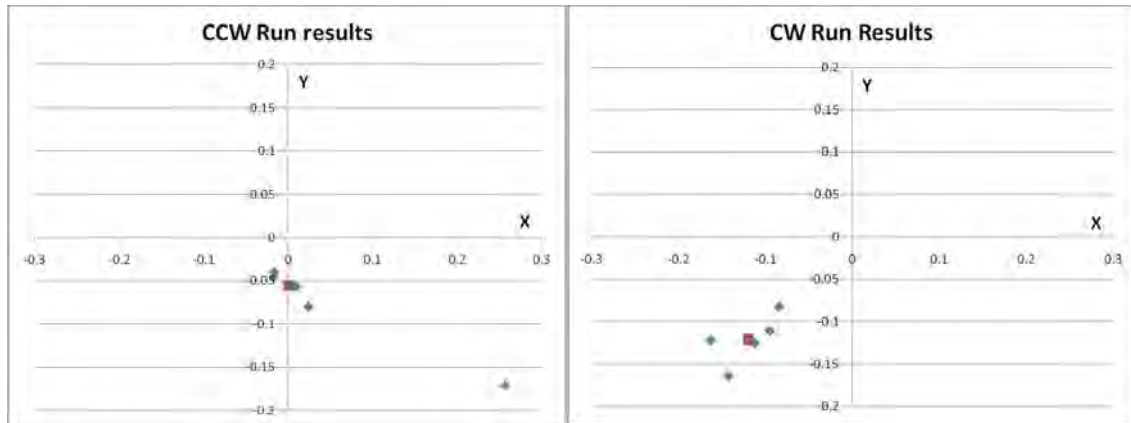


Figure 7.5: Calibration run errors

The blue dots in the graph show the difference in position between actual measured end-position by the laser range finder and where the robot reported it had stopped. The coordinates for these values are defined as $(\epsilon x_i, \epsilon y_i)$. The UMBmark method now follows the following steps to calculate the coordinates for the centre of gravity from the measurement results:

$$x_{c.g.} = \frac{1}{n} \sum_{i=1}^n \epsilon x_i \quad (7.14)$$

And:

$$y_{c.g.} = \frac{1}{n} \sum_{i=1}^n \epsilon y_i \quad (7.15)$$

The red dot in Figure 7.5 shows the centre of gravity for the group of results, where it must be mentioned that the single outlier in the CCW data was ignored. From the outcomes of these calibration runs, the largest of the two vectors towards the centre of gravity is now chosen as the measured error.

Due to unevenness of the wheels, an error is introduced when attempting to travel in a straight line. In Figure 7.6 an exaggerated situation is shown where the vehicle moves around an instantaneous centre of rotation.

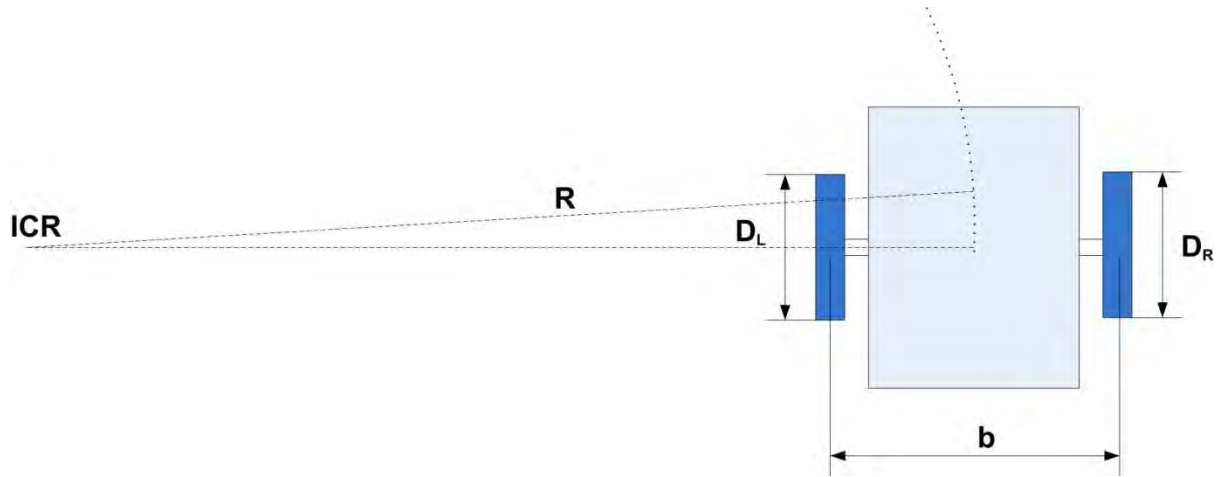


Figure 7.6: Effect of uneven wheels

The UMBMark procedure works through a number of derivations and some geometric relationships and then concludes with the formulas for the aforementioned E_d and E_b ratios, this time using the results from the calibration:

$$E_d = \frac{\frac{L/2}{\sin\left(\frac{y_{c.g.,cw} + y_{c.g.,ccw}}{-8L}\right)} + b/2}{\frac{L/2}{\sin\left(\frac{y_{c.g.,cw} + y_{c.g.,ccw}}{-8L}\right)} - b/2} \quad (7.16)$$

$$E_b = \frac{\pi/2}{\pi/2 - \frac{(y_{c.g.,cw} - y_{c.g.,ccw})}{-4L}} \quad (7.17)$$

In (7.16), 'L' is defined as the length of a single side of the test square. From (7.17) the size of actual base can be calculated when the results from the calibration runs are used. The base error is applied to the original base value, and the corrected value is used in the controller. Two calibration values for correction of wheel unevenness are derived. For the left wheel:

$$C_L = \frac{2}{E_d + 1} \quad (7.18)$$

And the right wheel:

$$C_R = \frac{2}{1/E_d + 1} \quad (7.19)$$

The total formula for an interval move in the system then becomes:

$$\Delta U_{L/R,i} = C_{L/R} C_m N_{L/R,i} \quad (7.20)$$

From the calibration run, the new base value was obtained using (7.17) and subsequently C_L and C_R errors are calculated from (7.16), (7.17) and (7.18). These values are stored in the configuration file which is transmitted to the controller on start-up of the system.

The Meercat controller is set to inspect the encoder results at an interval of every 40 ms and to calculate the odometry results using the formulas in (7.11), (7.12) and (7.13). The robot's orientation, position coordinates and velocity are then reported back to the player driver.

7.3 Localisation and navigation experiments

In this section a few of the more interesting experiments are described that were performed on the Meercat vehicle. All tests involved the use of the player software framework with different implementations of clients' and additional algorithms.

7.3.1 Wall-following experiments

A fairly simple algorithm was used containing a PID loop which attempted to keep a steady distance from a wall. Several experiments were performed in an unknown environment where no map definition was used, so as to evaluate the functioning of the sensors on the moving platform. The ultrasonic infrared beam and eventually the laser range finder data was used for sensing the corridor walls.

This test was a good initial experiment to evaluate the functionality of sensors, the embedded processors, communication algorithms and player project drivers in real-world circumstances. The outcome of the tests was that the ultrasonic and the laser range finder sensors worked well during the experiment. Whilst using the light sensors, the glass doors lining the corridor proved to be invisible which was an initially unforeseen but solvable problem. However, the infrared beam was discarded as a sensor for the office environment as it did not manage to correctly "see" the curved surfaces of pot plants in the corridors.

7.3.2 First odometry test

An indoor map detailing the office and laboratory environment was constructed from building drawings together with some measurement verifications at a resolution of 10 cm per pixel. After

the previously described UMBMark was completed, a verification run was performed over a distance of 30 m through an office corridor. The result of this experiment can be seen in Figure 7.7, where the green rectangle is the actual vehicle position, as verified manually, and the red rectangle is the position of the vehicle as reported by the odometry readings. The red 'x' on the right-hand side of the picture represents the starting position.

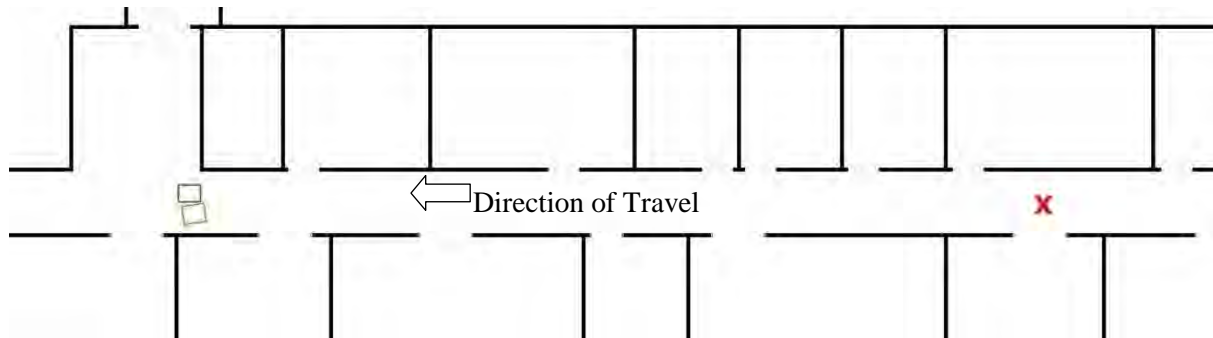


Figure 7.7: Experiment 1, Odometry

This result was very encouraging and confirmed the theoretical predictions. The error of the vehicle position as a result of odometry was mainly in the reported angle of the vehicle and much less in the X and Y positions.

7.3.3 Compass testing

Since the available literature is rather ambiguous about the performance of magnetometers in mobile robotics, an experiment was performed to test the performance of the compass. After the usual calibration run, the vehicle was moved from the laboratory section down a corridor and back again over a total distance of approximately 60 m whilst moving through the entire compass range.

In Figure 7.8, experimental data is shown where odometry outputs and compass outputs are matched, where the red data represents the compass data and the blue data represents the odometry data. Note that the “spiking” in the compass data is an artefact of the representation, where the heading wraps around from positive π to negative π . The repeatability of the heading is certainly very good as the start heading and the end heading are a very close match. Before and after each turn the vehicle was paused for a while, and this explains the straight lines. Some corrections were made during the movement from turn to turn and this is visible in the smaller angular changes in the data. The problem now is the fact that the compass data shows an angular change that is 150% larger on average during the small course corrections than that indicated by the odometry. Furthermore, it seems that in certain sections of the traversed trajectory

overshoots or undershoots in the sensor output take place. As we know that the odometry is fairly correct, the problem must lie with the compass data.

Attempts were made to find an explanation for this behaviour. A possible solution is that the motors were influencing the readings during motion. This was tested by putting the vehicle on blocks and actuating the motors equally and differentially. The results of these tests are shown in Figure 7.9, where the motors were intermittently actuated and rested as can be seen in the jump of the read-out. From the results it can be concluded that the motors do have an influence, but not to the degree that was seen in the movement experiment. Further detailed testing to get to the bottom of this effect would require additional equipment such as a controlled rate turn table.

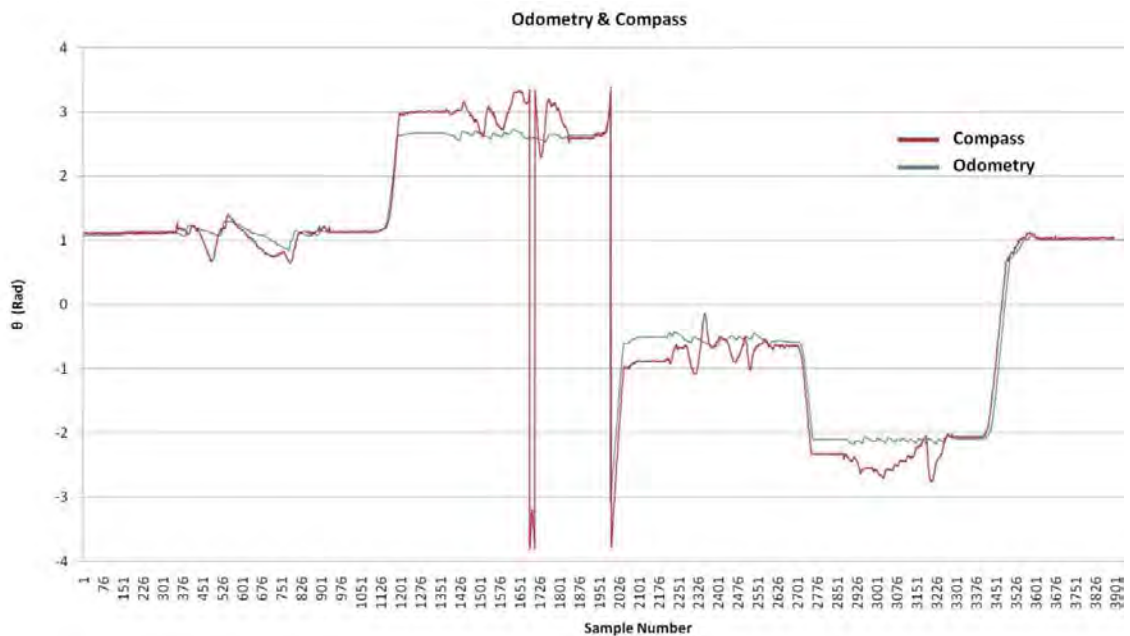


Figure 7.8: Odometry and compass data

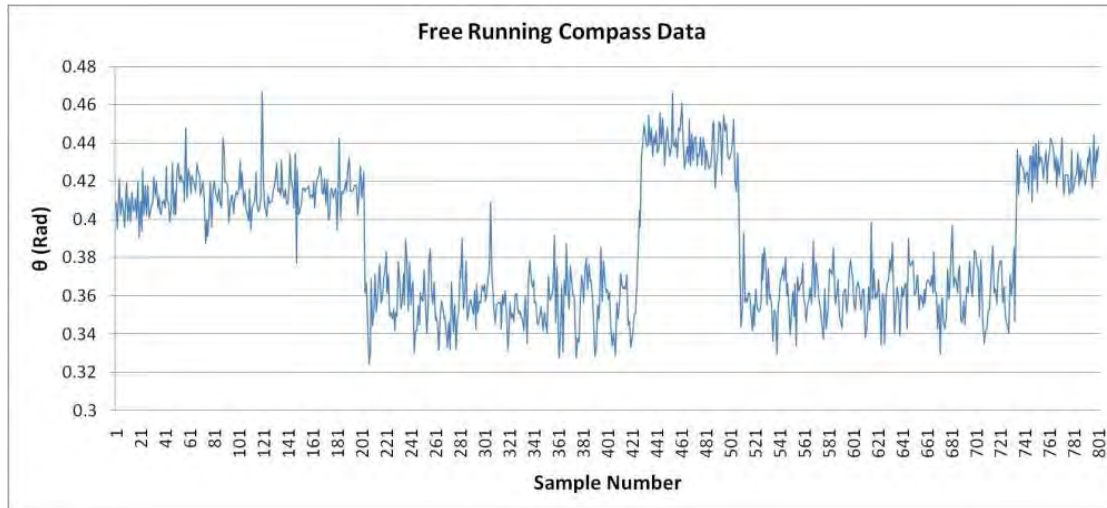


Figure 7.9: Compass data, motor effects

In conclusion, when a properly calibrated compass is used in this operational mode, it can be used as an indication of heading, but the margin of error is too extensive for direct use in navigation. However, the full capabilities of the device have not been used, and more accurate results might be possible.

7.3.4 Gyroscope testing

This experiment evaluated the on-board gyroscope sensor for its effectiveness as a sensor for vehicle heading determination. The same test track and test conditions as described in paragraph 7.3.3 were used for the experiment.

As the gyroscope experiences drift, this problem needed to be addressed. In Figure 7.10 the raw data as captured from the gyroscope can be seen in blue, together with an integrated value shown in red, and it is immediately clear that drift is significant.

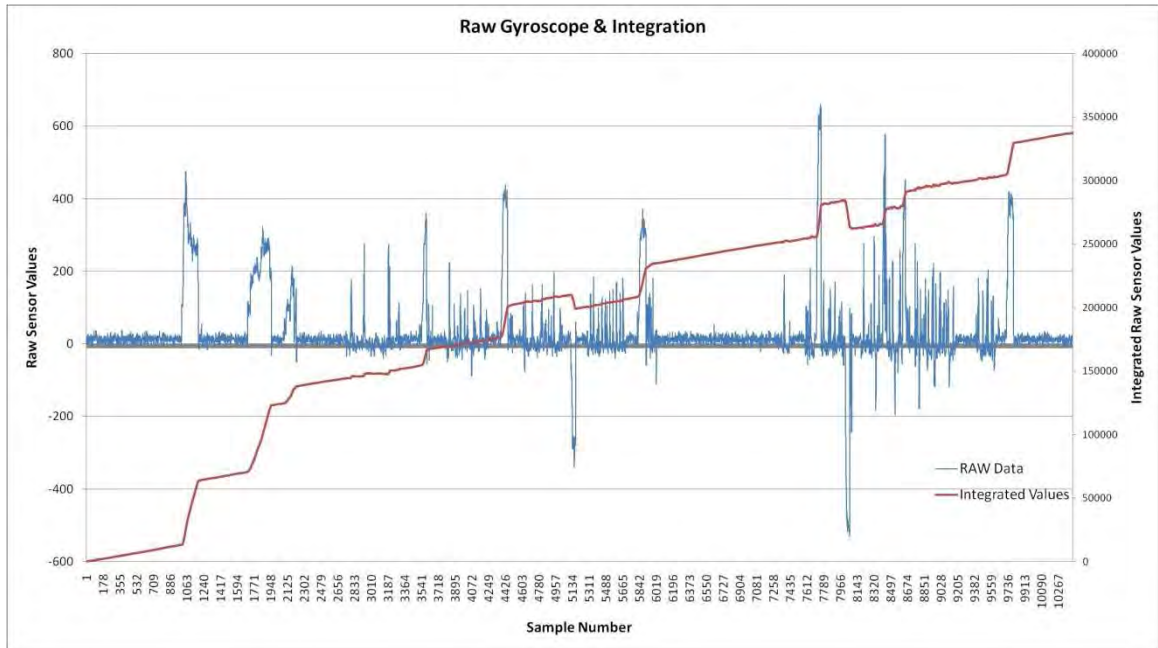


Figure 7.10: Raw gyroscope data

As literature [47] claims significant improvement in the sensor’s performance by careful calibration, it was attempted for this experiment. Data was sampled during an initial rest time of the vehicle and from this a drift trend was obtained. Further processing and translation to radians then gave the results as shown in Figure 7.11 where the gyroscope data was overlaid on the odometry results. In this figure, the green data represents the gyroscope data and the blue data represents the odometry data.

