Durham University

School of Engineering

Cooperative Mobile Robots

Paul M Furley

April 2009

Contents

1 Introduction & Literature Review **15**

1.1 Self-Navigation . . . 15

1.2 Applications of Self-Navigation . . . 16

1.2.1 Hazardous Environments . . . 16

1.2.2 Search & Rescue . . . 16

1.2.3 Health care . . . 17

1.2.4 Industry . . . 17

1.2.5 Domestic . . . 18

1.2.6 Consumer automobiles . . . 19

1.3 Multiple, Communicating Robots . . . 20

1.3.1 Redundancy . . . 20

1.3.2 Improves 'near-sightedness' . . . 20

1.3.3 Speed . . . 21

1.3.4 Reduced individual complexity . . . 21

1.3.5 Greater flexibility . . . 21

1.3.6 Mission Planning . . . 21

1.4 Project specification . . . 22

1.4.1 Project aims . . . 22

1.4.2 'Micro mouse' maze . . . 22

2 Hardware Design **24**

2.1 Component selection . . . 24

2.1.1 AIRAT2 Chassis . . . 24

2.1.2 Ranging sensors . . . 24

2.1.3 Radio-frequency transceiver . . . 26

2.1.4 Microcontroller board . . . 26

2.1.5 Batteries . . . 27

2.1.6 Voltage regulators . . . 28

2.2 Stepper motor driver . . . 28

2.2.1 Hardware vs Software . . . 28

2.2.2 Driver chip selection . . . 29

2.3 Chassis extensions . . . 29

2.3.1 Solidworks chassis model . . . 29
2.3.2 Sensor housing . . . 30
2.3.3 PCB mounting plate . . . 30
2.4 Printed Circuit Board . . . 30
2.4.1 Requirements . . . 30
2.4.2 Schematic design . . . 32
2.4.3 Track layout design . . . 32

3 Software Design **33**
3.1 High-level overview . . . 33
3.1.1 Interrupt routines . . . 33
3.1.2 Mapping approach . . . 34
3.1.3 Communication . . . 35
3.2 Motor Controller . . . 35
3.2.1 Requirements for driving stepper motors . . . 35
3.2.2 Generating step schedules . . . 36
3.2.3 Stepping with interrupt routines . . . 36
3.3 Nordic nRF24L01 interface . . . 38
3.3.1 SPI interface . . . 38
3.3.2 Higher level functions . . . 38
3.3.3 Interrupt routines . . . 39
3.4 Distance measurement . . . 39
3.4.1 Noise . . . 39
3.4.2 Sample & Average timers . . . 39
3.4.3 Conversion to millimetres . . . 39
3.5 Movement between squares . . . 41
3.5.1 Single-square approach . . . 41
3.5.2 Measuring position errors . . . 42
3.5.3 Compensating for position errors . . . 43
3.6 Command handling . . . 45
3.6.1 Command stack . . . 45
3.6.2 Parsing commands . . . 45
3.7 Communication protocol . . . 46
3.7.1 Assignment of identification numbers . . . 46
3.7.2 Command, response . . . 46
3.8 Base station . . . 47
3.8.1 Communications relay board (RF to UART) . . . 47
3.8.2 Python . . . 47
3.8.3 Serial (COM) port access . . . 47
3.8.4 Storing the maze . . . 47
3.8.5 Graphical output . . . 48

4 Debugging **49**
4.1 Nordic nRF24L01 . . . 49
4.1.1 SPI interface . . . 49
4.1.2 RF packet not sent . . . 49
4.1.3 RFSend Interrupts . . . 50

5 Performance Evaluation **52**
5.1 Motor control . . . 52
5.2 Sensors . . . 54
5.2.1 Distance measurement . . . 54
5.2.2 Accuracy . . . 54
5.3 Mapping . . . 55
5.3.1 Navigating through the maze . . . 55
5.3.2 Communicating with the base station . . . 55
5.3.3 System in action . . . 56

6 Discussion **57**
6.1 Results . . . 57
6.2 Relevance . . . 57
6.3 Shortcomings & Improvements . . . 57

7 Conclusions **59**
7.1 Further work . . . 59

A Relevant Technologies **63**
A.1 Sensors . . . 63
A.1.1 Camera systems . . . 63
A.1.2 Laser scanners, . . . 64
A.1.3 Radar systems . . . 64
A.1.4 GPS and Inertial Measurement Units (IMUs) . . . 64
A.1.5 One-dimensional rangers . . . 64
A.2 Computer Systems . . . 65
A.2.1 Microcontrollers (MCUs) . . . 65
A.2.2 x86 Architecture . . . 66
A.2.3 Field Programmable Gate Arrays (FPGA) . . . 67
A.2.4 Interrupt Routines . . . 67
A.3 Communication hardware . . . 68
A.3.1 Infrared communication . . . 68
A.3.2 Radio frequency transceivers . . . 69
A.3.3 Operating frequencies . . . 69
A.4 Motors . . . 69
A.4.1 DC Motors . . . 70

A.4.2 Servo motors . . . 70
A.4.3 Stepper motors . . . 70
A.5 Inter-device Communication . . . 72
A.5.1 Serial Peripheral Interface (SPI) . . . 72
A.5.2 Universal Asynchronous Receiver/Transmitter (UART) . . . 72
A.6 Printed Circuit Board (PCB) . . . 73
A.6.1 Analogue Digital Separation . . . 73
A.6.2 Grounding . . . 74
A.6.3 Capacitors . . . 74

B Photographs and Electronic media **75**
B.1 Photographs . . . 75
B.2 Screenshots . . . 75
B.3 Supporting CD-ROM . . . 75
B.4 Website . . . 80

C Risk Assessment and COSHH **81**

D PCB Designs **87**
D.1 Schematic . . . 87
D.2 Track layout . . . 87
D.3 3D model . . . 87
D.4 Gerber files . . . 87

E Mechanical Drawings **94**
E.1 Sensor housing . . . 94
E.2 PCB mounting plate . . . 94

F Software Design **97**
F.1 Command protocol . . . 97
F.2 Sensor lookup table . . . 97

G Schematics & Datasheet extracts **101**
G.1 Evaluation board . . . 101
G.2 Batteries . . . 101
G.3 Sensors . . . 101

H Software Versions **109**
H.1 Luminary Micro . . . 109
H.2 Base station software (python) . . . 109
H.3 Design software . . . 109

Acknowledgements

Many thanks to Jim Swift for his constant encouragement and expertise throughout his supervision of the project. Thanks to Mr. S. Watson of the Mechanical Workshop and Mr. N. Clarey of the Electronics Workshop for manufacturing designs promptly and to an excellent standard. Finally, thanks for Mr. T. McFarlane and everyone in the Electronics Workshop for providing equipment and advice, often at unreasonably short notice.

Summary

This project set out to design a system of cooperative mobile robots which could work together to map unfamiliar terrain. This required a multi-disciplinary approach involving the areas of communication, precise mechanical control, electronic design, interpreting sensor output, software development and human interface design.

For the purpose of this project, a simplified mapping environment was used which involved a flat maze with a uniform, regular layout. Two robots were built although the system was designed to support many more. The wider applications of the system were considered throughout the project such as reconnaissance and search and rescue. To this end, the system was designed to be scalable and extensible, making use of modern hardware and portable software.