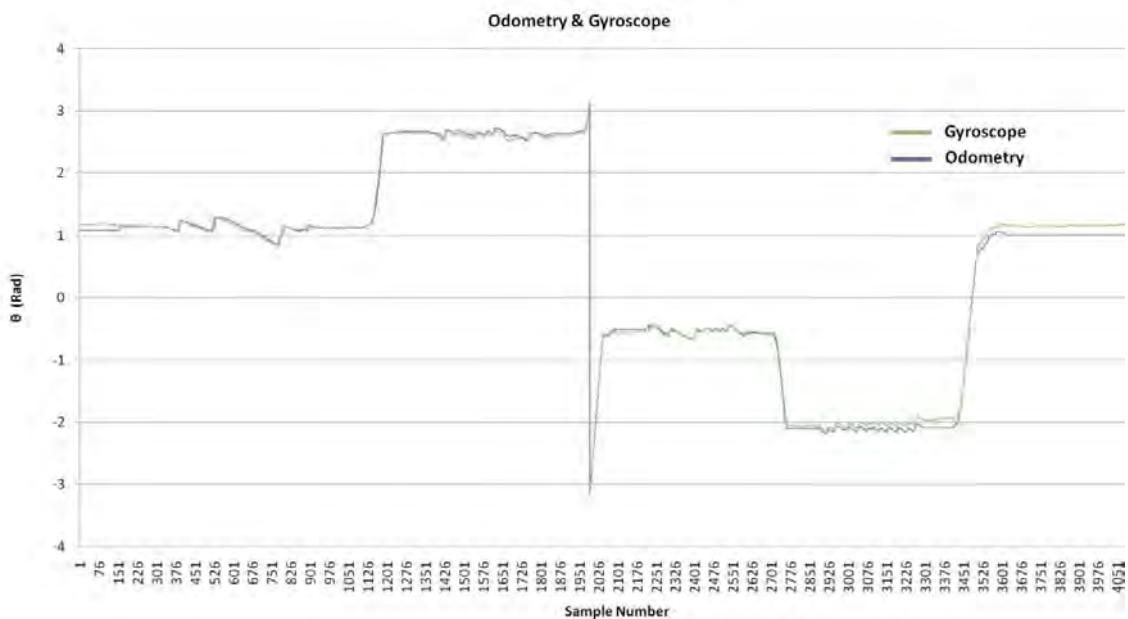


Figure 7.11: Corrected gyroscope data

As can be seen from the graph, the resulting data is surprisingly accurate and actually works much better than the compass sensor. A strong suspicion exists that this sensor actually provides better heading data than the odometry, as can be seen in the latter part of the graph.

Further improvements in the gyroscope's performance are still possible as no temperature compensation was performed. Some scheme of regular drift compensation whilst the vehicle is at rest would be a good simple method for obtaining an accurate heading indication. In conclusion, this sensor is a great asset for accurate determination of actual turning angles that were performed by the vehicle.

7.3.5 Odometry and GPS test

A combined indoor-outdoor experiment was performed to verify the accuracy of the odometry and GPS, the results of which are shown in Figure 7.12, where the green dots represent odometry results and the red dots represent GPS results. The indoor map was extended to also encompass an external area, where the full map shows an area of approximately 120 m x 120 m. The total distance driven by the robot is approximately 450 m. The experiment was started indoors in a corridor, followed by an outdoor move around the building, entering again and finishing up at the same spot. For reference purposes refer to the image in Figure 7.13, courtesy of the Google Earth program, to understand where the indoor and outdoor sections are situated.



Figure 7.12: Odometry and GPS, indoors and outdoors



Figure 7.13: CSIR Building 14F & 14E, Source: Google Earth

As predicted, and as shown in the previous test, the odometry lost track eventually, but the interesting part is that it can be clearly seen once again that the errors lie with the reported angles and not the travelled distance. The GPS disappointed indoors, but in the outdoors section it produced acceptable results. The top section of the map shows an extensive area of red dots, this was where the vehicle paused to wait for proper GPS acquisition.

7.3.6 Advanced Monte Carlo localisation experiment

In this experiment, the AMCL localisation driver from the player software collection was used on the Meercat robot to detect the position of the robot. For details on the principle behind this driver, refer to the literature survey chapter. The software was started such that the position of the robot was unknown and data from the laser range finder was used for the AMCL algorithm. The objective of the test was to determine if the AMCL algorithm would function in the office environment, the speed at which it functions and also to evaluate whether the algorithm would be able to function starting from an unknown robot position.

To read the data from the AMCL algorithm, the “playernav” utility was used, which accesses and visualises localisation interfaces. The GUI utility publishes data on the screen, but it can also be set to generate repetitive snapshots. For this experiment, the reporting rate was set for 5 Hz. The robot was positioned on a random place on the indoor map 30 m x 30 m in size and the player server and clients were started. The results of this experiment can be seen in Figure 7.14, and it is clear that the algorithm worked very well. The image is a composition of consecutive snapshots, which means that the algorithm was able to narrow down the robot’s position within a second of operation.

In the step one picture, the red zone indicates the possible areas where the algorithm is trying to find a solution; the darker red areas (very few) have a higher likelihood of producing a solution.

The little circle represents the robot, with the triangular section indicating the front of the robot. Note that the starting position and orientation in the image was random.

In the step two picture, the robot is repositioned in the centre of the map, and two areas of interest can be seen to be developing. In the step three picture, only one area of interest has survived, the algorithm's search area is adaptively reduced, and the reported position of the robot already approximates its actual position. In the step four picture, the process continues and the robot's position is basically determined.

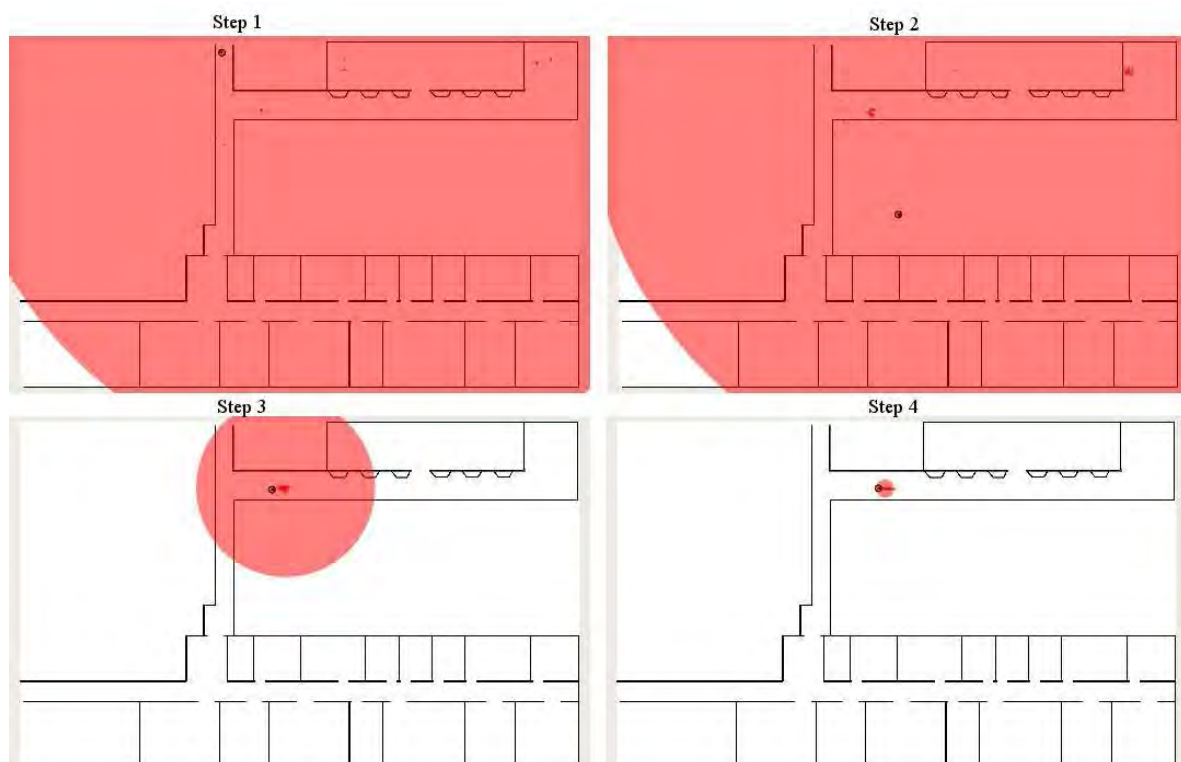


Figure 7.14: AMCL localisation sequence

Driving the robot further through the map showed that the algorithm was able to track the robot's position very well, even in the face of moving obstacles (people) and undefined obstacles. It is clear that by using this methodology for localisation, an autonomous robot will operate successfully in an indoor environment. The solution is robust and reliable and appeared to work well. The algorithm worked well even when the initial robot position was unknown.

7.3.7 Navigation experiment

Now that the AMCL experimentation has proven that localisation does work, the robot was commanded to autonomously proceed to a goal. The addition of the Wavefront algorithm for planning a route to the intended goal and the use of the Vector Field Histogram algorithm for

local obstacle avoidance ensure that the robot will navigate around any well-defined indoor map. The output from the Wavefront planner is shown in Figure 7.15, where the small triangles show the waypoints generated by the algorithm.

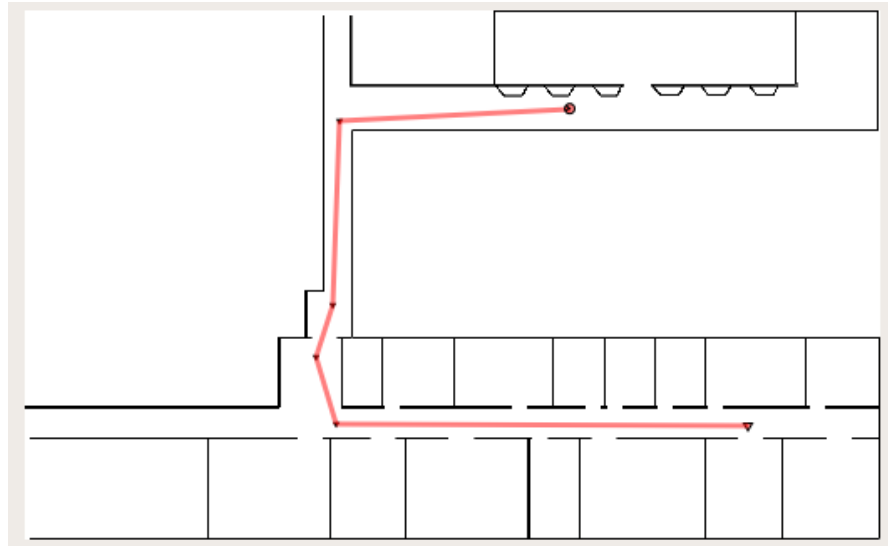


Figure 7.15: Wavefront route planning

7.3.8 Differential GPS navigation experiment

Having solved the problems of localisation and navigation in an indoor environment, a good working methodology had to be found for outdoor navigation. The outdoor environment is much more complex than an indoor environment, and creating a detailed map showing all obstacles would be a momentous task. A better approach is to create a map which defines go/no-go areas through which the robot can manoeuvre, relying on the available sensors for localisation and obstacle detection. Another option would be the use of many more sensors detecting the road edge and other obstacles, but this was not a viable option as the cost of such a solution would be too high. The GPS solution as tested in 7.3.5 was not accurate enough and it was decided to attempt the use of the differential GPS solution as described in 4.3.3.

For verification purposes, a detailed GIS survey map was used depicting a part of Pretoria showing the CSIR campus with a better accuracy than that obtained from the Google Earth application. The open source “MapWindow” application [118] was used for viewing and recording the waypoint information. An experiment was performed where the vehicle moved on the centre line of the CSIR ring road, and in Figure 7.16 a snapshot is shown of the result of these tests. The green dots in the image represent the position of the vehicle according to the GPS receiver and the purple dots represent the corrected position using the RTKLib software.



Figure 7.16: Differential GPS position determination

The image proves that the differential GPS solution is able to determine the robot's position accurately to within an error of 10 cm. The solution appears to be viable but cannot be fully trusted as signal obstruction and/or multipath reflections could still affect the position determination. Especially during initial experiments, higher buildings and trees close to the roadway affected the results dramatically. After consultation, a solution was found by adding a metal plate underneath the receiver's antenna, thus possibly cancelling out multi-path reflections.

It is clear that even the very accurate differential GPS solution cannot operate as a stand-alone solution as its availability is not assured and the rest of the robot's sensor suite must be used in conjunction with this system.

7.4 Summary

This chapter discussed the implementation of and experimentation with a system for the control, positioning and navigation of the Meercat mobile robot. A number of experiments were described which tested the performance of the various subsystems of the robot. From the experiments, it can be seen that a viable solution has been found and the mobile robot is able to move at least for a limited time autonomously through indoor and outdoor environments.

8 Validation and Conclusions

This chapter discusses the validation of the research study, covering all the research objectives on a paragraph by paragraph basis. Discussions through the previous chapters and the outcomes of the described experiments will assist with the evaluation of the performance of the Meercat courier robot.

Conclusions are then drawn on the level of success in implementing the system and a set of additional recommendations are made.

8.1 Validation of robot construction

The first objective of the research study was defined as follows:

The design and construction of a mobile courier robot which is able to traverse an indoor office environment as well as an outdoor campus environment, using sufficiently low-cost mechanical components, sensors, electronics, as well as power and drive systems; in addition, the robot must be equipped with its own stand-alone computing source and must be easily interfaced by remote control.

8.1.1 Mechanical components

Standard wheelchair components were selected to build the mechanical component of the robot since they are well priced and readily available. A custom design would have been much more expensive and it would have been difficult to build additional vehicles. The wheel size is the proper selection for indoor and outdoor environments resulting in a good trade-off in speed and available torque from the motors whilst being able to traverse smaller surface bumps without problems. In all environments (indoor office and outdoor paths and roads), the robot was able to travel without experiencing any mobility problems. The choice of standard aluminium extrusion elements with their coupling elements combined with aluminium sheets made the mechanical construction of the robot achievable without extensive tooling and at a relatively low cost.

8.1.2 Sensors

The sensor choices went through a number of optimising iterations which improved the performance of the mobile robot and its ability to sense its environment properly. The final set of sensors used on the platform is as follows:

- Bumper sensors
- Three-axis combined accelerometer and gyroscope sensors
- Compass sensor
- Motor encoder sensors
- Ultrasonic range finder sensors
- Laser range finder sensor
- Differential GPS

The bumper sensors were implemented to ensure that the platform stopped when required, and these functioned well for their given task. Integration of these sensors into the system was the simplest task of all sensors. Closing contacts were used, which are easily interfaced to the digital inputs on the embedded electronics. A possible improvement would be the addition of a bumper system at the back of the vehicle ensuring that the robot would not bump into obstacles while in reverse. However, the robot typically travels forward, reversing only when no other option is available, and then only for short distances over terrain that has already been covered. Another possible option, given the current electronic capabilities, is the implementation of a finer grained bumper system. The current bumper system only detects and reports left front and right front obstacles, whilst it would be possible to increase sensing at the front into eight distinct zones, allowing the robot to make better decisions on how to manoeuvre away from an obstacle.

The accelerometers and gyroscopes work very well, as is evident from the results in paragraph 7.3.4. Further integration work on these systems could potentially augment the odometry system, by for instance correcting for non-systematic errors such as bumps or dips in the travelled path using, for instance, the “Gyrodometry” [42] methodology.

The performance of the compass sensor was disappointing. Although a rough indication of platform heading is possible with this sensor, it does not generate very accurate results, as can be seen in Figure 7.8. Factors that make the sensor unsuitable for the mobile robot include the fact that the compass needs a calibration phase before every run, and this calibration does not last very long. The performance of the ultrasonic system was proven during the wall-following

experiments, with good results being obtained. However, the six-sensor configuration is a minimal implementation which is only sufficient for obstacle detection applications. These sensors cannot be used in mapping applications on their own. The robot would require many more additional sensors; typical examples are the 24 sensors on University of Michigan's CARMEL robot [119] which are evenly positioned around the entire periphery of the robot. An implementation like this would result in a much higher sensor cost and, in addition, the interfacing complexity would also increase. Even then, the ultrasonic sensors have their limitations as an array of 24 sensors would result in a total cycle time of over 1.2 s and data sections with an angular resolution no narrower than 15° .

The wheel encoder placement onto the motor shaft instead of the wheels was the correct decision as very high resolution position information is now available for further processing and conversion into accurate odometry information. The high interrupt rate resulting from the encoders was easily handled by the high-speed AVR32 processor, especially when the decision was made to limit the top speed of the vehicle. From a robust system design point of view, further optimisation could be achieved by the implementation of a small dedicated odometry processor, lightening the load on the larger processor.

The odometry of the robot worked exactly as all the literature predicted; it operates well and is reliable for short distance dead-reckoning, but for longer operating distances a different means of localisation should be used. The errors in the system were corrected by means of the UMBMark calibration scheme, minimising the systematic odometry errors. However, non-systematic errors still affect the system as can be seen from certain test runs. Refer to Figure 7.12 which shows how the odometry was affected during a test run because of a transition in the road causing minor wheel slip. Attempts to correct for this were not high priority as odometry is only partially relied upon for the localisation of the mobile robot.

The most expensive sensor system on the vehicle, the laser range finder is admittedly not a very low-cost solution, but it is expected that alternative lower cost solutions performing this function will be available in the near future. The sensor operated well at all times, producing good results and enabling the robot to perceive its environment well for obstacle detection, as well as giving inputs into localising the mobile robot. Software proved to be the biggest problem whilst using this system since, for unknown reasons, the standard driver never worked properly. Fortunately, the home-grown laser driver [116] was able to assist. Glass panels and doors are still an issue as they cause errors when the laser shines straight through them. This must be taken into account during navigation of the mobile robot. As well as this sensor works, the inputs of the other sensors are still relevant for obstacle detection purposes.

The differential GPS sensor was one of the unexpected successes in the project. At the start of the project, there was some uncertainty as to how outdoor localisation would take place. The implementation of this localisation solution had a positive effect on the success of the platform. The fact that this system is very low cost, by a factor 20 or more, than any other available commercial system whilst requiring less physical space and less power already makes it a very successful subsystem on the mobile robot. It was shown in paragraph 7.3.8 that this system gives very accurate localisation information. The conclusion is therefore that this sensor system works well in the intended outdoor environment.

8.1.3 Electronics

The electronic interfaces to the various sensors and computing resources, as well as the lower level embedded computing, was implemented on the AVR Mega128 equipped generic CPU board combined with the high-speed AVR32 equipped EVK1100 board. These systems performed their tasks as intended without any major problems and did not require high power during their operation. The complexity of this system is due to the extensive input and output requirements for the interfacing to the platform's sensors as well as the available embedded processing solutions. Although the integration effort was significant and complex, the resulting system works well. The current combined solution of an 8-bit and a 32-bit microprocessor requires skills in multiple programming environments, which although not a major issue, is still somewhat sub-optimal as a complete solution. Especially during debugging of the system, the requirements of programming tools, interface ports and several applications operating at the same time make for a complex environment. Further optimisation of the system could be achieved through the implementation of a more integrated design, although it should be noted that this would not be a trivial effort. The current system works well and certainly meets the intended outcomes with reserve input, output and processing capacity still available.