Having selected a suitable microprocessor board, ranging sensors and stepper motors, a PCB was designed to accommodate the power electronics and connect the microcontroller to the other components. The robot chassis was extended with additional aluminium parts which were designed to mount the sensors and PCBs. When the hardware design was complete, software was developed to interface with the motors, sensors and radiofrequency chip. Finally, a graphical software application was developed to serve as the coordinator and to illustrate the system to a human operator.

The system was developed to a point where two robots were able to reliably map an area, displaying the results in real time to the operator. This could be expanded beyond the uniform maze environment for mapping truly unfamiliar terrain. It could also act as a robust testing system for algorithm research or for teaching purposes.

Nomenclature

CAD	Computer Aided Design
CPU	Central Processing Unit
DARPA	Defense Advanced Research Projects Agency of the United States
FIFO	First-In, First-Out buffer
FPGA	Field Programmable Gate Array, a high-speed programmable logic device.
IMU	Inertial Measurement Unit
IRQ	Interrupt Request
ISR	Interrupt Service Routine
MCU	Microcontroller Unit
MOSFET	Metal Oxide Semiconductor Field-Effect Transistor
PCB	Printed Circuit Board
PRT	Personal Rapid Transit
RF	Radio frequency
SPI	Serial Peripheral Interface
UART	Universal Asynchronous Receiver/Transmitter
USB	Universal Serial Bus

List of Figures

1.1 International Micromouse Maze specification. 23

2.1 Placement of ranging sensors on the robot. 26
2.2 Sensor housing 3D CAD model. 30
2.3 SolidWorks final assembly. 31

3.1 Overview of communication between robots and base station computer. 35
3.2 Generating step schedules from two numerical inputs. 37
3.3 Sampling sequence of the four ADC channels. 40
3.4 Lookup table to convert ADC values to distances in millimetres. 41
3.5 Position errors as seen by the robot in a square. 42
3.6 Calculating angle measurement based on front and rear sensors. 43
3.7 Forward movement to compensate left errors. 44

5.1 Motor control tests. 53
5.2 Graph showing distance readings from sensor against actual distance. . 54

A.1 Stepper motor illustration. 71
A.2 SPI timing diagram. 73
A.3 UART data frames for 8-N-1 and 5-O-2 73

B.1 Photograph of the mobile robot hardware. 76
B.2 Photograph of the Luminary Micro LM3S1968 Evaluation board. 77
B.3 Photograph of two robots mapping a 4 x 9 maze after approximately 30 seconds of operation. 78
B.4 Screenshot of graphical base station immediately after activating two robots. 79
B.5 Screenshot of graphical base station after approximately 30 seconds of operation. 79

D.1 Top view of a 3D render of the PCB design. 90
D.2 Bottom view of a 3D render of the PCB design. 90

List of Tables

2.1 Minimum input/output requirements for peripherals on the microprocessor. 27
2.2 Comparison of three microcontrollers. 27
2.3 Current draw of components of the robot. 28
2.4 Voltage regulator specifications. 28
2.5 Requirements for printed circuit board. 32

3.1 Summary of the robot to base station command protocol. 46

5.1 Motor control test results - maximum errors. 53
5.2 Consistency of distance measurement. 55

Project Plan

Algorithm Research and Development

This will involve research into popular and theoretical maze solving algorithms ie flood fill and assessment of their suitability for multiple agent use. Where possible optimised variants will be developed. All algorithms to be investigated will be precisely and unambigiously specified to aid programming at a later stage. More general applications of multiple agent robotics will be considered at this stage.

Robot Hardware Construction

Research into different robot chassis models and exploring the possibility of designing and building one in-house. Research into the most suitable motors, sensors and RF modules. Design of sensor mounting for optimal performance given the maze and type of sensor.

Robot Electronic Construction

Research into microcontrollers and boards based on input/output requirements. Some PCB design will be necessary to connect the motor driver chips as well as connecting to batteries, sensors, RF module etc. The microprocessor could be mounted on the same PCB or connected from a separate board, for example an evaluation board. Sourcing and installation of MOSFETs to drive the motors.

Robot Software Development

This involves the setup of a PC base station for debugging over the radio frequency link. A communications protocol must be developed for sharing maze data over this RF link. Hardware interfaces must be written to drive stepper motors, read sensor values and send data over the RF link. A simple wall follower will be created to demonstrate control of motors and sensors. Position awareness code will be developed to

perform dead reckoning navigation, followed by a system to track the grid co-ordinate within a maze. Finally different solving algorithms will be introduced.

Algorithm Benchmarking

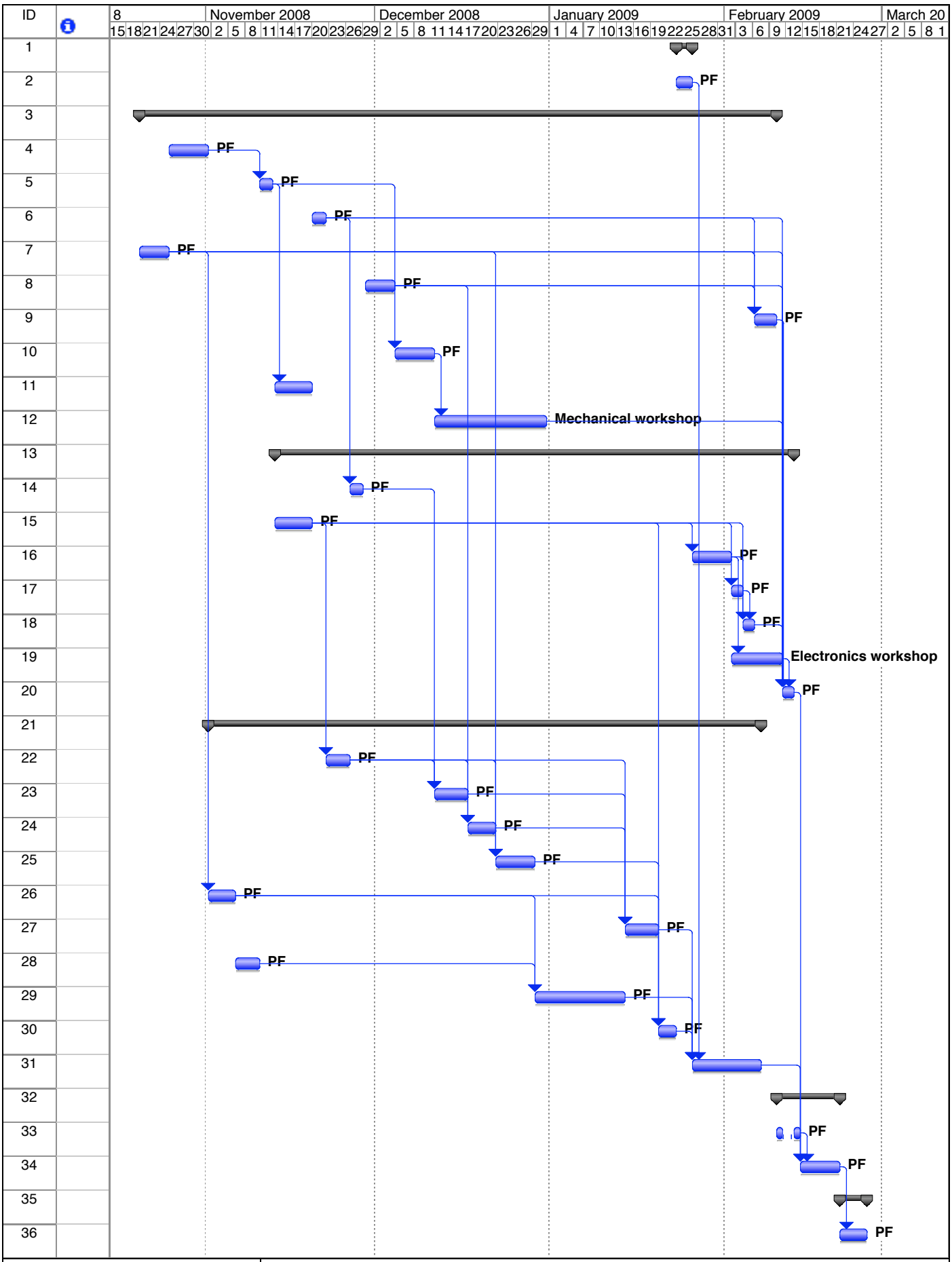
This will require the creation of a test suite defining which algorithms and mazes will be investigated and a performance metric to measure the relative success of a run. After this benchmarking test has been developed it will be carried out and data will be collected. If possible simulations will be performed and compared against physical runs.