8.1.4 Power and drive

The choice of the commercial RoboteQ motor controller made for a very reliable drive solution, which optimally drives the motors of the mobile robot. The solution is not extremely low cost, but its reliability and sturdiness is an important factor contributing to the success of the mobile robot. The Mechatronics Group made repeated efforts to construct custom motor drive solutions with a view to make available a lower cost solution for multiple robot systems. To date these efforts have still not come to proper fruition, which proves that designing motor drive systems should never be underestimated. Taking this input into account, the selection of a commercial-off-the-shelf (COTS) system was the right decision as it allowed for concentrated effort in other areas of the mobile robot design.

The 24 V and 12 V sealed lead acid battery combination is the optimal solution for this mobile robot in terms of cost and ease of operation. Although other battery technology solutions could be implemented, all of these would be at a significantly higher cost and would require more complex charging solutions. Currently the voltage levels of the 24 V battery system can be monitored via the motor controller, but the 12 V system is not measured. A potential small optimisation of the Meercat mobile robot system would be the incorporation of a monitoring system on the 12 V power rail. This would become relevant later in the robot's life when the battery capacity diminishes due to ageing.

8.1.5 Computing and communication

A proper low-cost embedded PC was chosen as a stand-alone computing solution, which has operated the Meercat robot without a single failure. Even though the PC was powered down often without the proper shutdown procedure, the operating system always started again properly without a single error being reported. Despite the limited space and the additional worry of unreliability due to flash memory wear, the choice of a solid-state hard-drive over magnetic rotating media has proven to be correct. This is especially true in the light of the fact that similar computing platforms on other mobile robots in the MMM group have had failures of magnetic hard-drives due to vibration.

Communication with the system for remote debugging and monitoring is via a standard access point. Some unexpected and irregular breaks in communication occurred because of power supply noise and this caused a delay in operations as communication had to be re-established before work could continue. Although the system worked well enough during experimentation and has always proven to be reliable enough, there is potential for improvement in the system through the implementation of an alternate wireless solution. Selection criteria for a new solution would be power usage, wireless range, speed of start-up and speed of recovery from errors.

This, however, is not an insignificant task as it means that the appropriate access point or alternate product such as a Wireless USB adapter has to be selected from a multitude of potential solutions without an immediate guarantee of success. The reason for this is that the reliability of the new solution will only be proven during further extensive trials of the robot.

8.2 Validation of localisation

The second objective of this research study was defined as:

The mobile robot must be able to accurately locate itself by different localisation methods in indoor and outdoor environments as applicable, such that it understands its position at all times.

8.2.1 Indoor localisation

Indoor localisation was achieved for short distances up till 30 meters through the use of dead-reckoning odometry starting from known positions. For any longer travelled distances the AMCL driver from the player project used the laser range finder data in indoor environments to localise the robot accurately as shown in paragraph 7.3.6. This latter algorithm proved to work so well that this localisation method was chosen as the primary method for the Meercat's localisation.

8.2.2 Outdoor localisation

Using the same indoor localisation method for outdoor localisation is not practical as this requires a detailed true obstacle map of a very large area. As the map for the outdoor environment focuses on go/no-go areas rather than the accurate definition of obstacles, another method had to be found. Initial tests using a standard GPS receiver to localise the Meercat robot in an outdoor environment produced promising results, as shown in paragraph 7.3.5, but the method was not deemed to be accurate enough for sustained outdoor localisation and navigation. The implementation of a differential GPS solution, as discussed in paragraph 7.3.8, resulted in a solution that proved to be extremely accurate and very reliable.

An approach towards making the system even more robust could be the addition of a low-pass filter on the differential GPS data. Better still would be the use of the odometry sensor data to inspect the differential GPS results and make intelligent corrections to jumps in position caused by disturbances in the received signals.

8.3 Validation of navigation

The third objective of this research study was defined as:

The mobile robot must be able to accurately navigate itself in indoor and outdoor environments, it must be able to reach intended goals and whilst navigating, avoid expected and unexpected obstacles.

By using the pre-defined map of the terrain, combined with the use of a wavefront planning algorithm, as shown in Paragraph 7.3.7, the Meercat robot was able to plan a path towards the intended goals. During all of the tests, the bumper sensors, ultrasonic sensors and laser range finder worked together to ensure that obstacles were avoided at all times. Unexpected stationary and moving (humans) obstacles were detected and avoidance of these obstacles was achieved through the use of the player project's VFH driver.

A possible further optimisation of the Meercat to make it more robust, would be the addition of a sensing solution for very low obstacles or unexpected (pot) holes in the road. Potential solutions for this could be an additional ultrasonic sensor that is placed facing downwards in front of the vehicle although some testing would be required to ensure rapid and more importantly correct response. It should be noted that this is only a hypothetical problem since this was never an issue for the Meercat robot.

8.4 Validation of specifications

With all three research objectives having a positive outcome, the specifications that were defined for the platform need to be examined, as follows:

- Physical dimensions
- Payload
- Endurance
- Speed
- Manoeuvrability
- Vehicle Position Accuracy.

The first parameter, physical dimensions, is easily met as the Meercat robot measures less than 50 cm in width, easily allowing passage through corridors and door openings.

The payload capabilities of the vehicle easily match the intended 5 kg weight requirements, although more than this can easily be carried without significant performance penalties.

The endurance of the system has been addressed in the electronics section of this document, and has been shown to match the requirements. Confirming this practically in a tongue-in-cheek

fashion, the drive batteries became known as “magic batteries” after they managed to operate for several days of infrequent testing without requiring any recharge.

Speed tests of the vehicle showed a potential speed far in excess of human walking speed, so much so that the mobile robot’s speed was limited in the software as the top speed was deemed dangerously high in the light of available sensor data. The limit however is not a static implementation such as a percentage limit on the motor output but rather a dynamically monitored top speed of the platform. In this way a higher platform payload is compensated for and the resultant top end speed remains the same.

The differential drive nature of the system allows the platform to easily manoeuvre in confined spaces, turning on the spot and overcoming caster wheel friction without trouble. The small physical size of the robot assists with this manoeuvrability as even in corridors the vehicle fits quite easily and can turn when needed without requiring too much open space.

The experiments that were performed on localisation met the required positional accuracies, indoors as well as outdoors. For shorter lengths of navigation the odometry, gyroscope and even compass sensors can be used, whilst during longer indoor movements the location is accurately tracked through the use of the AMCL algorithm. Outdoor localisation through the use of the differential GPS solution worked well and it has been shown that the mobile robot’s location can be very accurately tracked.

8.5 Conclusions

It is clear that this project has met all of its intended outcomes. The project has been very useful in the creation of a robotics platform with numerous sensory inputs, a good locomotion system that is able to go where needed in the intended environment, and a logically structured software framework.

The availability of the player project in grouping the contributions of various robotics scientists has been a major contributing factor in achieving success with Meercat.

Numerous lessons have been learned from literature, construction and experimentation on the platform, strengthening the knowledge on the theory and practice of mobile robotics systems not only for the author but also for the rest of the CSIR Mechatronics group. The Meercat is a very good stepping stone - already the platform is being used by colleagues for other experimentation and evaluation, specifically the testing of a home-grown SLAM algorithm.

Although all navigation experiments resulted in the successful operation of the robot, all of these experiments were performed in supervised circumstances and more work needs to be done to ensure that the Meercat is a truly autonomous robot. The first few steps towards such a solution could be the creation of one or more docking stations for charging purposes, an operator (receptionist) interface to easily command the robot and the availability of a situational status display interface with alarming functions when situations arise that the robot cannot cope with.

8.6 Summary of contributions

Contributions were made in the fields of mechanical, electric/electronic and software engineering. The mechanical engineering contribution comprised the concept design, physical design and platform construction of the eventual mobile robot, where attention was given to ease of procurement and low cost components.

The electronic contribution was the integration of a diverse set of sensors, wireless interfaces, motor controller, embedded processors and computing platform. These solutions were designed to be easily accessible on the robot, which is important for repair and especially during development of the system.

The software contribution was the creation of a player driver to easily interface to the functionality of the robot from higher levels; the creation of sets of software on multiple embedded platforms interfacing to sensors and communicating to the embedded computer; and the creation of a number of supporting programs for the testing and evaluation of results from several systems.

8.7 Recommendations

The intent is to continue development on this vehicle with more advanced sensors, further position and navigation algorithms, and to that end, the vehicle is ready for further experimentation.

New sensors are constantly becoming available, which could possibly augment or improve the operation of the Meercat vehicle. One example is a newer version of an inertial sensor from Analog Devices, known as the ADIS16385, which is form-factor compatible with the current solution, but claims to offer tactical grade accuracies. Another example is LEA-6R GPS module which combines GPS reception with a “Dead Reckoning” solution where an external gyroscope and odometry input can be used to estimate position in the case of signal loss. Evaluation of

these sensors and comparing their performance to the currently implemented sensors might lead to an even lower cost solution.

Further work should be done to find a solution for an accurate low-cost obstacle avoidance sensor. The SICK laser solution works extremely well but is not cost-effective. Although costs are constantly coming down, an order of magnitude jump in costs is required to make things really interesting. As mentioned before, it is probably only a matter of time until a low-cost laser range finding solution becomes available. Other solutions could be explored however; the referenced work on a simple laser beam combined with an image sensor is a possible route to explore.

Currently the platform is not equipped with any camera solutions and in principle the use of multiple low-cost webcams in a stereo-vision arrangement could potentially provide a much richer source of information on the environment. However it must be noted that the use of cameras brings its own problems as varying light intensity is still a major problem for reliable recognition of objects. A possible augmentation could be a camera solution combined with a transmitted laser pattern thus enabling a simpler distance measurement solution.

An alternative option, although discarded in the current project because of costing, would be the use of a so-called Swiss Ranger infrared camera/range finder [120]. This device generates a rich point cloud of distance measurements and seems ideal to cope with office environments.

As a result of the work on the AVR32 platform in this project, the CSIR Mechatronics Group has decided to create a new embedded platform, known as the “Green Mamba”. This platform combines the AVR32 processor with a modular expansion solution, which implements an extensive array of peripherals. The primary idea behind this solution is to standardise the electronics solutions in projects that are being undertaken in the group, but secondary the idea is to make the solution available to a much wider group of people. Students from several universities in South Africa have already expressed their interest in obtaining these modules for their projects, as procurement of processing modules remains a problem. This then creates an active community of people across South Africa, working together on Mechatronics and Robotics projects. An upgrade of Meercat to the new processor board would be a good exercise in order to prove the functionality of the Green Mamba solution and to provide a reference design for a robotics solution to the rest of the community.

Much work remains to be done on localisation, mapping and navigation since so many options can be explored and researched. The SLAM system would be a welcome addition to the robot, as mapping all of the office environments manually would be a tedious job.

When Meercat was implemented, the player software structure was the correct way to go as many sources were actively using it. Since then the development of ROS, referenced in the literature chapter, has taken off with a huge international following. It is also clear that the development rate of this solution is taking place at a more coordinated and faster pace than the Player/Stage/Gazebo project. Through collaborative efforts with other groups in the CSIR it has been decided to move away from Player/Stage to the new ROS. It is recommended that the Meercat project be upgraded to this new solution, especially since ROS provides for a number of very good reporting and debugging tools, making evaluation of new sensors and algorithms an easier task.

Bibliography

- [1] (2010, October) The International Federation of Robotics. [Online].
<http://www.ifr.org/service-robots/>
- [2] (2010, October) IEEE Robotics and Automation Society: The Technical Committee for Service Robots. [Online]. <http://www.service-robots.org/about.htm>
- [3] (2010, October) Internation Federation of Robotics: Service Robots. [Online].
<http://www.ifr.org/service-robots/>
- [4] Brady Forrest. (2009) Tweenbots: Cute Beats Smart. [Online].
<http://radar.oreilly.com/2009/04/tweenbots-cute-beats-smart.html>
- [5] (2010, March) Willow Garage: About Us. [Online].
<http://www.willowgarage.com/pages/about-us/overview>
- [6] (2009, June) iRobot Corporation: Roomba. [Online].
<http://store.irobot.com/home/index.jsp>
- [7] (2010, October) iRobot Coporation: Our History. [Online].
<http://www.irobot.com/sp.cfm?pageid=203>
- [8] (2010, October) Evolution Robotics - NorthStar System Delivers Robotic Localization for Mobile Device Navigation or Robot Collaboration. [Online].
<http://www.evolution.com/products/northstar/>
- [9] (2010, March) The Cricket Indoor Location System: An NMS Project @ MIT CSAIL. [Online]. <http://cricket.csail.mit.edu/>
- [10] (2008, March) DARPA: Urban Challenge. [Online].
<http://www.darpa.mil/grandchallenge/index.asp>
- [11] Ben Crystall. (2007, July) New Scientist Blog: A programmable robot from 60 AD. [Online]. <http://www.newscientist.com/blog/technology/2007/07/programmable-robot-from-60ad.html>
- [12] (2010, October) Wikipedia: Teletank. [Online]. <http://en.wikipedia.org/wiki/Teletank>
- [13] (2010, October) Wikipedia: Goliath tracked mine. [Online].
http://en.wikipedia.org/wiki/Goliath_tracked_mine
- [14] Chris Trueman. (2010, October) The V Weapons. [Online].
http://www.historylearningsite.co.uk/v_weapons.htm

-
- [15] N Nilsson, "A MOBILE AUTOMATON: AN APPLICATION OF ARTIFICIAL INTELLIGENCE TECHNIQUES," in *Proceedings International Joint Conference on Artificial Intelligence*, Washington, 1969.
- [16] (2010, October) Wikipedia: Lunokhod Programme. [Online].
http://en.wikipedia.org/wiki/Lunokhod_program
- [17] Allan M. Thompson, "THE NAVIGATION SYSTEM OF THE JPL ROBOT," in *Proceedings of the 5th international joint conference on Artificial intelligence - Volume 2*, Cambridge, USA, 1977, pp. 749-757.
- [18] Hans P. Moravec, "Visual Mapping by a Robot Rover," in *Proceedings of the 6th international joint conference on Artificial intelligence - Volume 1*, Tokyo, Japan, 1979, pp. 598-600.
- [19] J. Matijevic, "MARS PATHFINDER MICROROVER - IMPLEMENTING A LOW COST PLANETARY MISSION EXPERIMENT," Jet Propulsion Laboratory, Pasadena, USA, IAA-L-0510, 1997.
- [20] Wolfram Burgard et al., "The Interactive Museum Tour-Guide Robot," in *Proceedings of the AAAI-98*, Madison, Wisconsin, USA, 1998.
- [21] Joel Chestnutt et al., "Footstep Planning for the Honda ASIMO Humanoid," in *Proceedings of the IEEE International Conference on Robotics and Automation*, 2005.
- [22] Hans Moravec, "Robots, After all," *Communications of the ACM*, vol. 46, no. 10, pp. 90 - 97, 2003.
- [23] Bill Gates, "A Robot in Every Home," *Scientific American*, no. January 2007, January 2007.
- [24] Erico Guizzo, "Three engineers, Hundreds of robots, One warehouse," *IEEE Spectrum*, July 2008.
- [25] (2003) NASA: A Robot to Help Make the Rounds. [Online].
http://www.sti.nasa.gov/tto/spinoff2003/hm_4.html
- [26] (2010, October) Aethon - Technology. [Online].
<http://www.aethon.com/how/technology.php>
- [27] Ali Gürcan Özkil et al., "Service Robots for Hospitals: A Case Study of Transportation Tasks in a Hospital," in *Proceedings of the IEEE International Conference on Automation and Logistics*, Shenyang, China, 2009, pp. 289 - 294.
- [28] Isaac Asimov, *I, Robot*. London, UK: HarperCollins, 1996.
- [29] Christoph Bartneck and Michio Okada, "Robotic User Interfaces," in *Proceedings of the Human and Computer Conference (HC2001)*, Aizu, Japan, 2001, pp. 130-140.

-
- [30] Kerstin Severinson-Eklundh, Green Anders, and Helge Hüttenrauch, "Social and collaborative aspects of interaction with a service robot," in *Proceedings of Robotics and Autonomous Systems 42 (2003)*, 2003, pp. 223–234.
- [31] (2010) HRI2010 - 5th ACM/IEEE International Conference on Human-Robot Interaction, March 2-5 2010, Osaka. [Online]. <http://hri2010.org/>
- [32] R. Siegwart and I. R. Nourbakhsh, *Introduction to Autonomous Mobile Robots*. Cambridge, Massachusetts: The MIT Press, 2004.
- [33] J. Borenstein, H.R. Everett, and L. Feng, *Where am I ? Sensors and Methods for Mobile Robot Positioning*. Ann Arbor, USA: University of Michigan, 1996.
- [34] Parag H. Batavia and Illah Nourbakhsh, "Path Planning for the Cye Personal Robot," in *Proceedings of IEEE/RSJ International Conference on Intelligent Robots and Systems, 2000. (IROS 2000)*, Takamatsu, Japan, 2000, pp. 15-20.
- [35] (2010, November) RMP 100 | Segway Robotic Mobility Platforms (RMPs). [Online]. <http://rmp.segway.com/rmp-100/>
- [36] (2009, June) MobileRobots Seekur outdoor research robot platform. [Online]. <http://www.mobilerobots.com/researchrobots/researchrobots/seekurugv.aspx>
- [37] F. Ribeiro, I. Moutinho, P. Silva, C. Fraga, and N. Pereira, "Three Omni-Directional Wheels Control on a mobile robot," in *Proceedings of the IEEE Control Conference 2004*, Bath, UK, 2004.
- [38] Olaf Fiegel, Aparma Badve, Glen Bright, Johan Potgieter, and Sylvester Tlale, "Improved Mecanum Wheel Design for Omni-directional robots," in *Proceedings of the Australasian Conference on Robotics and Automation*, Auckland, New Zealand, 2002, pp. 117-121.
- [39] J. Borenstein and L. Feng, "UMBMARK - A Method for Measuring, Comparing, and Correcting Dead-reckoning Errors in Mobile Robots," University of Michigan, Ann Arbor, 1994.
- [40] Edouard Ivanjko, Ivan Komsic, and Ivan Petrovic, "SIMPLE OFF-LINE ODOMETRY CALIBRATION OF DIFFERENTIAL DRIVE MOBILE ROBOTS," in *16th Int. Workshop on Robotics in Alpe-Adria-Danube Region (RAAD 2007)*, Ljubljana, 2007.
- [41] Evangelos Papadopoulos and Michael Misailidis, "On Differential Drive Robot Odometry with Application to Path Planning," in *Proceedings of the European Control Conference*, Kos, Greece, 2007, pp. 5492-5499.