Analysis and Further Study

Different components of the system will be tested such as motor control, sensors and using the RF link. Multiple robots will be used to test the performance of the system as a whole.

Where appropriate a comparison of algorithms will be carried out, identifying strengths and weaknesses in particular algorithms as well as performance bottlenecks. The effect of using multiple agents will be analysed for each algorithm.

ID	Task Name	Duration	Start	Finish	Predecessors	Resource Names
1	**Algorithm Research and Development**	**1 day**	**Fri Jan 23**	**Mon Jan 26**		
2	Research existing maze traversing algorithms	1 day	Fri Jan 23	Mon Jan 26		PF
3	**Robot Hardware Construction**	**82 days**	**Mon Oct 20**	**Tue Feb 10**		
4	Research available chassis models	5 days	Sat Oct 25	Sat Nov 1		PF
5	Select and purchase a chassis	2 days	Mon Nov 10	Wed Nov 12	4	PF
6	Research and select stepper/servo/DC motors	2 days	Wed Nov 19	Sat Nov 22		PF
7	Research suitable communications hardware	5 days	Mon Oct 20	Sat Oct 25		PF
8	Research and select suitable sensors	4 days	Sat Nov 29	Thu Dec 4		PF
9	Research and select suitable batteries	2 days	Fri Feb 6	Tue Feb 10	6,7,8	PF
10	Design chassis extensions	5 days	Thu Dec 4	Thu Dec 11	5	PF
11	Take delivery of chassis	5 days	Thu Nov 13	Wed Nov 19	5	
12	Chassis extensions manufactured by workshop	14 days	Thu Dec 11	Wed Dec 31	10	Mechanical workshop
13	**Robot Electronic Construction**	**67 days**	**Thu Nov 13**	**Fri Feb 13**		
14	Research and select motor controller chip	2 days	Wed Nov 26	Fri Nov 28	6	PF
15	Research and select microcontroller board	5 days	Thu Nov 13	Wed Nov 19		PF
16	Design power electronics PCB	5 days	Mon Jan 26	Mon Feb 2	15	PF
17	Allocate microcontroller output pins	2 days	Mon Feb 2	Wed Feb 4	15	PF
18	Produce crimped, socketed wires to connect microcontroller to PCB	2 days	Wed Feb 4	Fri Feb 6	17,16,15	PF
19	PCB manufactured by electronics workshop	7 days	Mon Feb 2	Wed Feb 11	16	Electronics workshop
20	Complete construction of two robots	2 days	Wed Feb 11	Fri Feb 13	18,19,12,6,7,8,9	PF
21	**Robot Software Development**	**71 days**	**Sat Nov 1**	**Sat Feb 7**		
22	Setup compiler for microcontroller	3 days	Sat Nov 22	Wed Nov 26	15	PF
23	Design and write motor controller	4 days	Thu Dec 11	Wed Dec 17	14,22	PF
24	Design and write sensor interface	3 days	Wed Dec 17	Mon Dec 22	8,22	PF
25	Design and write communications chip interface	5 days	Mon Dec 22	Mon Dec 29	7,22	PF
26	Design communication protocol	3 days	Sat Nov 1	Thu Nov 6	7	PF
27	Write wall-following software routine	4 days	Wed Jan 14	Tue Jan 20	23,24,22	PF
28	Select appropriate programming environment and setup on PC	3 days	Thu Nov 6	Mon Nov 10		PF
29	Design and write PC base station software	12 days	Mon Dec 29	Wed Jan 14	26,28	PF
30	Implement communications protocol in robot	3 days	Tue Jan 20	Fri Jan 23	26,15,25	PF
31	Implement algorithms in robot	10 days	Mon Jan 26	Sat Feb 7	29,30,2,27	
32	**Algorithm Benchmarking**	**9 days**	**Tue Feb 10**	**Sat Feb 21**		
33	Design test suite for testing algorithms	2 days	Tue Feb 10	Sat Feb 14		PF
34	Carry out testing of algorithms	5 days	Sat Feb 14	Sat Feb 21	33,31,20	PF
35	**Analysis and Further Study**	**3 days**	**Sat Feb 21**	**Thu Feb 26**		
36	Compare algorithms	3 days	Sat Feb 21	Thu Feb 26	34	PF



Project: Cooperating Mobile Robots
Date: October 2008

Task		Project Summary	
Split		External Tasks	
Progress		External Milestone	
Milestone		Deadline	
Summary			

Chapter 1

Introduction & Literature Review

1.1 Self-Navigation

As with many technical developments, improvements in both safety and efficiency have been the driving force behind research into automation. Minimising human exposure to danger has been a priority in just about every area: military, industry, manufacturing, automobiles and research. Reducing cost is another priority and in developed countries, human labour is a significant overhead. Industrial automation has demonstrated that certain tasks can be performed at a lower cost and more consistently by machines. So in general, our strive for automation is about replacing humans with machines in the hope that they can perform more safely, cheaply and consistently.

One critical ability of humans is understanding and interacting with their immediate surroundings. As humans, we often take for granted fundamental abilities like recognising objects and walking among obstacles. These have proved to be immensely complex tasks to reproduce in a machine and we find ourselves at various stages of success in doing so.

Self-navigation is an important capability of an autonomous machine and it depends upon a detailed understanding of the immediate surroundings, or *environment awareness*. Arguably, this is the most difficult part; capturing raw sensor data and processing it into a machine understandable form. Planning a route and moving around within this environment, once understood, is relatively simple. In some projects self-navigation is not considered the goal but has occured as a logical progression from a system with robust environment awareness.

The capabilities of self-navigating machines can be enhanced by communication with other, similar machines. Tasks can be distributed and machines can be utilised differently depending on their suitability for a job. This kind of interaction requires the use of communication protocols and may be co-ordinated by a central control system or handled in a distributed manner.