-
- [42] J. Borenstein and L. Feng, "Gyrodometry: A New Method for Combining Gyros and Odometry in Mobile Robots," in *Proceedings of the 1996 IEEE International Conference on Robotics and Automation*, Mineapolis, USA, 1996, pp. 423-428.
- [43] M. Maimone, Y. Cheng, and L. Matthies, "Two years of Visual Odometry on the Mars Exploration Rovers," *Journal of Field Robotics*, vol. 24, no. 3, pp. 169–186, January 2007.
- [44] Janne Haverinen and Anssi Kemppainen, "Global indoor self-localization based on the ambient magnetic field," in *Robotics and Autonomous Systems Volume 57, Issue 10*, 2009, pp. 1028-1035.
- [45] (2010, November) Jane's Strategic Weapons Systems. [Online].
<http://www.janes.com/articles/Janes-Strategic-Weapon-Systems/LGM-30G-Minuteman-III-United-States.html>
- [46] H.S. Liu and G.K.H Pang, "Accelerometer for Mobile Robot Positioning," *IEEE Transactions on Industrial Applications*, vol. 37, no. 3, pp. 812-819, May / June 2001.
- [47] Hakyoung Chung, Lauro Ojeda, and Johann Borenstein, "Accurate Mobile Robot Dead-reckoning with a Precision-calibrated Fiber Optic Gyroscope," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 1, pp. 80-84, February 2001.
- [48] Johann Borenstein and Lauro Ojeda, "Heuristic reduction of gyro drift in vehicle tracking applications," *International Journal of Vehicle Autonomous Systems*, vol. 2, no. 1/2, pp. 78-98, 2009.
- [49] (2010, October) ART Product 2. [Online].
<http://www.robot techno.com/GyroCalibration.htm>
- [50] Valentin Magori, "Ultrasonic Sensors in Air," in *IEEE ULTRASONICS SYMPOSIUM*, 1994, pp. 471-481.
- [51] G Benet, F Blanes, J.E. Simó, and P. Pérez, "Using infrared sensors for distance measurement in mobile robots," in *Robotics and Autonomous Systems 40*, 2002, pp. 255-266.
- [52] K. Konolige, J. Augenbraun, N. Donaldson, and P. Shah, "A low-cost laser distance sensor," in *IEEE International Conference on Robotics and Automation (ICRA), 2008*, Pasadena, 2008, pp. 3002 - 3008.
- [53] (2010, October) Neato Robotics. [Online]. <http://www.neatorobotics.com/>
- [54] Todd Danko. Webcam Based DIY Laser Rangefinder. [Online].
http://sites.google.com/site/todddanko/home/webcam_laser_ranger

-
- [55] (2008, July) Lawn mowers – Eco-Friendly & Easy to use Lawn Mowers for the Perfect Lawn Maintenance Provided by Robomow. [Online]. <http://www.robomow.com/>
- [56] Cheng-Chih Lin and R. Lal Tummala, "Mobile Robot Navigation Using Artificial Landmarks," *Journal of Robotic Systems*, vol. 14, no. 2, pp. 93–106, 1997.
- [57] Hongbo Wang and Takazuku Ishimatsu, "Vision-Based Navigation for an Electric Wheelchair," *Journal of Intelligent and Robotic Systems Using Ceiling Light Landmark*, vol. 41, pp. 283–314, 2004.
- [58] Illah Nourbakhsh, Rob Powers, and Stan Birchfield, "DERVISH An Office-Navigating Robot," *AI Magazine*, vol. 14, no. 2, pp. 53-60, Summer 1995.
- [59] Dieter Fox, Wolfram Burgard, Frank Dellaert, and Sebastian Thrun, "Monte Carlo Localization: Efficient Position Estimation for Mobile Robots," in *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI'99)*, Orlando, USA, 1999, pp. 343-349.
- [60] M.W.M.G. Dissanayake, P. Newman, S. Clark, H.F. Durrant-Whyte, and M. Csorba, "A solution to the simultaneous localization and map building (SLAM) problem," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 229 - 241, June 2001.
- [61] (2007) Wikipedia: Global Positioning System. [Online]. http://en.wikipedia.org/wiki/Global_Positioning_System
- [62] (2010) The Global Positioning System. [Online]. <http://www.gps.gov/>
- [63] (2010) Wikipedia: GLONASS. [Online]. <http://en.wikipedia.org/wiki/GLONASS>
- [64] (2010) Wikipedia: Galileo (satellite navigation). [Online]. http://en.wikipedia.org/wiki/Galileo_%28satellite_navigation%29
- [65] (2009) GPS Magazine. [Online]. http://www.gpsmagazine.com/2009/10/gps_us_household_penetration.php
- [66] (2010, November) u-blox NEO-5 module series. pdf File.
- [67] Martin Peterzon, "Distribution of GPS-data via Internet," Geodetic Research Department at the Nation Land Survey of Sweden, Gävle, Sweden, Thesis 2004.
- [68] (2010, November) Navigation Services. [Online]. http://www.faa.gov/about/office_org/headquarters_offices/ato/service_units/techops/navigation/gnss/waas/
- [69] (2010, November) John Deere Greenstar Precision Farming. [Online]. http://www.deere.com/en_US/ag/online_brochures/greenstar/static/greenstar_zmags.html
- [70] (2010, November) Worldwide Differential Global Positioning Service - Omnistar USA, Inc. [Online]. <http://www.omnistar.com/>

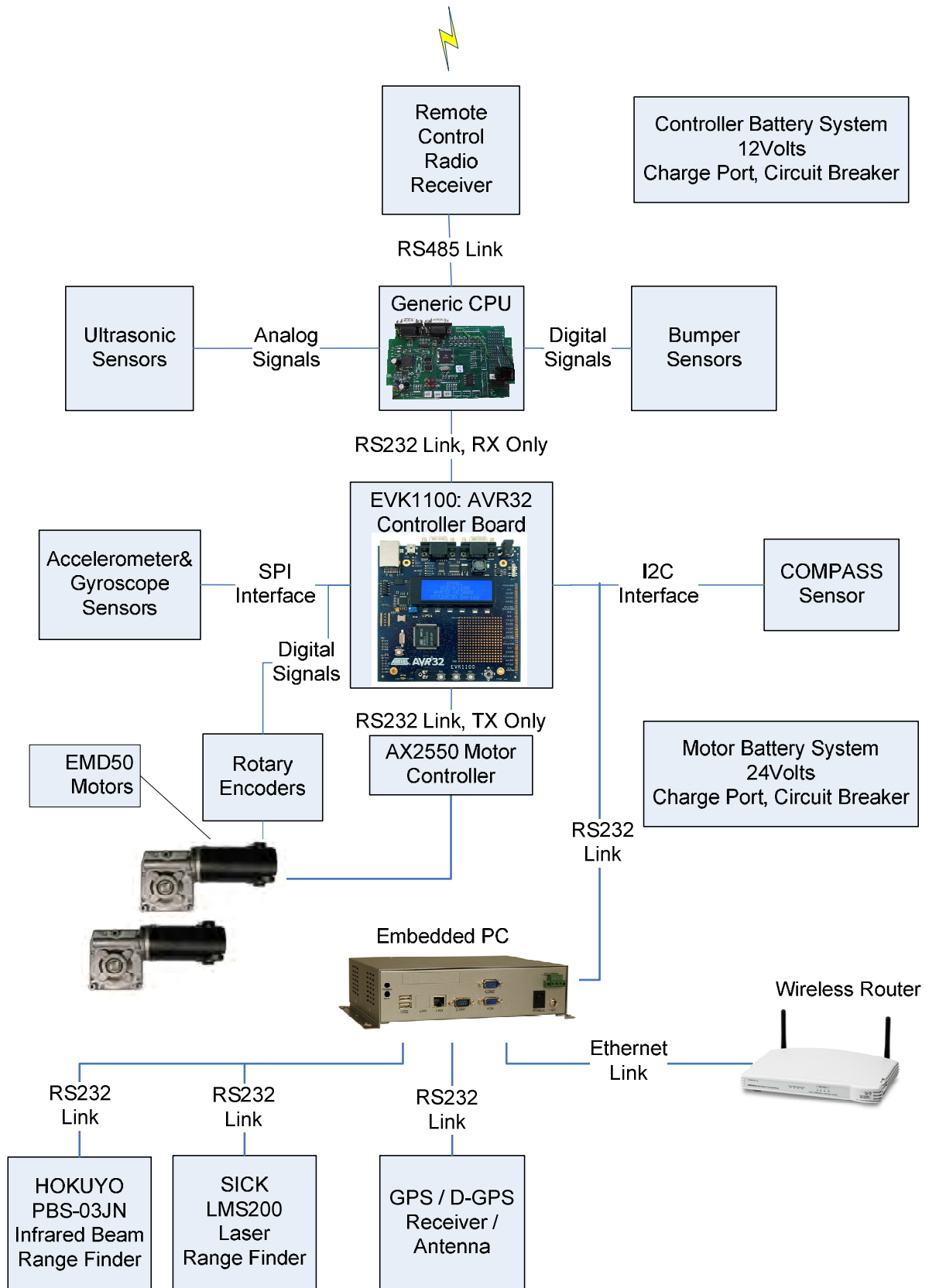
-
- [71] (2010, October) NASA - Mars Pathfinder and Sojourner. [Online].
http://www.nasa.gov/mission_pages/mars-pathfinder/
- [72] (2010, October) Automated / Automatic Guided Vehicle Systems (AGVs). [Online].
<http://www.jbtc-agv.com/>
- [73] (2010, October) Transbotics. [Online]. <http://www.transbotics.com/>
- [74] (2010, October) AGVs - Automated, Automatic Guided Vehicles. [Online].
<http://www.egeminusa.com/>
- [75] (2010, January) Micromouse UK Home. [Online]. <http://micromouse.cs.rhul.ac.uk/>
- [76] (2009, June) Robots with Motivity Deliver Remote Surveillance and Assistance on Demand. [Online].
<http://www.mobilerobots.com/RobotApplications/SecurityRobots.aspx>
- [77] Alexandre Lampe and Raja Chatila, "Performance Measure For The Evaluation of Mobile," in *Proceedings of the 2006 IEEE International Conference on Robotics and Automation*, Orlando, Florida USA, 2006, pp. 4057-4062.
- [78] Sebastian Thrun. (2010, October) Official Google Blog: What we're driving at. [Online].
<http://googleblog.blogspot.com/2010/10/what-were-driving-at.html>
- [79] F. Aurenhammer and R. Klein, "Voronoi Diagrams," in *In Handbook of Computational Geometry*, J. Sack and G. Urrutia, Ed.: Elsevier Science Publishing, 2000, ch. V, pp. 201-290.
- [80] (2010, October) Wikipedia: A* search algorithm. [Online].
http://en.wikipedia.org/wiki/A*_search_algorithm
- [81] E. W. Dijkstra, "Note on two problems in connexion with graphs," *Numerische Mathematik*, vol. 1, pp. 269–271, 1959.
- [82] V.J. Lumelski and A.A. Stepanov, "Dynamic Path Planning for a Mobile Automaton with Limited Information on the Environment," *IEEE TRANSACTIONS ON AUTOMATIC CONTROL*, vol. AC-31, no. 11, pp. 1057-1063, November 1986.
- [83] J. Borenstein and Y. Koren, "Real-time Obstacle Avoidance for Fast Mobile Robots in Cluttered Environments," in *Proceedings of the 1990 IEEE International Conference on Robotics and Automation*, Cincinnati, USA, 1990, pp. 572-577.
- [84] Javier Minguez and Luis Montano, "Nearness Diagram (ND) Navigation: Collision Avoidance in Troublesome Scenarios," *IEEE TRANSACTIONS ON ROBOTICS AND AUTOMATION*, vol. 20, no. 1, pp. 45-59, February 2004.

-
- [85] Issa A.D. Nesnas, Anne Wright, Max Bajracharya, Reid Simmons, and Tara Estlin, "CLARAty and Challenges of Developing Interoperable Robotic Software," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, Las Vegas, USA, 2003, pp. 2428 - 2435.
- [86] Michael Montemerlo, Nicholas Roy, and Sebastian Thrun, "Perspectives on standardization in mobile robot programming: The Carnegie Mellon navigation (CARMEN) toolkit," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS 2003)*, 2003, pp. 2436 - 2441.
- [87] Brian P. Gerkey et al., "Most Valuable Player: A Robot Device Server for Distributed Control," in *Proceedings of the Second International Workshop on Infrastructure for Agents, MAS, and Scalable MAS*, Wailea, Hawaii, 2001, pp. 1226-1231.
- [88] Brian P. Gerkey, Richard T. Vaughan, and Andrew Howard, "The Player/Stage Project: Tools for Multi-Robot and Distributed Sensor Systems," in *Proceedings of the International Conference on Advanced Robotics (ICAR 2003)*, Coimbra, Portugal, 2003, pp. 317-323.
- [89] (2007, May) GNU General Public License. [Online].
<http://www.gnu.org/licenses/gpl.html>
- [90] (2010, October) Wikipedia: Player Project. [Online].
http://en.wikipedia.org/wiki/Player_Project
- [91] Richard T. Vaughan and Brian P. Gerkey, "Really Reusable Robot Code and the Player/Stage Project," in *Software Engineering for Experimental Robotics*, D. Brugali, Ed.: Springer, 2006.
- [92] James Kramer and Matthias Scheutz, "Development environments for autonomous mobile robots: A survey," *Autonomous Robots*, vol. 22, no. 2, pp. 101-132, October 2007.
- [93] Morgan Quigley et al., "ROS: an open-source Robot Operating System," in *IEEE International Conference on Robotics and Automation (ICRA 2009)*, Kobe, Japan, 2009.
- [94] (2010, October) Willow Garage. [Online]. <http://www.willowgarage.com/>
- [95] Morgan Quigley, Eric Berger, and Andrew Y. Ng, "STAIR: Hardware and Software Architecture," in *Proceedings of Twenty-Second Conference on Artificial Intelligence (AAAI-07)*, Vancouver, Canada, 2007, pp. 31-37.
- [96] David P. Anderson. (2006, January) nBot Balancing Robot. [Online].
<http://www.geology.smu.edu/~dpa-www/robo/nbot/>
- [97] (2007, January) Segway Personal Transporter. [Online]. <http://www.segway.com/>

-
- [98] (2007, July) Dassault Systèmes: Solidworks Corp. [Online]. <http://www.solidworks.com/>
- [99] (2005, July) RS Components South Africa. [Online]. <http://za.rs-online.com/web/>
- [100] (2009, July) Nubotics. [Online]. <http://www.nubotics.com/products/ww02/index.html>
- [101] ABRAHAM SAVITZKY and MARCEL J.E. GOLAY, "Smoothing and Differentiation of Data by Simplified Least Squares Procedures," *Analytical Chemistry*, vol. 36, no. 8, pp. 1627-1639, July 1964.
- [102] (2010) Maxbotix. [Online]. http://www.maxbotix.com/MB1000_LV-MaxSonar-EZ0.html
- [103] Majd Alwan, Matthew B Wagner, Glenn Wasson, and Pradip Sheth, "Charaterisation of Infrared Range-Finder PBS-03JN for 2D Mapping," in *Proceedings of the 2005 IEEE International Conference on Robotics and Automation*, Barcelona, 2005, pp. 3947-3952.
- [104] (2010, April) RTKLIB: An Open Source Program Package for GNSS Positioning. [Online]. <http://gpspp.sakura.ne.jp/rtklib/rtklib.htm>
- [105] (2010, March) Reference Station Network. [Online]. <http://www.trignet.co.za/>
- [106] T Takasu and A Yasuda, "Development of the low-cost RTK-GPS receiver with an open source program package RTKLIB," in *International Symposium on GPS/GNSS*, Jeju, Korea, 2009.
- [107] Charles Merry, "NTRIP - the Future of Differential GPS ?," *PositionIT*, pp. 53-55, Jan/Feb 2007.
- [108] (2001, July) Atmel Corporation - Industry Leader in the Design and Manufacture of Advanced Semiconductors. [Online]. <http://www.atmel.com/>
- [109] (2007) Roboteq - AX2550. [Online]. <http://www.roboteq.com/brushed-dc-motor-controllers/ax2550-ax2850-dual-120a-brushed-dc-motor-controller.html>
- [110] (2010, February) Festival. [Online]. <http://www.cstr.ed.ac.uk/projects/festival/>
- [111] (2005, July) WINAVR. [Online]. <http://winavr.sourceforge.net/>
- [112] (2008, May) RFC1055, A NONSTANDARD FOR TRANSMISSION OF IP DATAGRAMS OVER SERIAL LINES: SLIP. [Online]. <http://tools.ietf.org/html/rfc1055>
- [113] (2010, October) FreeRTOS-A Free professional grade RTOS supporting 26 architectures, including ARM7, ARM9, Cortex-M3, RX600, MSP430, MicroBlaze, AVR, x86, PIC32, PIC24, dsPIC, H8S, HCS12 and 8051. [Online]. <http://www.freertos.org/>
- [114] (2005, July) The Modbus Organisation. [Online]. <http://www.modbus.org/>
- [115] (2007, July) The Player Project. [Online]. [The Player Project](http://www.player-project.org/)

- [116] Piet Terblanche. (2009) MMMSick Laser driver. C++ Builder program.
- [117] Piet Terblanche. (2009) ScanParse. C++ Builder program.
- [118] (2010, October) MapWindows Open Source GIS - Home Page. [Online].
<http://www.mapwindow.org/>
- [119] J. Borenstein and Y. Koren, "Histogramic In-Motion Mapping for Mobile Robot Obstacle Avoidance," *IEEE Journal of Robotics and Automation*, vol. 7, no. 4, pp. 535-539, 1991.
- [120] (2010, November) Mesa Imaging AG - SwissRanger SR4000 - miniature 3D time-of-flight range camera. 3D Camera. [Online]. <http://www.mesa-imaging.ch/>
- [121] (2010, March) Willow Garage Blog. [Online].
<http://www.willowgarage.com/blog/2010/04/27/reinventing-wheel>

Appendix A: Meercat Electrical System

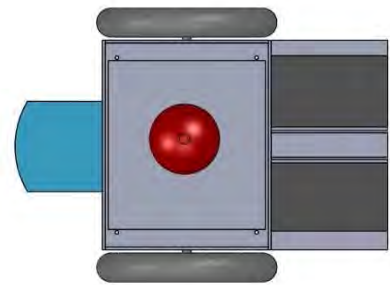
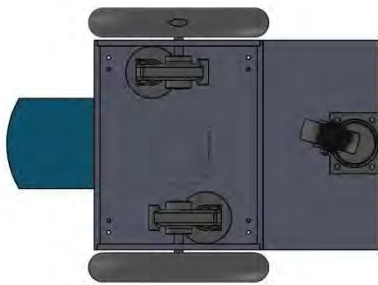
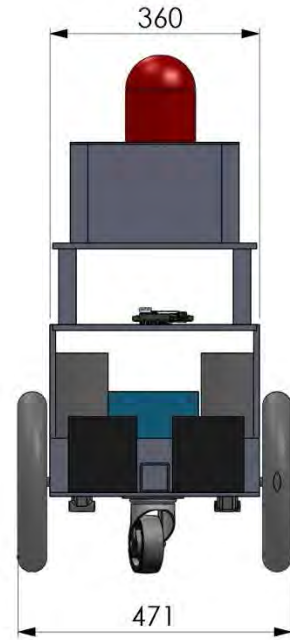
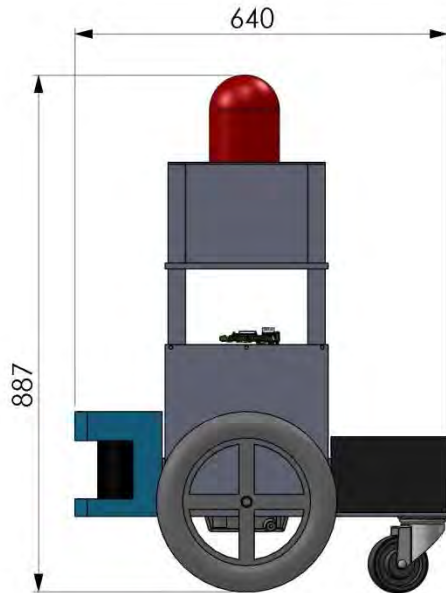


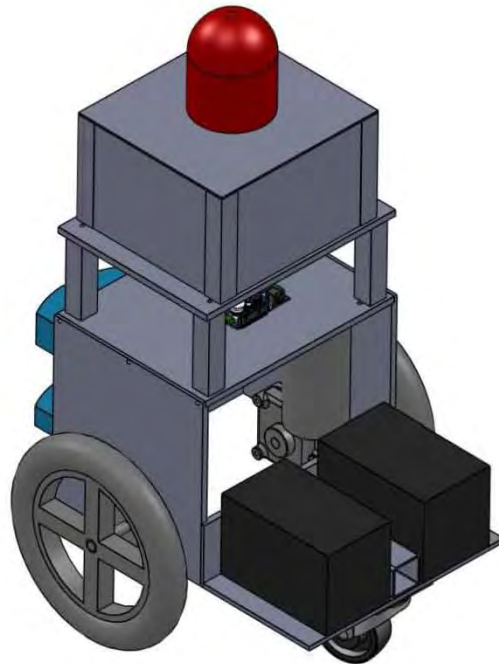
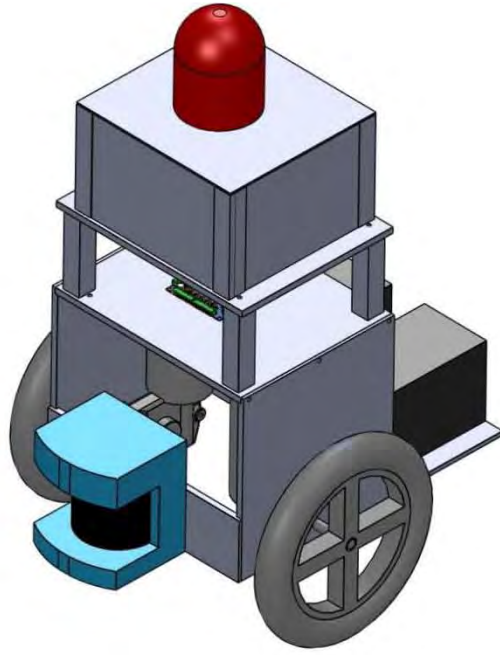
Appendix B: How Robotics Research keeps re-inventing the Wheel



Source: [121]

Appendix C: Meercat Mechanical Drawings

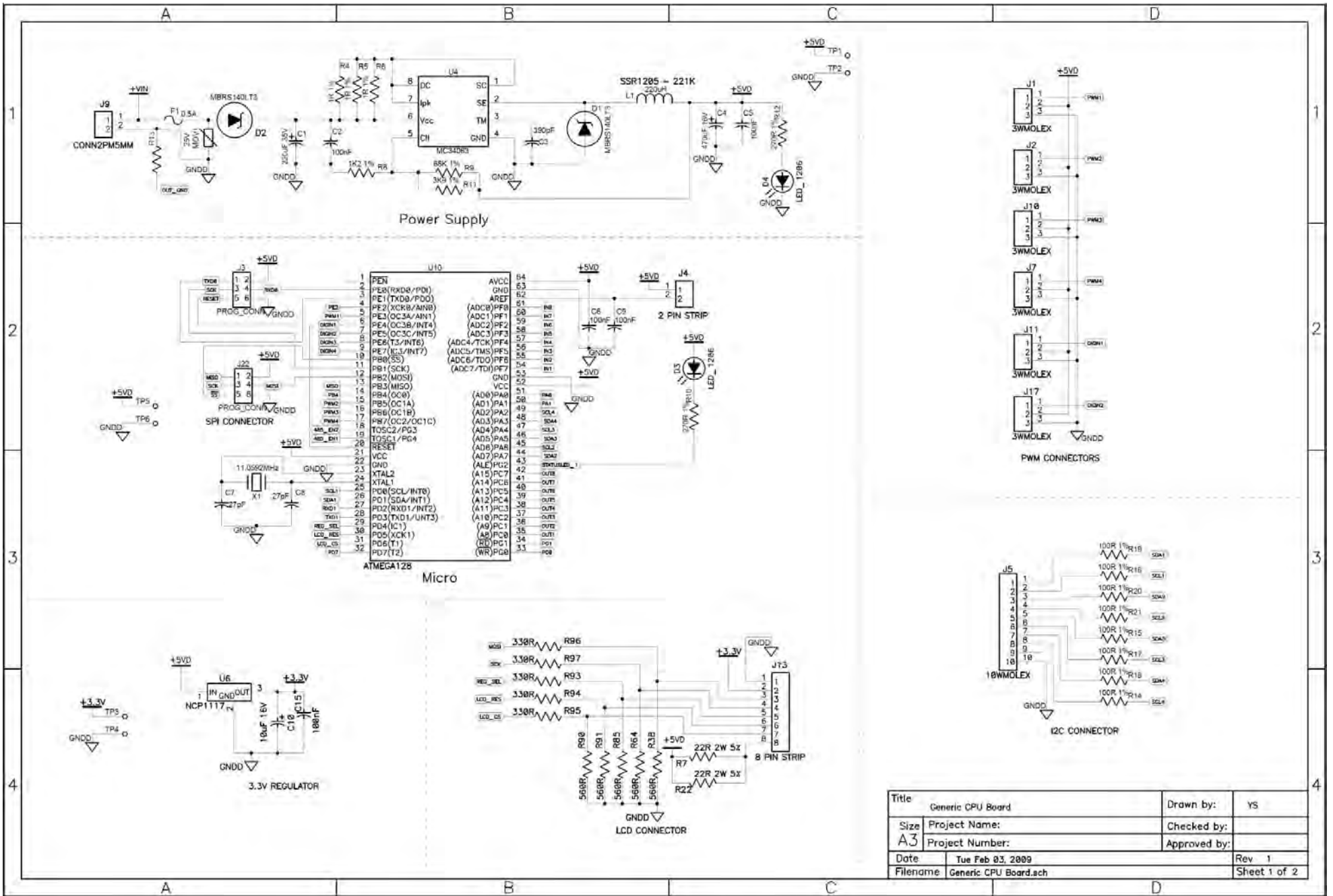




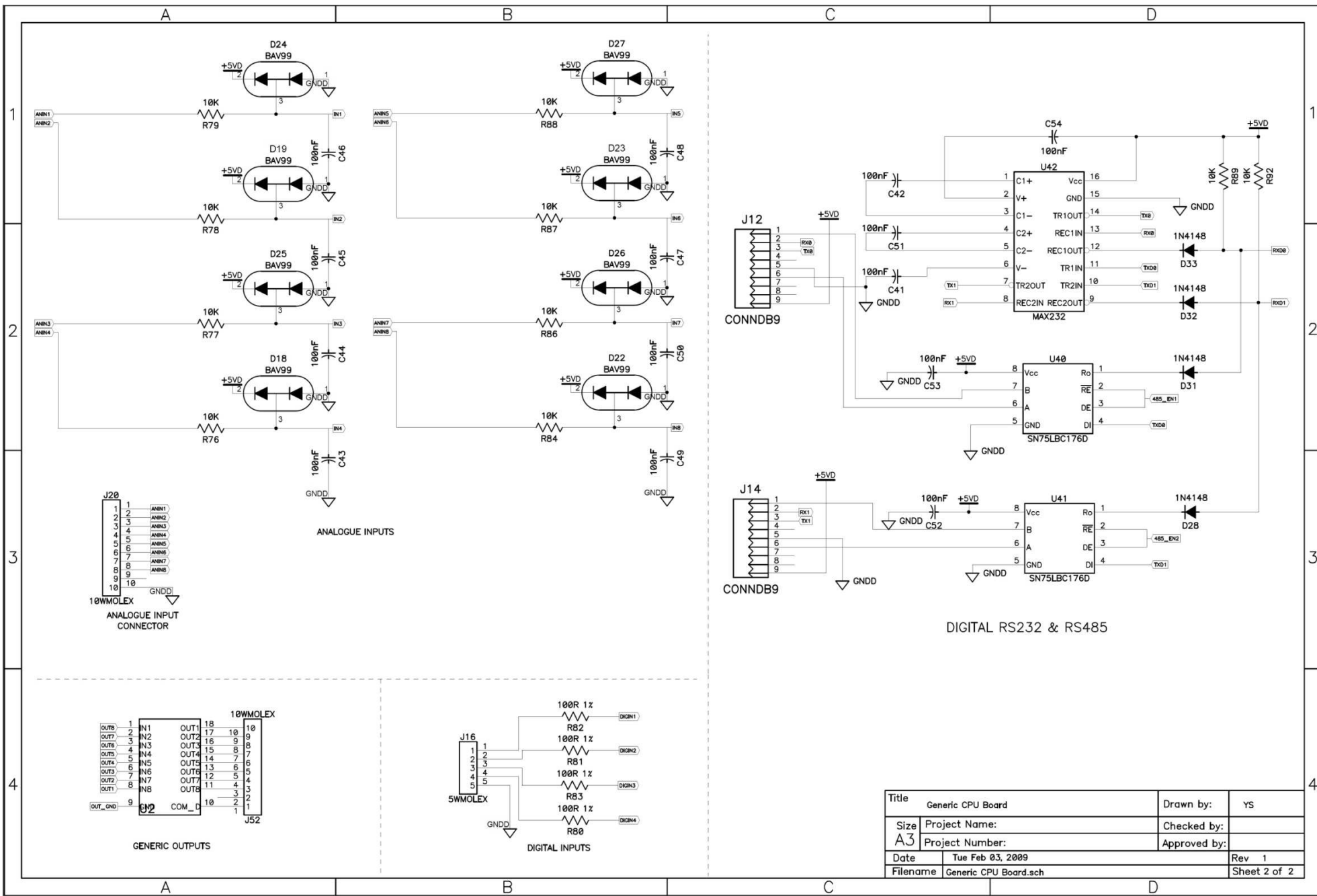
Appendix D: “Generic CPU Board Schematic”

This appendix shows on the following two pages the schematic design of the Generic CPU board that was used on the Meercat Robot for interfacing to basic sensors.

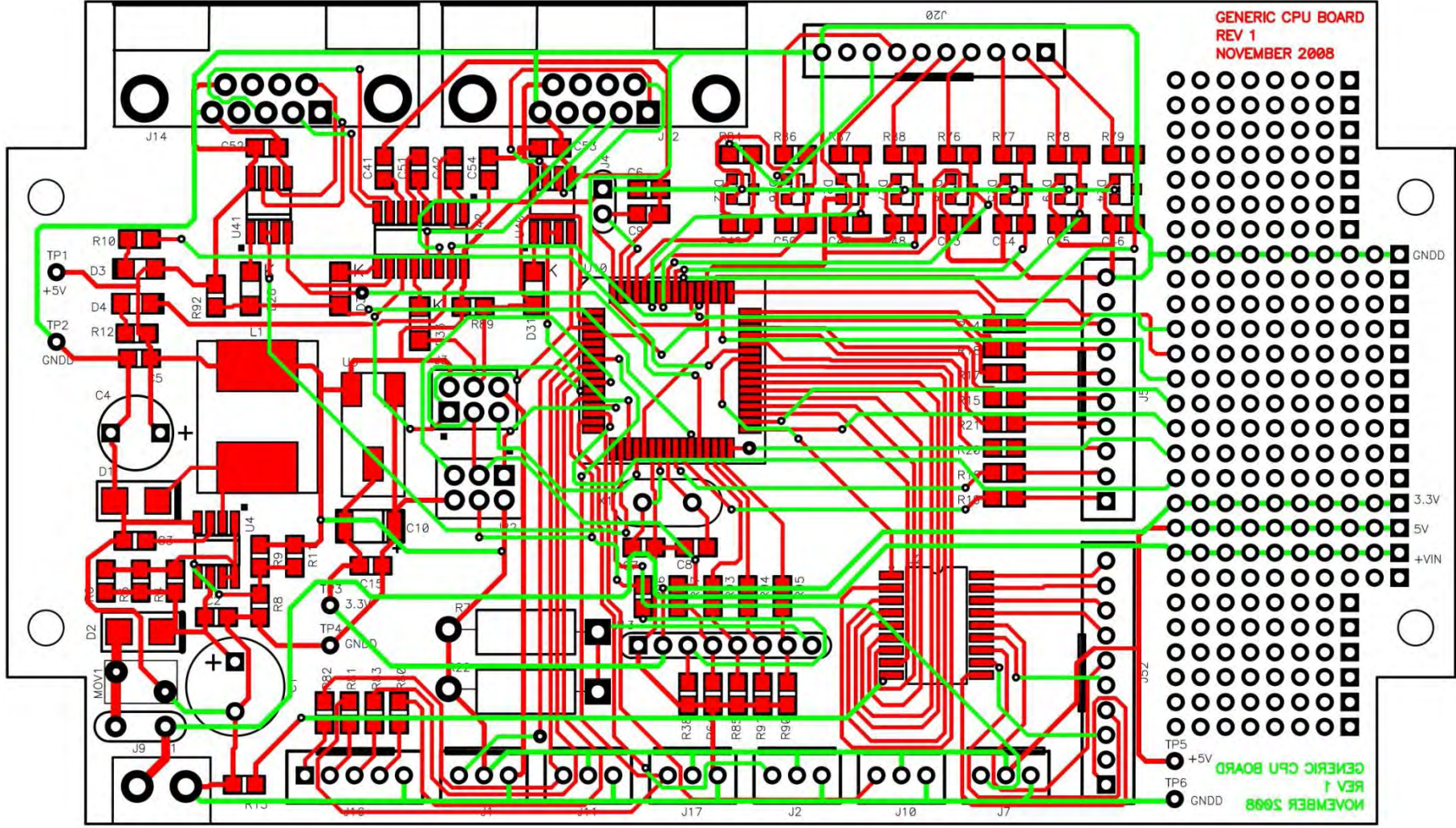
The last page of this appendix shows the double sided Printed Circuit Board layout.



Title	Generic CPU Board	Drawn by:	YS
Size	Project Name:	Checked by:	
A3	Project Number:	Approved by:	
Date	Tue Feb 03, 2009		Rev 1
Filename	Generic CPU Board.sch		Sheet 1 of 2



Title	Generic CPU Board	Drawn by:	YS
Size	Project Name:	Checked by:	
A3	Project Number:	Approved by:	
Date	Tue Feb 03, 2009		Rev 1
Filename	Generic CPU Board.sch		Sheet 2 of 2

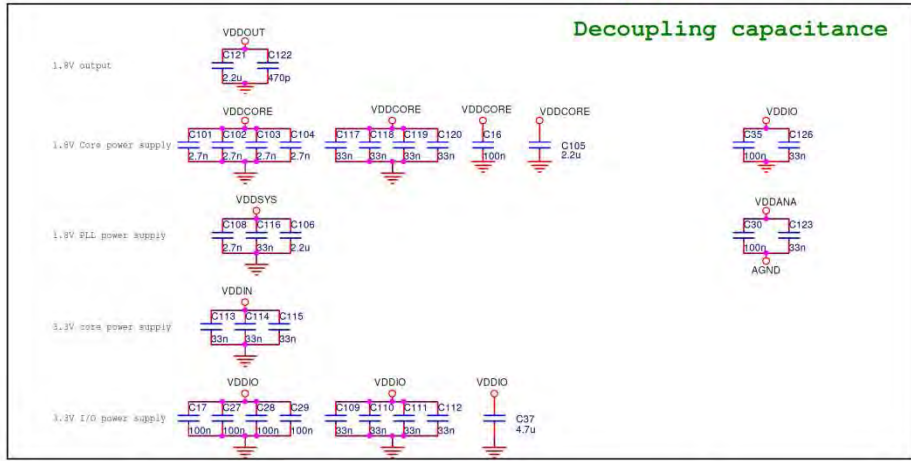
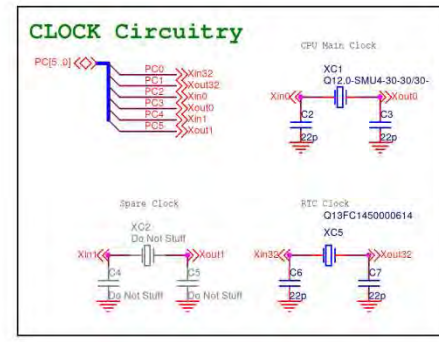
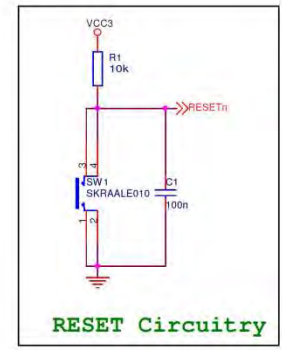
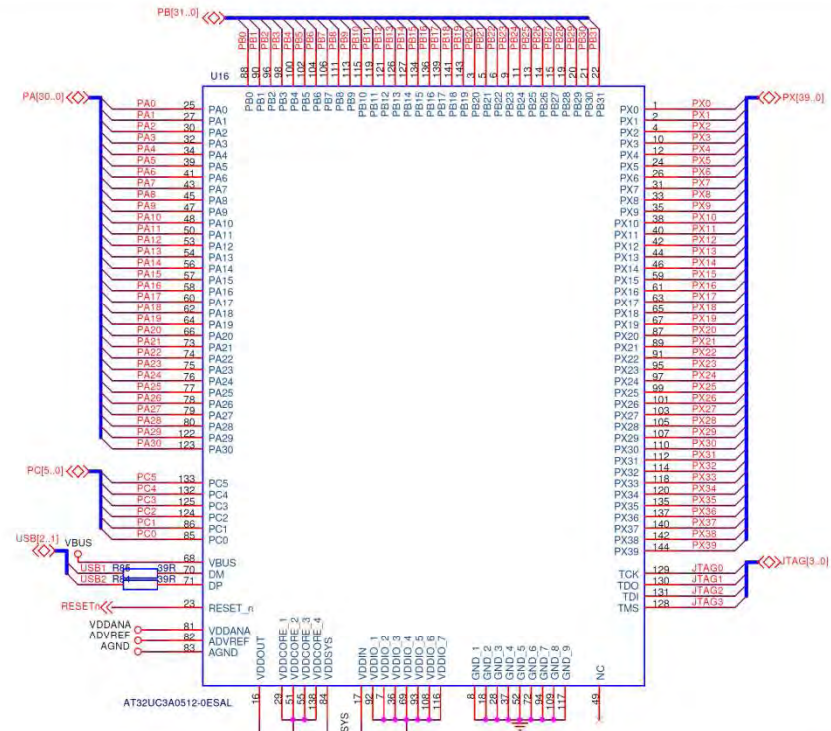


Appendix E: “AVR32 CPU Board Schematic”

This appendix shows on the following thirteen pages the schematic diagrams of the AVR32-UC3A based ATMEL EVK1100 board, the central embedded controller for the Meercat Mobile Robot.