1.2 Applications of Self-Navigation

1.2.1 Hazardous Environments

Many environments are dangerous or inaccessible to humans and these have been key to development. Areas of high radiation, such as the inside of a nuclear reactor, can only be accessed using machines. If a disaster like Chernobyl were to happen again, much of the investigation would likely be carried out by semi-autonomous machines.

Space exploration has pushed the boundaries of autonomy as live remote control is impossible at such great distances. At the time of writing there were two Mars Exploration Rovers autonomously exploring Mars, regularly sending images and scientific data back to Earth. The robots, Spirit and Opportunity, landed in 2004 on a mission to discover the history of water on Mars. They were expected to survive for three months but have currently been operating successfully for five years. These robots have produced far more useful data than expected and so funding is likely to remain for future exploration.

Military zones have produced a requirement for autonomy to reduce human casualties. In 2004 and 2005 Defense Advanced Research Projects Agency (DARPA) ran its 'Grand Challenge' in which teams competed to build an autonomous vehicle to navigate across a desert. The aim for these vehicles is secondary military use; autonomous delivery of supplies, rescue of wounded soldiers, bomb detection and disposal etc.

In 2005, five teams completed the 132 mile course, a huge improvement from the unanimous Failed to Qualify awarded the previous year[1]. The winners, Stanford Racing Team, used four laser range finders, a radar system, a stereo camera pair, and a monocular vision system[2] and focused their labour on image processing software rather than overly specialised hardware. 2007 saw the successful completion of the 'Urban Grand Challenge', a significantly more complex task requiring interaction with other vehicles and obedience of traffic rules[3].

1.2.2 Search & Rescue

Search and Rescue is often carried out in hazardous environments so the argument for autonomy is clear. In some situations the conditions, such as weather or low light, could actually advantage a purpose-built machine. An article in the Engineer[4], 'Subsea Saviour', proposes an autonomous underwater vehicle which swims beneath the surface, immune to poor weather conditions. Instead of the traditional search for a human head *above* the surface of the water, the vehicle travels under the surface measuring the 'depth of water' above it, revealing human bodies as sudden changes. Canadian company, International Submarine Engineering, have developed an air-deployed, GPS-enabled dinghy which can be steered towards survivors. Currently, the technology is still remotely operated but with suitable self-navigation capability it could scan

a large area of water around a rescue boat very quickly.

Reconnaissance & Surveillance

In August 2007 the UK Ministry of Defence launched their own Grand Challenge in which teams had to build fully-autonomous solutions to seek out various hazards in an urban environment[5]. Currently, Unmanned Air Vehicles (UAVs) are used to perform this task but operating them requires total supervision from specialist teams. The MOD wishes to reduce the need for human intervention, allowing them to deploy the system and wait for it to produce a report.

The winning team, Stellar, used three autonomous vehicles; high and medium altitude UAVs and a ground vehicle. The high altitude vehicle was launched first, circling the area of interest and relaying aerial photographs to a controlling base station. The base station analysed the incoming images, identified threats and tasked the other two vehicles to further investigate areas of interest. Their live video data and other sensor data was relayed to the control station, allowing it to update its mission data and issue further instructions to the three robots.

Among others, the United States Army has enlisted the the iRobot PackBot for reconnaissance in urban environments. The company has been awarded a five-year contract valued up to $200 million[6]. While not self-navigating, the PackBot is equipped with multiple cameras and 'various other sensors', but the US Military is likely to push autonomy as part of its long term plan; read more on page 19.

1.2.3 Health care

Aethon, Pittsburgh, have developed a self-navigating robot, or Automated Robotic Delivery System which they call TUG[7]. It is essentially an autonomous, mobile cupboard which can deliver meals, pharmaceuticals, IVs, waste etc from one area of a hospital to another. TUG primarily uses infrared ranging sensors to locate its position and it contains a pre-programmed map of the environment. Aethon claims that TUG allows staff to spend less time walking hallways and riding elevators and more with the patient, and that over 1000 hospitals in the US are using one.

1.2.4 Industry

Carnegie Mellon University's Robotics Institute are working with Caterpillar, a major sponsor of the winning team in the DARPA Urban Grand Challenge, to develop autonomous industrial vehicles[8]. The large-haul vehicles are part of a bigger autonomous mining haulage system and are planned to be introduced by 2010.

The US National Institute of Standards & Technology's (NIST) has developed self-navigation algorithms as part of its Industrial Autonomous Vehicle (IAV) project. The project's goal is shown below[9].

> Reduce costs and improve efficiency in industrial material handling by providing to the industrial AGV industry performance tests to support the use of non-contact safety sensors and appropriate control systems architectures and standards to enable the use of advanced navigation techniques based on such non-contact sensors.

At present, the technology is deemed too expensive to justify incorporating into existing forklifts and other vehicles but this may change as the technology matures.

1.2.5 Domestic

Technology has moved a step closer to realising the dream of the robotic butler in every home. The Roomba vacuum cleaner from iRobot is an autonomous, self-navigating domestic cleaning robot which aims to remove vacuuming from the list of household chores. The Roomba 560 senses its environment using infrared ranging sensors but it is *near-sighted*, having no environment awareness further than its immediate surroundings.

Public Transport Systems

Personal Rapid Transit (PRT) was a vision of the past, involving self-navigating personal vehicles operating co-operatively within a dedicated infrastructure. While previous attempts have been commercially unsuccessful, Advanced Transport Systems Ltd (ATS) have secured a deal with BAA to install the world's first PRT system called ULTra at Heathrow airport. The system currently comprises a fleet of eighteen electric, self-navigating vehicles which shuttle passengers and their luggage between the car parks and the Terminal 5 building[10].

Users of the ULTra system select their destination at one of multiple stations which are distributed like bus stops. The selection is passed to a central control system which allocates a vehicle and sends it instructions describing the path and timings of the journey. The control system manages unoccupied vehicles, allowing it to send them to areas with the greatest demand. Once loaded with instructions, the vehicles operate autonomously, navigating using a combination of embedded causeway magnets and local ultrasonic and laser rangers. They operate independently of the central control system until they arrive at their next destination.

Corby Borough Council is currently considering introducing the ATS ULTra system to connect a number of new greenfield housing developments to the town centre[11]. As part of an expensive reinvention of the town, a PRT system could encourage more use of public transport and provider better green credentials for the council. A study by consultants Colin Buchanan and Partners concluded that the costs would be similar to that of light rail but that a PRT system could attract 70% more passengers.

1.2.6 Consumer automobiles

Consumer automobiles have leapt forward in their environment awareness capabilities with many modern cars being equipped with sophisticated sensor systems. Taking Volvo's Collision Avoidance Package as an example, the car is equipped with a grill-mounted automobile radar and a road facing camera[12].The radar constantly measures the distance to the vehicle in front and can intervene if a rear-end collision is imminent. The camera uses complex object recognition software to monitor road markings and audibly warns the driver if the car drifts out of its lane at an unexpected time.

Many obstacles lie in the path of *fully* autonomous consumer automobiles, in particular the legality and potential liability of autonomously controlled machines which are extremely dangerous when handled incorrectly. Opponents of the idea claim that people enjoy driving too much and would never trust a computer to do it for them. Others suggest that roads would be safer without fallible human drivers, and that roads will one day become so congested that no-one would enjoy driving on them. Opinion is divided but some, including Dr. Peter Schulmeyer, Freescale Semiconductor Inc (Austin, Texas), argue that full autonomy is a logical progression[13] from today's capabilities:

> "We will get there, step by step, not because someone is developing a self-navigating car, but because the enabling technologies are starting to show up for other reasons, One day, someone will realize that we are almost there, and they'll take that last step."