Specific connections to this board can be found in the file ‘avr32-EVK1100-pinouts.xls’ under the ‘\AVR32 - UC3A\Hardware’ directory on the provided data disk.

(NM : Not Mounted)



born-MassProduction

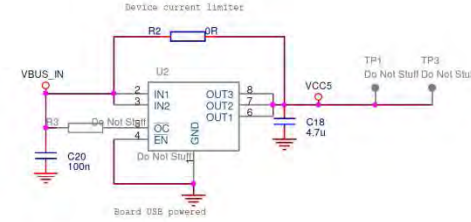
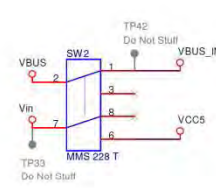
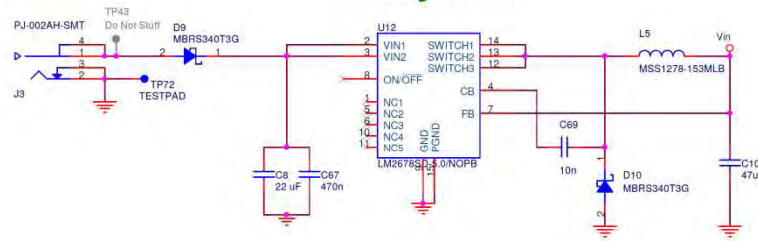
ATMEL Nantes SA
La Chantrerie BP 70602
44306 Nantes Cedex 3
FRANCE

Title	EVK1100	
Size	Document Number	PROCESSOR
Date:	Friday, July 13, 2007	Sheet 1 of 13

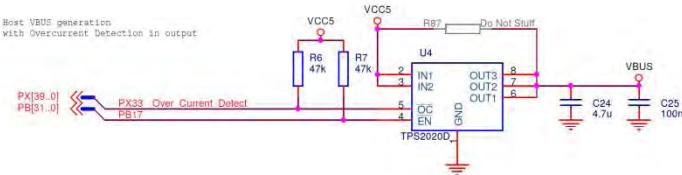


(NM : Not Mounted)

5V Regulator

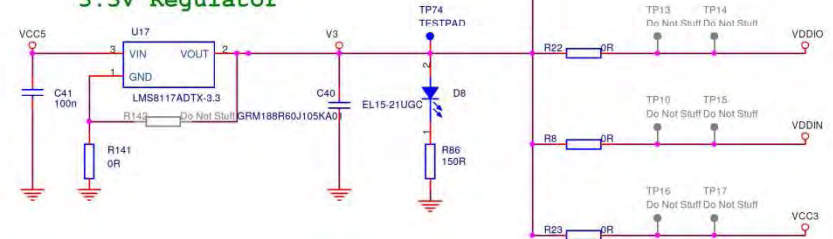


Host VBUS generation with Overcurrent Detection in output

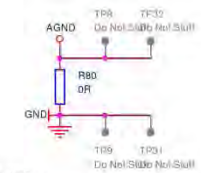


Shall only be added if TP82020B

3.3V Regulator



VDDANA intends to provide a clean 3.3V power supply for analog blocks
 VDDIO intends to provide a clean 3.3V power supply for ATUC3A0512 buffers
 VDDIN intends to provide a clean 3.3V power supply for ATUC3A0512 internal regulator
 VCC3 intends to provide a clean 3.3V power supply for all peripherals



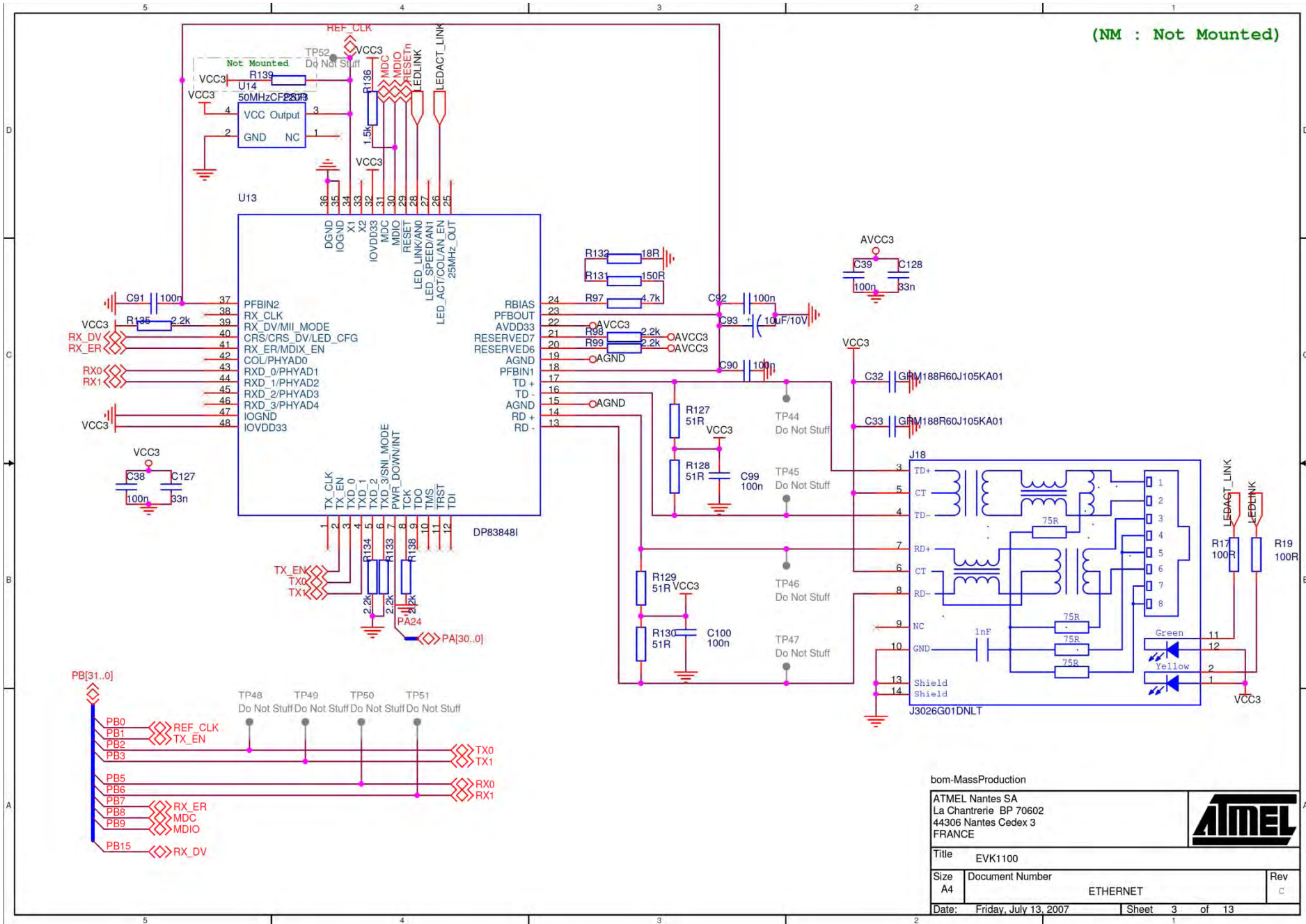
born-MassProduction

ATMEL Nantes SA
 La Chanterie BP 70602
 44306 Nantes Cedex 3
 FRANCE



Title	EVK1100		Rev	
Size	Document Number		POWER	
A3				
Date	Friday, July 13, 2007	Sheet	2 of 19	

(NM : Not Mounted)



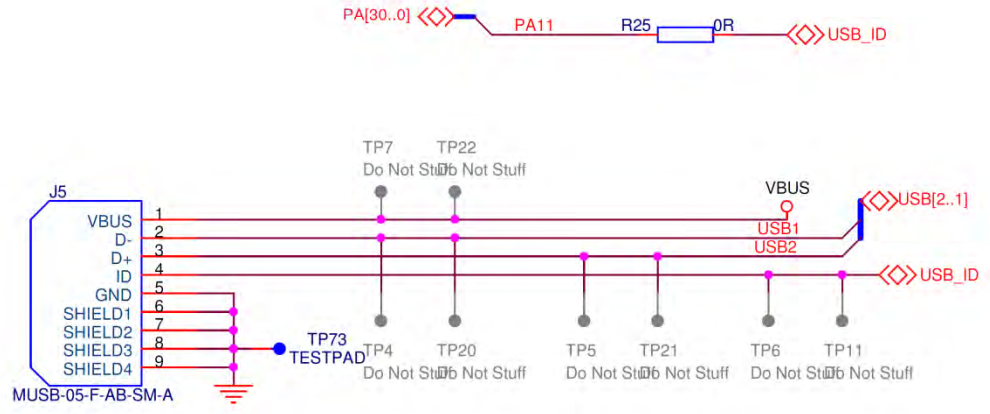
bom-MassProduction

ATMEL Nantes SA
 La Chantrerie BP 70602
 44306 Nantes Cedex 3
 FRANCE



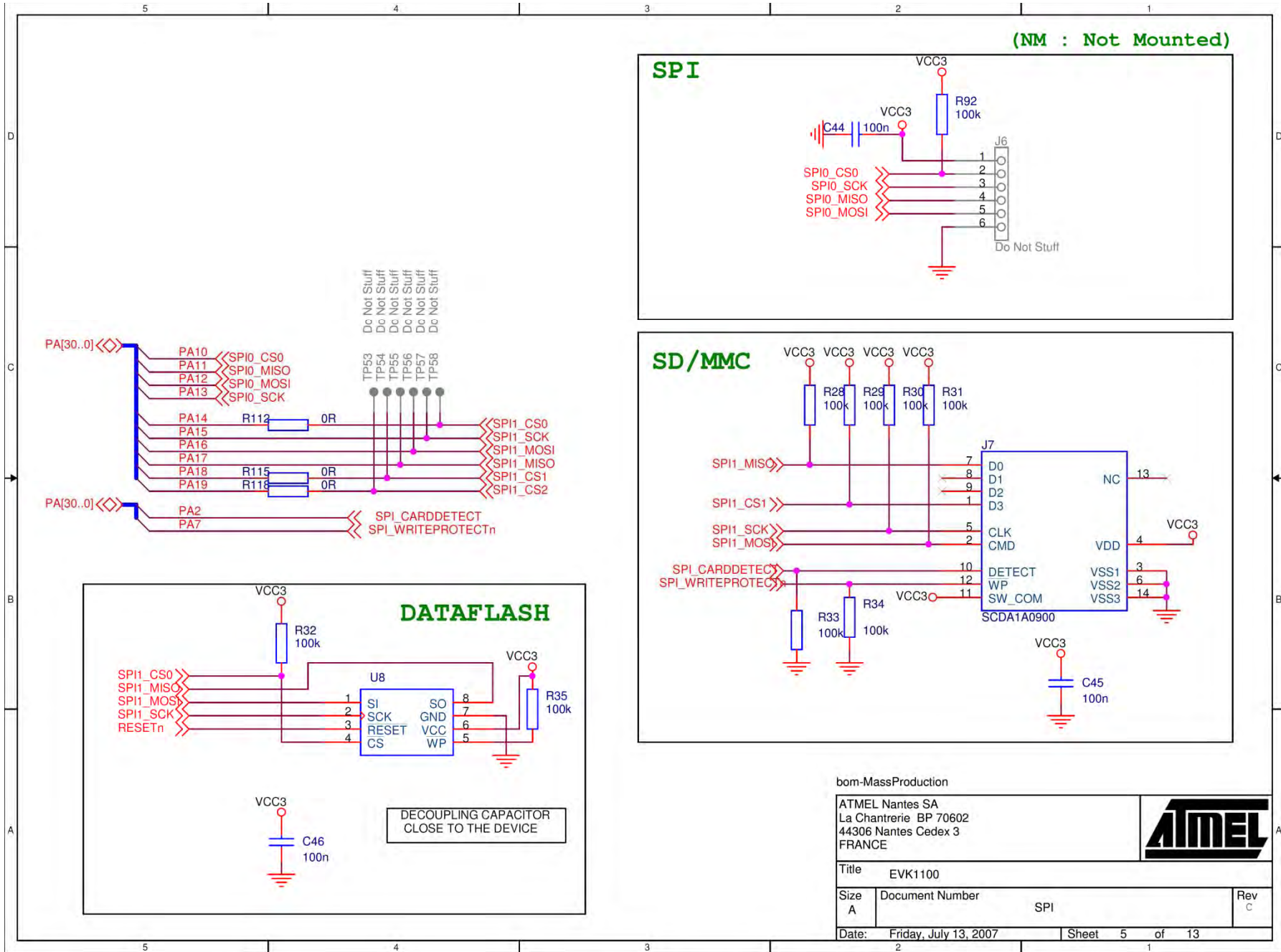
Title	EVK1100		Rev	C
Size	A4	Document Number	ETHERNET	
Date:	Friday, July 13, 2007	Sheet	3	of 13

(NM : Not Mounted)

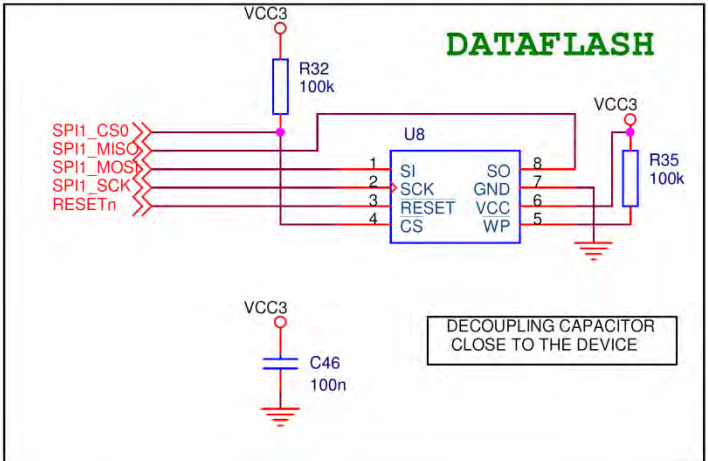
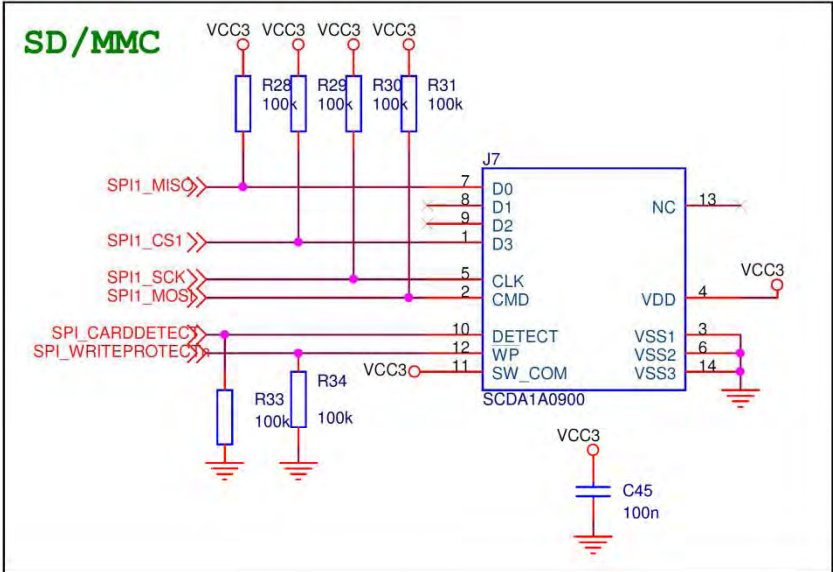
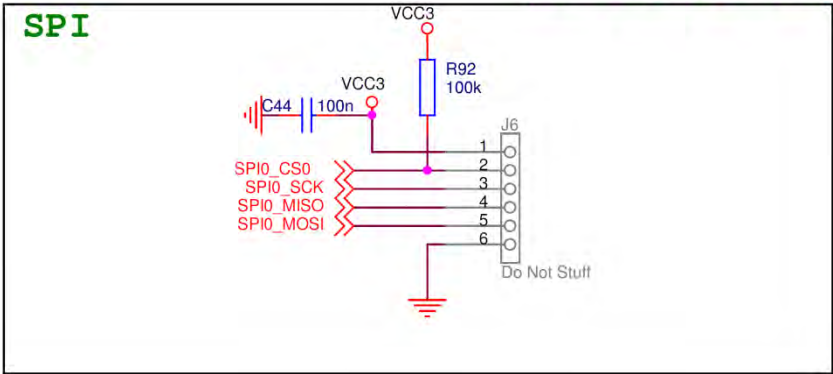


122

bom-MassProduction		
ATMEL Nantes SA La Chantrerie BP 70602 44306 Nantes Cedex 3 FRANCE		
Title EVK1100		
Size A	Document Number USB	Rev C
Date: Friday, July 13, 2007	Sheet 4 of 13	

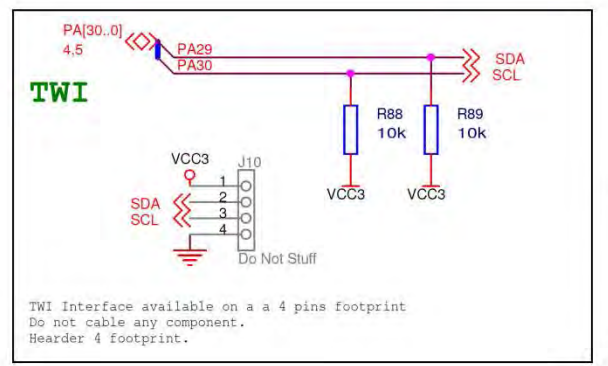
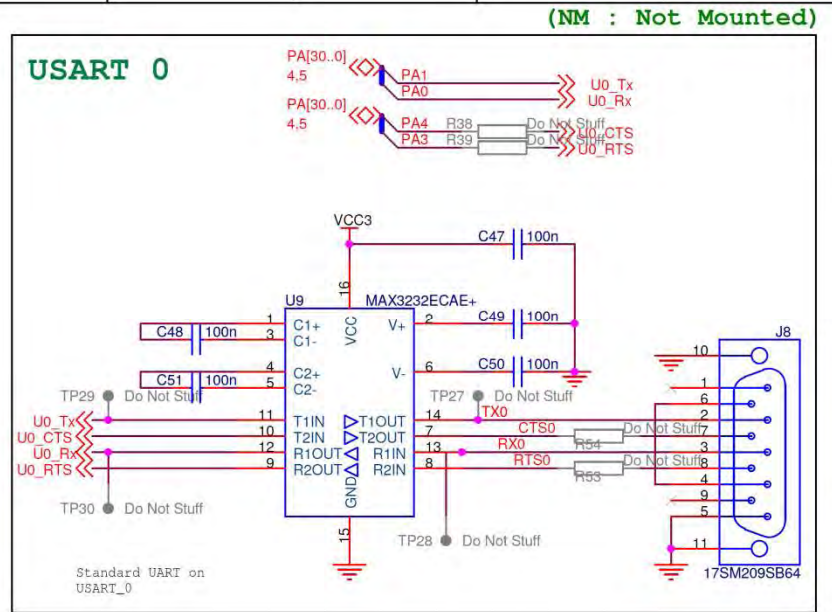
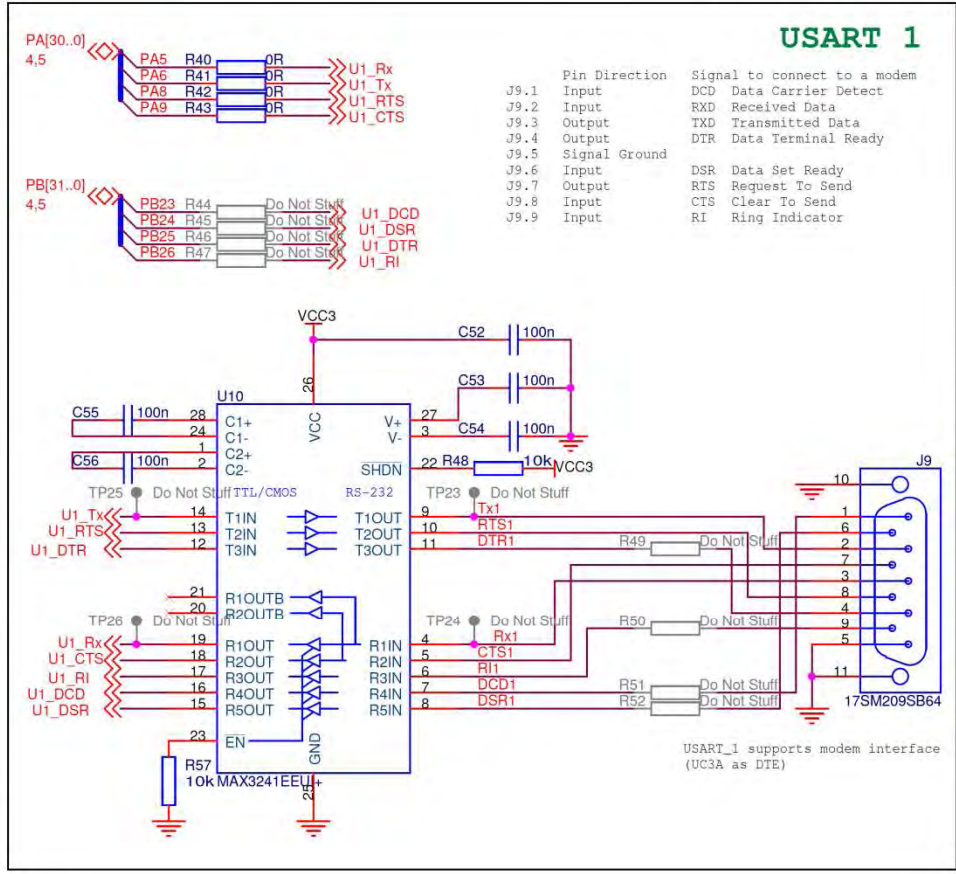


(NM : Not Mounted)



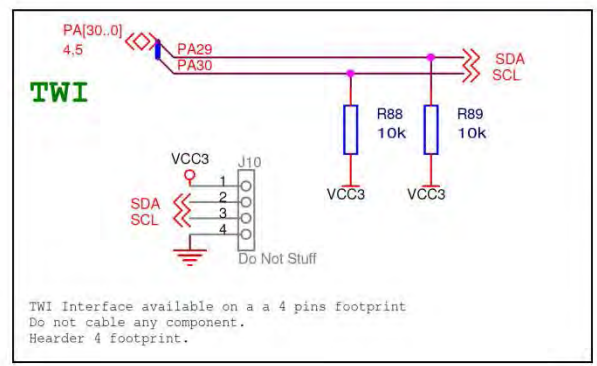
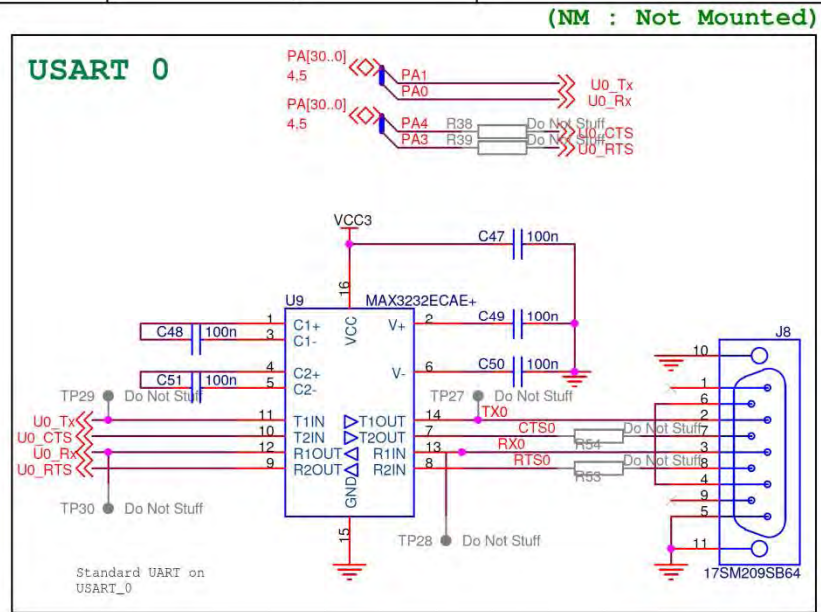
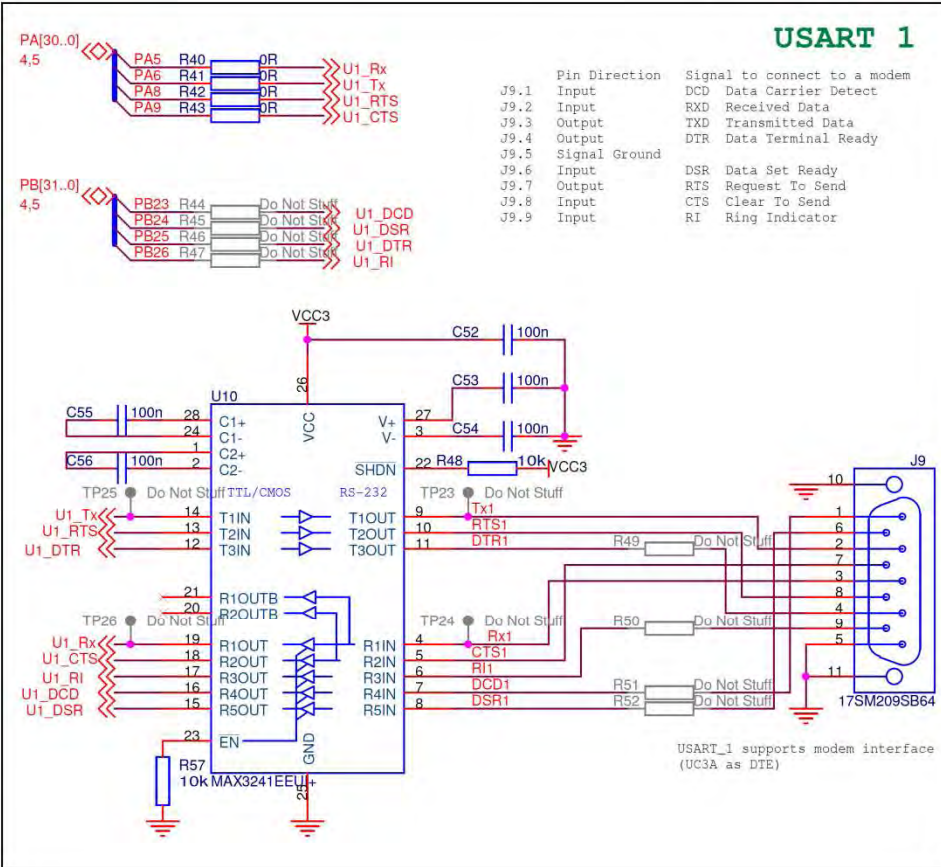
bom-MassProduction

ATMEL Nantes SA La Chantrerie BP 70602 44306 Nantes Cedex 3 FRANCE		
Title	EVK1100	
Size	Document Number	Rev
A	SPI	C
Date:	Friday, July 13, 2007	Sheet 5 of 13



bom-MassProduction		
ATMEL Nantes SA La Chantrerie BP 70602 44306 Nantes Cedex 3 FRANCE		
Title		EVK1100
Size	Document Number	USART
A4		Rev C
Date:	Friday, July 13, 2007	Sheet 6 of 13

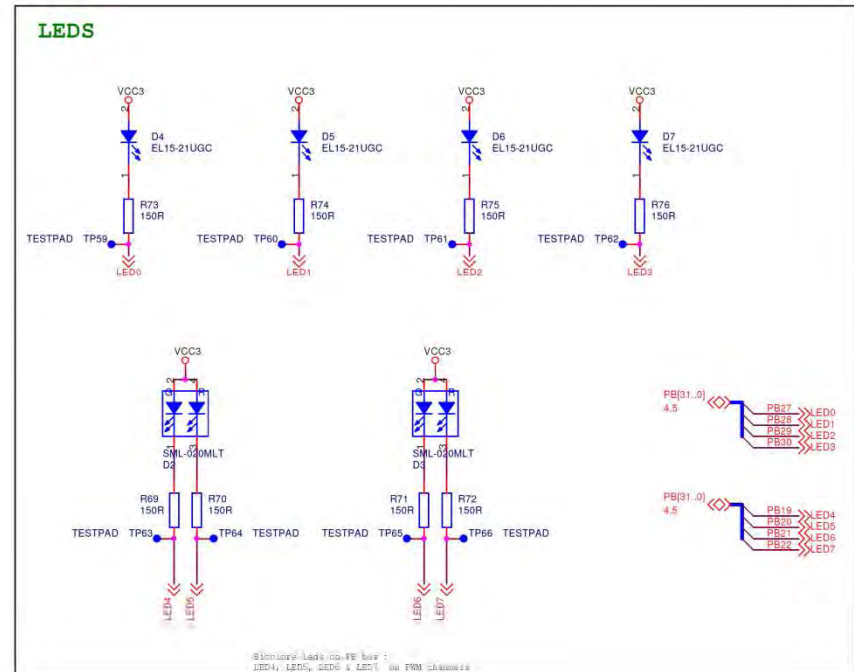
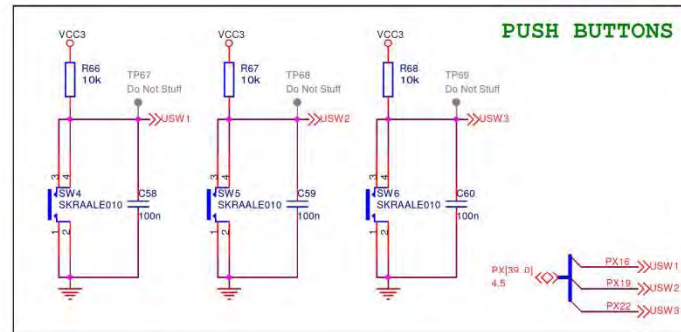
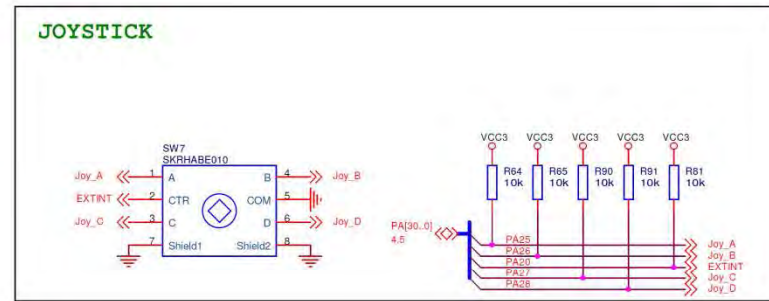
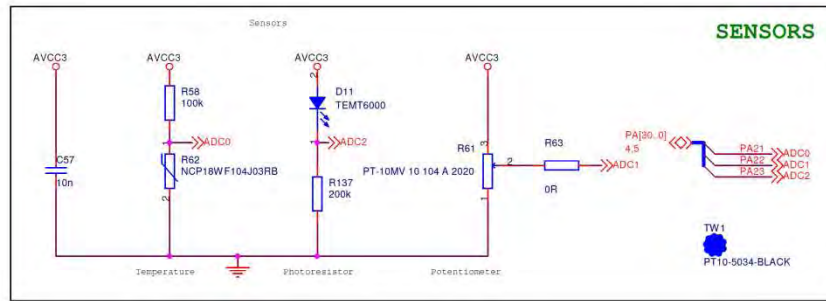




bom-MassProduction		
ATMEL Nantes SA La Chantrerie BP 70602 44306 Nantes Cedex 3 FRANCE		
Title		EVK1100
Size	Document Number	USART
A4		Rev C
Date:	Friday, July 13, 2007	Sheet 6 of 13



(NM : Not Mounted)

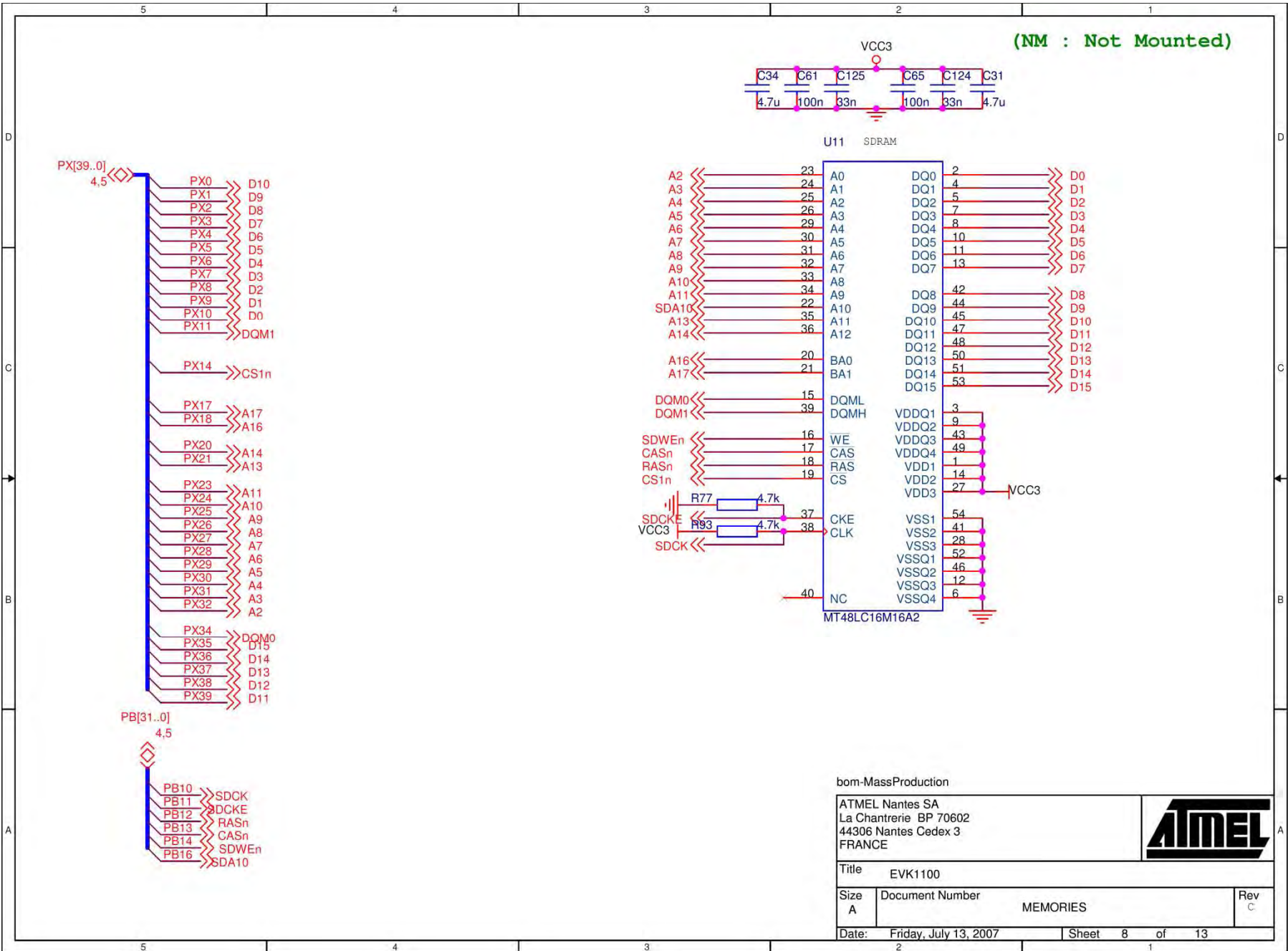


bom-MassProduction

ATMEL Nantes SA
La Chantrerie BP 70602
44306 Nantes Cedex 3
FRANCE




Title	EVK1100		Rev
Size	Document Number	SENSORS	
Date	Friday, July 13, 2007	Sheet	7 of 13