Unmanned Military

In the United States, much recent research into self-navigation has been the result of legislation introduced in 2001. This was the motivation behind the DARPA Grand Challenge, whose description is shown below.

> Created in response to a Congressional and DoD mandate, DARPA Grand Challenge is a field test intended to accelerate research and development in autonomous ground vehicles that will help save American lives on the battlefield.

The text is referring to a Congressional Mandate from 2001 which contained the extract shown below[14].

> SEC. 220. UNMANNED ADVANCED CAPABILITY COMBAT AIRCRAFT AND GROUND COMBAT VEHICLES.
>
> (a) GOAL.—It shall be a goal of the Armed Forces to achieve the fielding of unmanned, remotely controlled technology such that—
>
> (1) by 2010, one-third of the aircraft in the operational deep strike force aircraft fleet are unmanned; and

(2) by 2015, one-third of the operational ground combat vehicles are unmanned.

As an aside, it is worth pointing out that the mandate itself states the objective of 'unmanned' technology and DARPA specifies 'autonomous'. It may seem academic but the difference is vital; autonomous implies 'without any human decision' which could be extended to the decision to kill. 'Unmanned' allows for *human-controlled,* remotely operated machines. Unfortunately various reporters and academics have jumped on the idea of the US Military designing robots to "make their own decisions about lethality"[15], a quote which is taken out of context from its original publication in the book titled "Autonomomous Vehicles in Support of Naval Operations"[16].

Weapons aside, environment awareness and self-navigation look set to boom with the help of various military funding. The report[17] titled "Unmanned Systems Roadmap 2007-2032" is an interesting read, describing the current state of unmanned robotics and the objectives and technologies required for the next two decades. Aspects of this technology are likely to filter down to the consumer market in one form or another.

1.3 Multiple, Communicating Robots

Some projects have gained functionality from employing a team of connected robots. Teams Stellar and Swarm used multiple robots to explore an urban environment from different points-of-view. The Personal Rapid Transit system required communication between the vehicles and the base station to coordinate them and improve speed and efficiency. In general, connecting robots together can expand their functionality and make them more useful than a single robot. Some reasons for this are outlined below.

1.3.1 Redundancy

Using a team of co-operating machines, or 'agents', provides redundancy which is beneficial for a number of reasons. As one of the key areas of use is hazardous environments, the safety and security of the agents are not guaranteed. A system which comprises disposable, replaceable components is more able to remain useful in the event of a failure. This applies to all manner of failure; manufacturing defects, software faults, power supply problems etc.

1.3.2 Improves 'near-sightedness'

DARPA have acknowledged the problem of 'near-sightedness' of robotic systems which arises from the limited range of sensor equipment coupled with the lack of learning from experience. This leads to robots repeatedly struggling with the same obstacle

or travelling down a cul-de-sac. In response they set up a project[18] called Learning Applied To Ground Robots (LAGR) in which they aimed to develop an improved awareness of the environment through machine learning.

Using a team of robots which can communicate effectively can sidestep this problem by providing multiple points of view. Provided each robots has an idea of its position relative to others, it is possible to build up a bigger instantaneous picture than any individual robot could. This was clearly demonstrated by team Stellar in the MOD Grand Challenge, in which the ground vehicle was issued directions based on imagery from the high-altitude UAV.

1.3.3 Speed

Multiple agents should be able to carry out a task more quickly than a single agent by, for example, covering more terrain in a given period. However, there are likely to be a limits as the complexity of the control system increases and an area becomes too crowded for useful work to be possible. Communication capacity could also limit the number of agents used so scalability should be designed in from the outset.

1.3.4 Reduced individual complexity

Using a number of highly specialised, heterogeneous robots would reduce the individual complexity. For example different agents could carry different sensor packages and hence the individual cost of manufacture (and replacement) could be reduced. This would be balanced with the reduction in cost gained from standardisation of parts.

1.3.5 Greater flexibility

Using heterogeneous robots also proves useful in mission planning. Specific mission or task requirements could be accommodated by deploying different numbers of each type of robot. For example, a mission to be carried out entirely in the dark may have little use for robots with conventional cameras but could use a higher proportion of infra-red imaging. As well as the ability to tailor the system to a mission, this approach could be scalable, accommodating assignments varying in size and complexity.

1.3.6 Mission Planning

Using an interconnected system of robots could allow the use of algorithms for optimising the work performed by each agent. Single agents may struggle to have a wide enough awareness of the environment. A key part of this type of system is the communication method employed and the reporting of position and sensor output to a central control system.

1.4 Project specification

1.4.1 Project aims

The project set out to research some of the technologies involved in a co-operating mobile robot system. The aims of the project are shown below.

- Select a suitable microprocessor system capable of processing sensor data, short range communications, controlling motors and storing a local representation of the environment.
- Research and select suitable motors, sensors and a communications module.
- Design hardware to interface the microprocessor with motors, sensors and communications module.
- Produce microprocessor code to control motors, read sensors and communicate through a communication module.
- Create an interface board to allow a PC to communicate with the robots using the same communications module.
- Develop a command protocol for reporting position and environment data between the robot and desktop computer.
- Create a graphical display on the PC to show a real-time view of multiple robots in their environment.

1.4.2 'Micro mouse' maze

In order to focus on the communication and diagnostic capabilities of the system, the project was undertaken using a well defined, regular maze to simulate an environment. This reduced the task to two-dimensional exploration which allowed the use of simple ranging sensors.

The maze used was based on that used in the International Micromouse Competition. This comprises a regular array of 180mm squares divided by walls as shown in Figure 1.1.

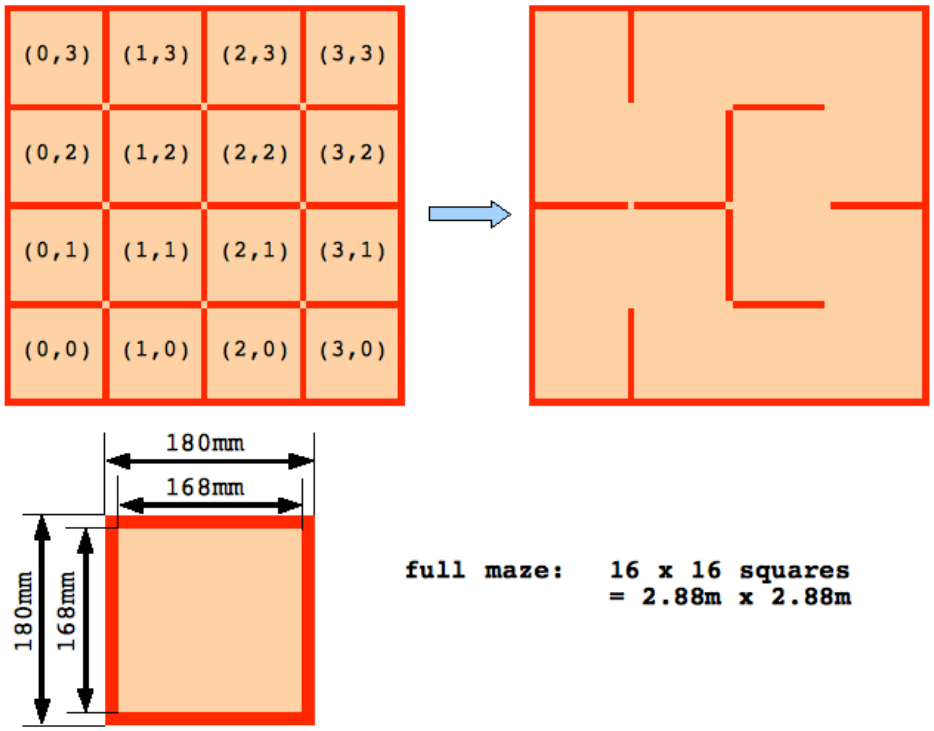


Figure 1.1: International Micromouse Maze specification.

Chapter 2

Hardware Design

2.1 Component selection

2.1.1 AIRAT2 Chassis

A chassis containing suitable stepper motors, wheels and a mounting platform was found available for sale. It was decided that it would be cheaper and faster to buy in this chassis rather than buying stepper motors and designing a chassis to be made in the workshop.

The AIRAT2 chassis was designed for use in the micromouse competition and so was ideally dimensioned. The wheels were milled from aluminium with a rubber grip to reduce slipping, and measured 52mm in diameter. The body of the chassis was made from folded aluminium and featured two casters to allow smooth contact with the ground at the front and rear of the robot.

The chassis featured Sanyo Denki 103H564-0440 stepper motors. The stepper motors were required to be able to accelerate a 2kg robot at a rate of 2.5m/s^2. This was used to estimate the torque or moment required by the stepper motor. Assuming a wheel diameter of 52mm, the force required at the wheel radius is roughly given by $F = ma = 2 \times 2.5\text{N}$. This gives a moment of 0.13Nm and assuming two motors, 0.065Nm per motor. The quoted torque of the stepper motor was 0.147Nm which met the acceleration requirement.

2.1.2 Ranging sensors

2.1.2.1 Requirements

The dimensions of one side of a square in the maze was 180mm so a ranging sensor had to be able to operate effectively in the range of around 40mm to 180mm. The accuracy had to be no less than 10mm and continuous readings were preferable.

2.1.2.2 Ultrasonic ranger

An affordable range of ultrasonic sensors from Devantech were analysed. Model SRF05 had a minimum distance of 34mm, slightly greater than the quoted 30mm, based on the quoted residual vibration duration of 100us.

Multiple ultrasonic sensors must be used alternately as they interfere with each other and the datasheet quoted a period of 65ms between use of adjacent sensors so a group of four could only be read every 0.26 seconds.

The transducers operate at 40 kHz which has a period of 25us, giving a wavelength of around 8.6mm in air. The sensors use eight pulses which equates to a maximum possbile error of $7 \times 8.6 = 60.2$mm which is considerably worse than the quoted accuracy. The datasheets available were of poor quality as they were made primarily for hobby electronics so it is difficult to know how well they would perform in reality.

2.1.2.3 Infrared intensity

These sensors were not available as packages so design and calibration would have been more extensive than an off-the-shelf sensor. The disadvantage of infrared intensity was external factors such as reflectivity of the surface of the maze, angle of the surface and ambient light conditions all affect the readings. Micromouse teams have reported variations in readings due to ambient light so calibration would have to be performed on every use.

2.1.2.4 Infrared triangulation

Infrared triangulation sensors were deemed to be the most suitable for this application due to their resistance to ambient light effects, continual measurement and lack of interference with each other. Many teams involved in the micromouse project report success in using these sensors. There is a selection of infrared rangers from Sharp and the closest match to the requirements was the GP2D120.

2.1.2.5 Configuration

To navigate through the maze the robot needed to be able to detect walls in three directions: forward, left and right, based on the knowledge that there is no wall in the direction from which the robot entered.

To detect errors in orientation the robot needed to be able to measure its angle relative to a wall. Adding a fourth sensor at the rear left allowed it to compare the front and rear left measurements and calculate an angle.

The placement of the sensors is shown in Figure 2.1.

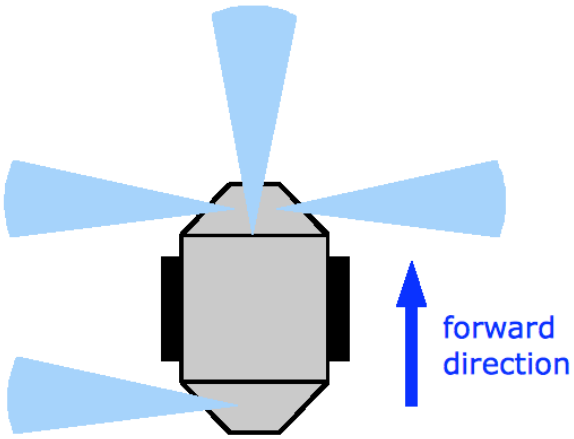


Figure 2.1: Placement of ranging sensors on the robot.

2.1.3 Radio-frequency transceiver

The Nordic nRF24L01 2.4GHz radio transceiver was selected for use on the robot. The chip was available on a breakout board at a reasonable price with an RP-SMA antenna. There was experience of the chip within the School of Engineering and an excellent tutorial was found online which clarified many of the shortcomings of the datasheet.

The 2.4GHz chip featured auto-acknowledge and auto-retransmit as well as builtin two-byte CRC checking and five-byte addressing. The range was quoted to be around 100m line-of-sight which was sufficient for the micromouse maze environment which had a maximum distance of $\sqrt{2 \times (16 \times 0.18m)^2} = 4.073$m.

The data rate of the chip was advertised as a maximum of 2Mbps. Using a 32-byte payload with a five-byte address, two-byte CRC and one-byte preamble gave an 80% utilisation, or 1.6Mbps data rate.

The nRF24L01 was could operate at any of 128 frequency channels. In accordance with the Frequency Allocation Table 2008 from OFCOM, only the first 84 channels were allowed to be used.

2.1.4 Microcontroller board

The minimum specification for a microprocessor was the input/output requirements for communicating with peripherals, as outlined in Table 2.1. The following additional features were deemed attractive.

- Large volatile memory (>4KBytes)
- Large non-volatile (program) memory (>128KBytes)
- Hardware SPI interface for interfacing with nRF24L01
- LCD or OLED Screen for outputting numerical debugging information.

Device	GPIO pins	Analogue I/Ps
Stepper motor (x2)	8	
Sharp IR ranger (x4)		4
Nordic RF transceiver	6	
Total:	14	4

Table 2.1: Minimum input/output requirements for peripherals on the microprocessor.

Manufacturer	Model	Architecture	Non-volatile memory	Volatile memory
Luminary Micro	LM3S1968	ARM	256KB	64KB
Texas Instruments	MSP430F1611	MSP430	48KB	10KB
AVR	ATMega128	AVR	128KB	4KB

Table 2.2: Comparison of three microcontrollers.

- High clock speed for fast computation.

Table 2.2 shows a comparison of three microcontrollers with which there was experience in the School of Engineering.