born-MassProduction

ATMEL Nantes SA
La Chantrerie BP 70602
44306 Nantes Cedex 3
FRANCE

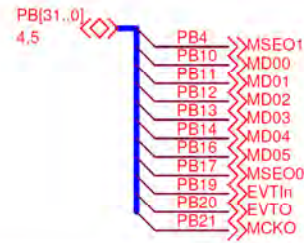


Title EVK1100

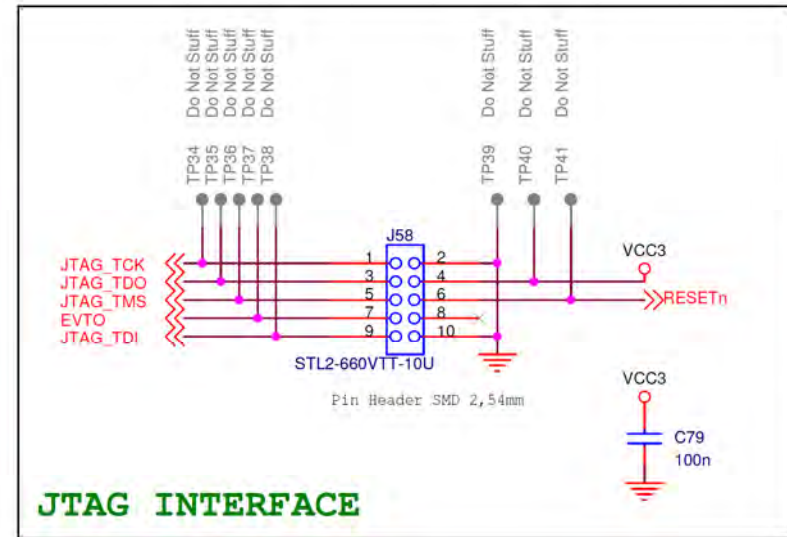
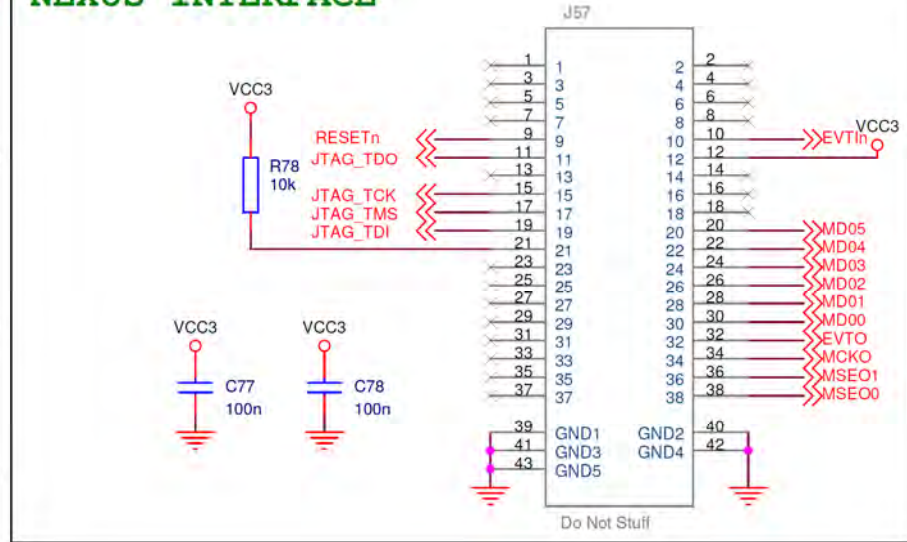
Size A	Document Number	MEMORIES	Rev C
--------	-----------------	----------	-------

Date: Friday, July 13, 2007 Sheet 8 of 13

(NM : Not Mounted)



NEXUS INTERFACE



JTAG INTERFACE

bom-MassProduction

ATMEL Nantes SA
La Chantrerie BP 70602
44306 Nantes Cedex 3
FRANCE



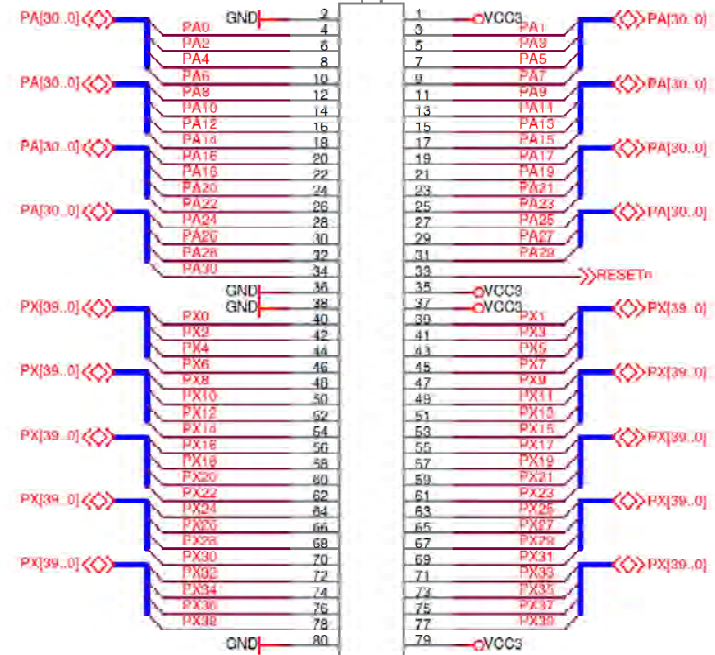
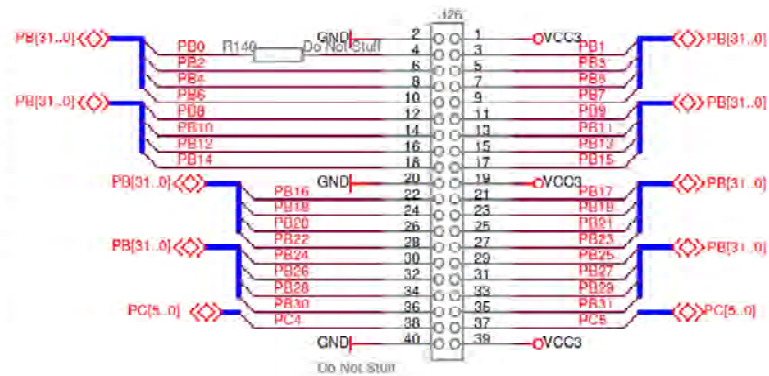
Title		EVK1100	
Size	Document Number	MEMORIES	Rev
A			C
Date:	Friday, July 13, 2007	Sheet	9 of 13

(NM : Not Mounted)

No component shall be implemented for this connector.
Only a footprint of Header 1.27mm shall be present on PCB

This connector allows easy connection of peripheral in the prototyping area

J27



born-MassProduction

ATMEL Nantes SA
La Chantrerie BP 70602
44306 Nantes Cedex 3
FRANCE



Title EVK1100

Size A4 Document Number

EXPANSION

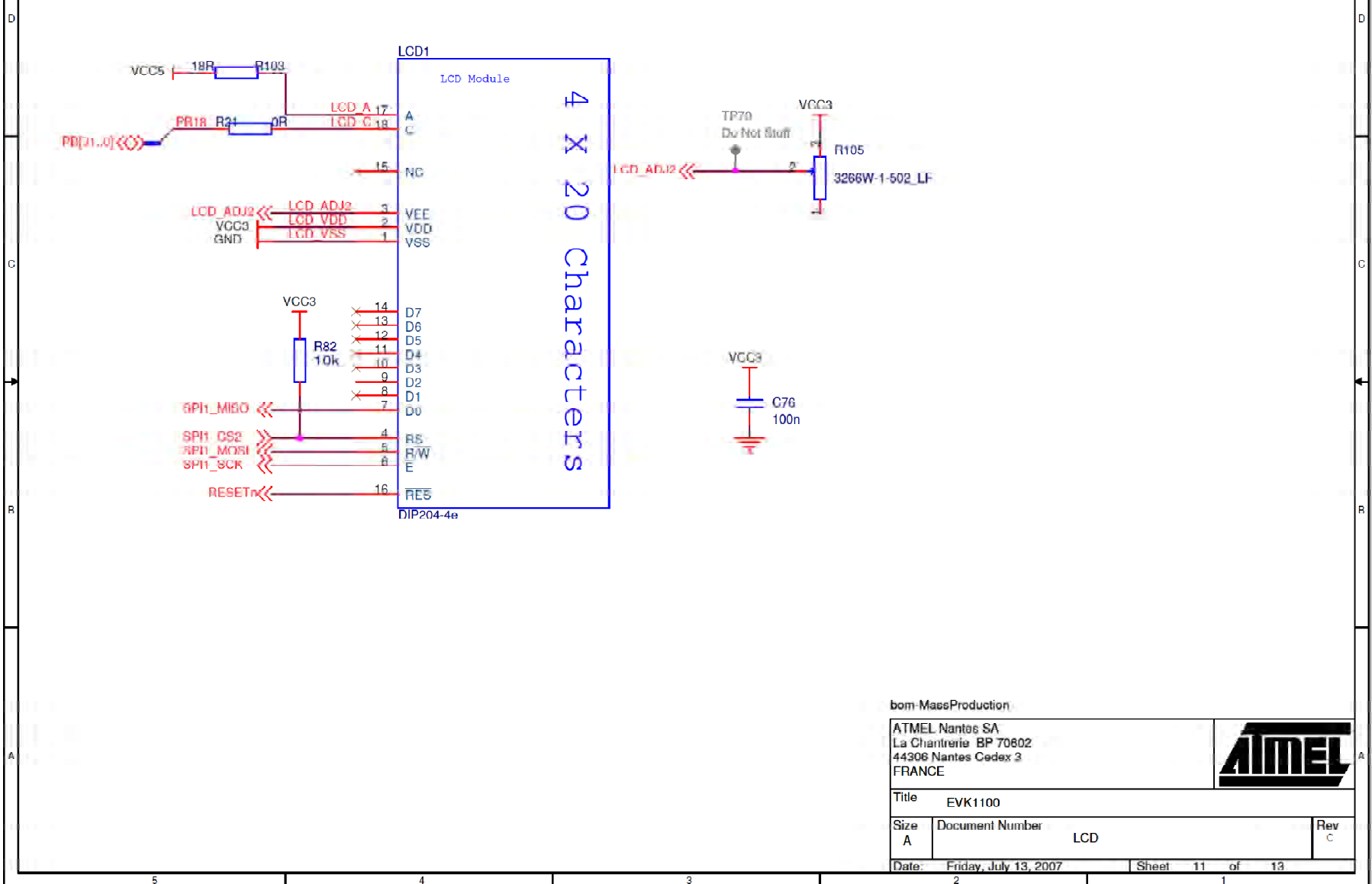
Rev c

Date: Friday, July 13, 2007

Sheet 10 of 13

(NM : Not Mounted)

EA-DTP204-4 Blue white LCD
KS0073 controller



bom-MassProduction

ATMEL Nantes SA
La Chantretrie BP 70602
44306 Nantes Cedex 3
FRANCE



Title EVK1100

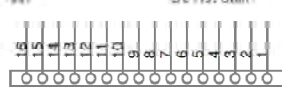
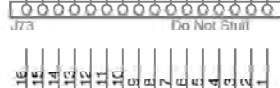
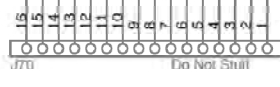
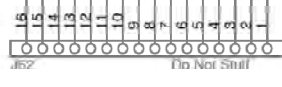
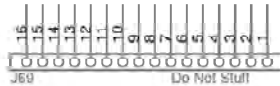
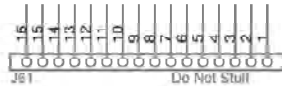
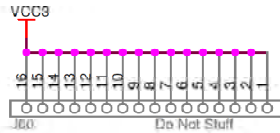
Size A Document Number LCD

Rev C

Date: Friday, July 13, 2007 Sheet 11 of 13

(NM : Not Mounted)

24 lines of array
+ 1 line of VCC
+ 1 line of VSS



E13
SJ-5076



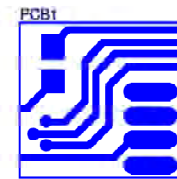
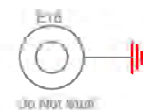
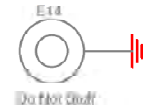
E12
SJ-5076



E11
SJ-5076



E10
SJ-5076



ATMEL Nantes SA
La Chantrerie BP 70802
44306 Nantes Cedex 3
FRANCE



Title		EVK1100	
Size	Document Number	PROTO	Rev
A4			C
Date:	Friday, July 13, 2007	Sheet	12 of 13

ERRATA

REV C/

USART 0

U0_CTS (PA.4) must be connected to U9.9
U0_RTS (PA.3) must be connected to U9.10
U9.8 (R2IN) must be connected to J8.7
U9.7 (T2OUT) must be connected to J8.8

USART 1

U10.11 (DTR1) must be connected to J9.6
U10.8 (DSR1) must be connected to J9.11

LEDS

LED 5/Green is connected to GPIO PB.19 and not to PB.20
LED 5/Red is connected to GPIO PB.20 and not to PB.19
LED 6/Green is connected to GPIO PB.21 and not to PB.22
LED 6/Red is connected to GPIO PB.22 and not to PB.21

SPI

Wrong Silkscreen : MISO and SCK are swaped

ATMEL Nantes SA
La Chantrerie BP 70602
44306 Nantes Cedex 3
FRANCE



Title EVK1100

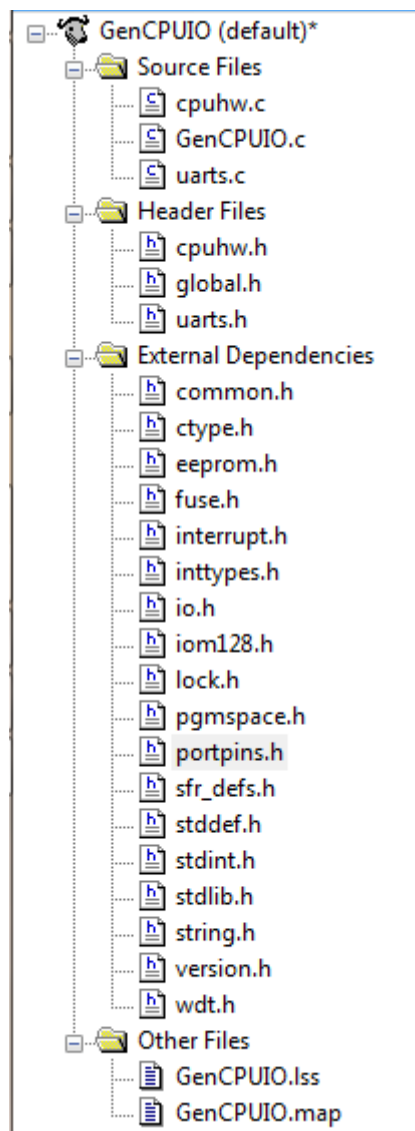
Size A Document Number ERRATA Rev C

Date: Thursday, July 12, 2007 Sheet 13 of 13

Appendix F: “Generic CPU Project Definition and Files”

This appendix shows the project definition and files for the Generic CPU Project.

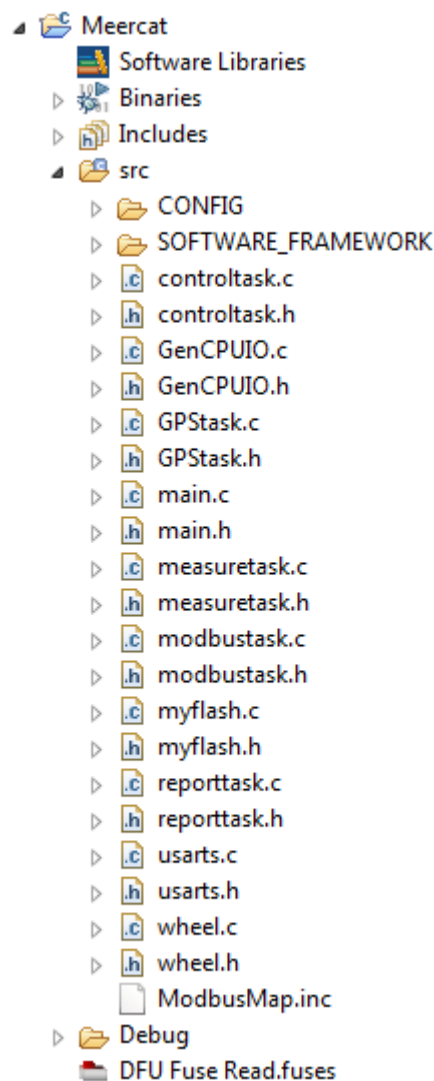
The files listed under ‘Source Files’ and ‘Header Files’ are part of the source code for the Generic CPU project on the Meercat Mobile robot. These files and the project definition can be found on the accompanying data disk under the \Generic CPU\Software directory.



Appendix G: “AVR32 Project Definition and Files”

This appendix shows the project definition and files for the AVR32 Meercat Project.

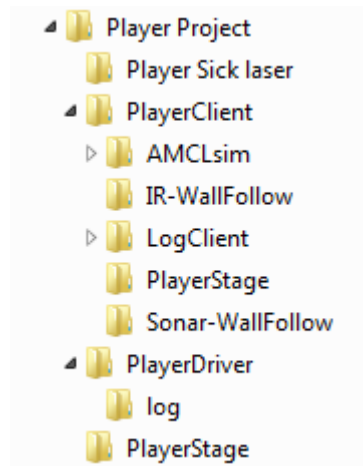
The CONFIG and SOFTWARE_FRAMEWORK sub-directories and their contents are part of the ATMEL tools. The rest of the listed files under ‘src’ are part of the work performed for the Meercat project. These files and the project definition can be found on the accompanying data disk under the \AVR32 - UC3A\Software directory.



Appendix H: “Player Driver & Client Files”

The software related to the Player software framework is stored on the data disk under the directory ‘Player Project’. A number of versions of clients and drivers exist, and these are all contained within the directory structure.

The structure of the directories is as follows:



For the Meercat driver files the following files are of importance:

libSenseCtrl.so	: the compiled driver code
SenseCtrl.c	: core ‘c’ code for the driver
Modbus-Support.c	: communications module ‘c’ code
makefile	: used to compile the source code
Meercat.cfg	: Player Configuration file for Meercat
Meercat-SickLaser.cfg	: Alternate Player Configuration file for Meercat
B14F-East-Plan-10cm.png	: Map for indoor only experimentation
B14F-E-InOut-10cm.png	: Map for combined indoor/outdoor experimentation

For the client files the following files are of importance:

client.c	: ‘c’ code for a client
makefile	: used to compile the source code
Meercat.scr	: file describing scripted moves