The Luminary Micro LM3S1968 contained an impressive list of hardware features and the large volatile memory was attractive for storing environment data.

The LM3S1968 was available as an evaluation board which featured 128 x 96 pixel OLED screen and several user-configurable buttons. The screen was considered very valuable for outputting debug information and using evaluation boards would allow the project to progress without having to wait for a PCB to be manufactured.

The LM3S1968 runs at 50MHz which was very fast for a microcontroller at the time of writing. This allowed the use of computationally expensive tasks where necessary.

2.1.5 Batteries

2.1.5.1 Power consumption

The requirements for a battery were decided as below.

1. Provide a sufficient voltage to supply 3V and 3.3V voltage regulators.
2. Be able to provide a current sufficient for the highest possible current draw from the motors, sensors, microprocessor and radio transceiver.
3. Provide several hours of continuous use.

It was decided that a battery pack comprising four NiMH cells made by GP Batteries would be used. The cells, GP410LAH, had a maximum output current of 12.3A which was around 50% greater than the maximum current draw of the robot. The capacity was 4100mAh which would provide $2\frac{1}{4}$ hours of use at the estimated current

Component	Maximum current (mA)	Typical average current (mA)
2x Stepper motors	2 x 4000 = 8000	1500
5x IR Sensors (GP2D120)	5 x 50 = 250	5x33=165
1x Evaluation board	350	100
1x RF transceiver (nRF24L01)	12.3	13
	8612mA	**~1800mA**

Table 2.3: Current draw of components of the robot.

Component	Voltage regulator	Current required (mA)	Current output (mA)	Drop-out at max current (V)
Motor	UCC283−3	2000	3000	0.45
Evaluation board	LF33ABV	350	500	0.4
Sensors	LP2950	250	300	0.38

Table 2.4: Voltage regulator specifications.

of 1800mA. Recharging could be achieved quickly by a charge-controlled cycle or in approximately 14 hours at a constant current of 410mA.

The form factor was also suitable for mounting on a robot, the batteries measuring 18.3mm diameter by 65.5mm length. These were soldered and taped together in fours to make a 4.8V battery pack.

2.1.6 Voltage regulators

Voltage regulators were required to supply the evaluation board and motors at 3.3V and the sensors at 3V. It was decided that the motors would use their own voltage regulators to isolate them from the evaluation board.

A key requirement was low drop-out, the minimum different between supply voltage and output voltage, allowing the voltage regulators to continue to operate as far as possible through the batteries' discharge cycle. Again using the estimated current consumption of 1800mA, the battery datasheet showed that after two and a quarter hours of use each battery would have dropped to 0.9V or 3.8V for the whole pack. The dropout therefore had to be below 0.5V for the 3.3V regulators and 0.8V for the 3.0V regulator.

2.2 Stepper motor driver

2.2.1 Hardware vs Software

There exist a number of stepper motor driver chips such as A3977 from Allegro MicroSystems, Inc which generate waveforms based on a direction and step input as well

as configuration inputs to select the type of waveform. The A3977 is advertised as being the 'ideal fit for applications where a complex microprocessor is unavailable or over-burdened'.

The alternative approach is to generate waveforms in software, controlling digital outputs and using a simple transistor chip to switch enable the current in the winding. This requires more execution time on a microprocessor but offers a higher level of control.

It was a requirement that the motors could be synchronised but using chips with their own timing could have made this difficult. The microprocessor was not over-burdened and synchronisation was a priority so it was decided a quad transistor chip would be used for each motor. Additionally, more modern stepper motor driver chips were only available in surface mount packages which are unsuitable for prototyping.

2.2.2 Driver chip selection

According to the stepper motor datasheet each of the four coils had an internal resistance of 3.15Ω and the recommended operating voltage was 3.15V to give a current of 1A/phase. Therefore each transistor had be able to switch at least 1A of output current.

A chip called ULN2065B was found to be suitabled for this purpose, having an output current of up to 1.5A (absolute maximum 1.75A) and containing four NPN darlington switches. It was available in a DIP-16 package which was ideal for prototyping.

When the supply voltage is suddenly removed from an inductive load like a stepper motor winding, the rapid current change in the inductor induces a voltage spike across the inductor. A *flyback* diode is placed across the inductive load to equalise the voltage and prevent the voltage spike causing damage to the transistor. The ULN2065B chip includes two internal flyback diodes for this purpose which increases integration and requires fewer external components.

2.3 Chassis extensions

2.3.1 Solidworks chassis model

The AIRAT2 chassis had mounting holes but it was necessary to design some additional parts to mount the sensors, microcontroller board and power electronics PCB. To simplify this design process, the off-the-shelf chassis was reverse engineered and modelled using SolidWorks, a 3D CAD package.

As additional parts were conceived they could quickly be incorporated into the chassis design to ensure check for interference and allow measurement of important dimensions such as from the sensors to the wheel hubs.

After the parts were manufactured in the workshop, the robot was partially constructed and the parts fitted as predicted by the model. Additionally, various improve-

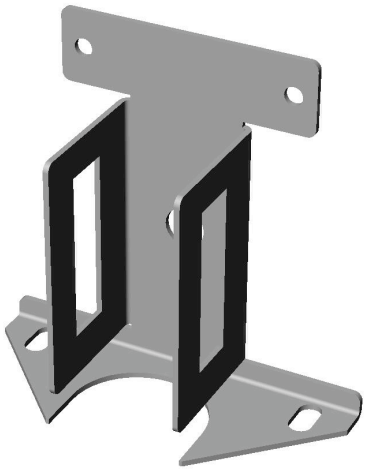


Figure 2.2: Sensor housing 3D CAD model.

ments were then made to the AIRAT2 chassis design itself and are included for future use. The mechanical drawings for all parts can be found in Appendix E.

2.3.2 Sensor housing

It was necessary to produce housing to mount the Sharp GP2D120 sensors on the AIRAT2 chassis. The minimum operating range of the sensors was 40mm so it was necessary to mount them as close to the centre of the robot as possible. This required a number of design revisions but the final design is shown in Figure 2.2.

2.3.3 PCB mounting plate

In order to mount the two PCBs on the chassis, a mounting plate was designed with holes matching those on the AIRAT2 and those on the evaluation board. Stacking pillars were used to mount the plate above the chassis then stack the two PCBs above this, with space for batteries in between. The SolidWorks Assembly illustrates this in Figure 2.3.

2.4 Printed Circuit Board

2.4.1 Requirements

Having decided opted to use a microprocessor evaluation board, a secondary board was required to house motor transistor chips, voltage regulators and connectors for peripherals. This required the design of a printed circuit board (PCB) to accommodate

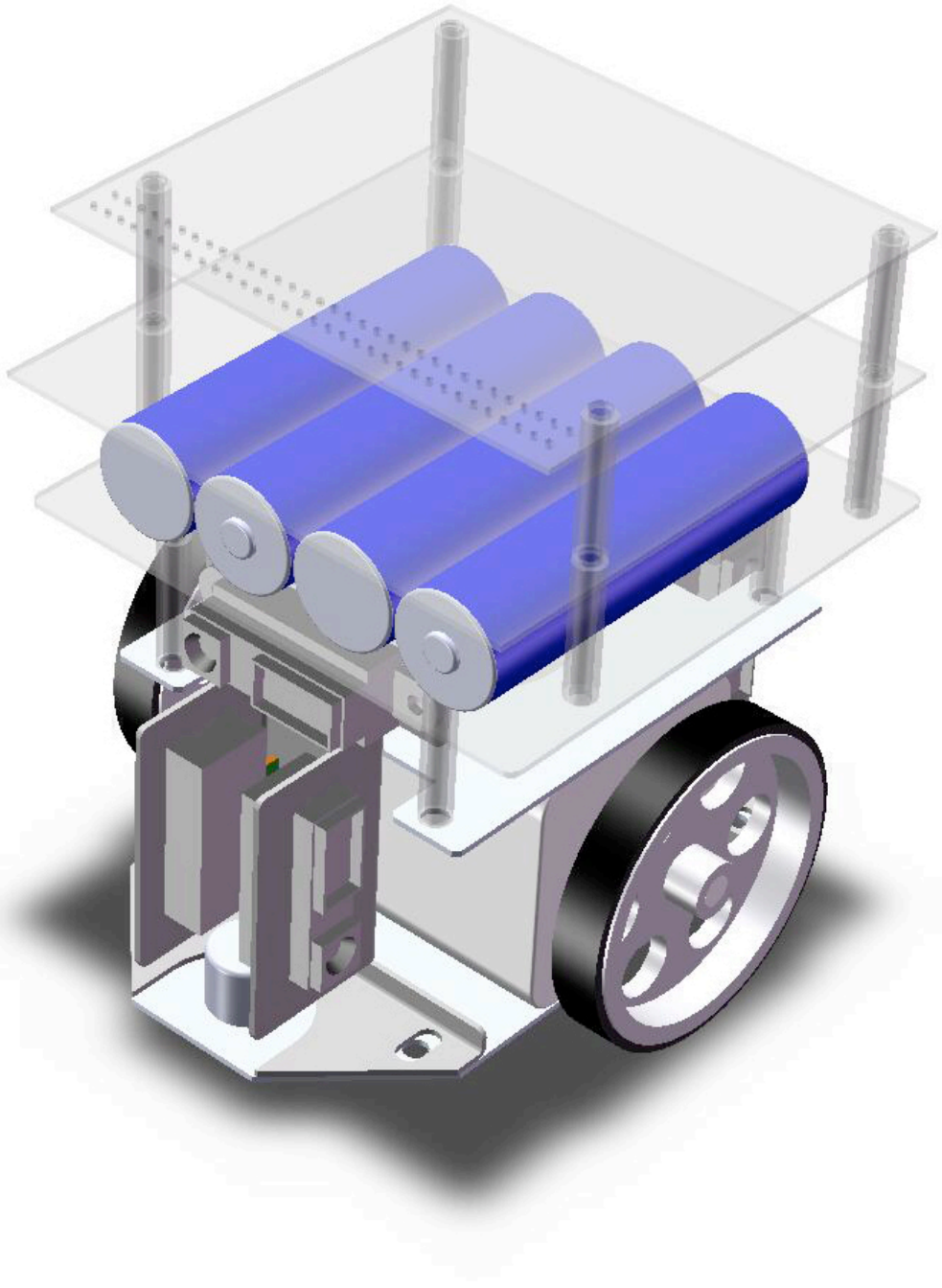


Figure 2.3: SolidWorks final assembly.

Components	Connectors
2x Quad MOSFET chips	1x Battery
2x 3.3V voltage regulator for motor	5x Sensor (3-wire)
1x 3V voltage regulator for sensors	RF breakout board (8 pin header)
1x 3.3V voltage regulator for EVB	Battery level (1 pin)
Power switch	
Power indicator LED and resistor	
Battery capacitors	
Voltage regulator capacitors	
Potential divider circuit for battery level ADC	
Decoupling capacitors	

Table 2.5: Requirements for printed circuit board.

the components shown in Table 2.5. The PCB was designed to have mounting holes in the same location as those on the LM3S1968 evaluation board.

2.4.2 Schematic design

The schematic design was created using CadSoft Eagle Layout Editor v5.3.0. Several component footprints had to be created using the component editor as they were not available in the standard libraries. The final revision of the schematic can be seen in Appendix D.1 on page 87.

2.4.3 Track layout design

Once the schematic diagram was complete, Eagle Layout Editor was used to create a PCB design. The design was of relatively low complexity so a two-sided board was used. The component footprints were placed according to the design principles discussed in Appendix A.6: the headers for the ADCs were placed as far as possible from the transistor chips and a ground fill was used on both sides of the board.

Version 1 of the PCB was produced and a number of design alterations were considered. The placement of most of the headers was awkward so these were moved to the edges of the board. The isolation around the ground fill was 0.016″ which caused short circuits in a number of places so this was doubled for version 2.

Additionally, it was decided that a power switch and indicator LED would be useful additions, as well as a potential divider with an ADC header for measuring the voltage level of the battery. The schematic design was altered to include these features and version 2 of the PCB was manufactured. The track layout can be seen in Appendix D.2.

Chapter 3

Software Design

3.1 High-level overview

3.1.1 Interrupt routines

The robot had a minimal main program loop to ensure valuable instruction cycles weren't wasted by polling hardware and waiting for the response. As an ideal, most functionality was provided by the following six interrupt service routines.

1. Stepping one or both motors by outputting the next value from the step sequence.
2. Initiating an ADC sampling sequence, triggered by a 50Hz periodic timer.
3. Reading the ADC value at the end of the sample sequence, triggered by the ADC hardware itself.
4. Averaging the last ten ADC values and converting voltage level to millimetres, triggered by a 5Hz periodic timer.
5. Stepping one or both motors to their next position, triggered by two dynamically configured one-shot timers.
6. Interpreting a message from the RF chip and loading new packets onto a command stack for later use.

Some functions were slower to execute so were more suited to execution in the main control loop. The flow of the main loop is shown below.

- Identify walls of current square.
- Calculate the direction of the next move or read it from an instruction from the base station.
- When permitted, calculate the step schedule (see Section 3.2.2) required to move into the centre of the next square and initiate that movement.

- Parse and act upon new commands in the RF command stack, sending replies where necessary.

3.1.2 Mapping approach

It was decided that the robots would initially have a minimal understanding of their environment, allowing the base station to co-ordinate movement and mapping instructions.

The robots navigated the maze on a single square basis. The procedure for movement is described below.

1. Enter square.
2. Identify walls.
3. Enter *freeze* mode.
4. If a POLL command is received:
   (a) Send a REPORT with the current location, direction, walls detected and status information (such as freeze/moving flags).
5. If an UNFREEZE OR MOVE_TO command is received:
   (a) Establish direction of next square to move to.
   (b) Calculate position error and compensation (see Section 3.5.2).
   (c) Generate step schedule to move to centre of next square.
   (d) Start step schedule.
   (e) Load new position into temporary 'destination' variables.

The robots were programmed to enter a square and enter a *freeze* mode whereby they were unable to move until instructed. Initially the base station was configured to regularly *poll* each robot in turn to which the robot would respond with a *report* command containing information about the current square, whether frozen, or moving, etc.

Upon receiving a REPORT command from a frozen robot the base station could optionally send an UNFREEZE or MOVE_TO command. In the former case, the robot would make its own simple decision about where to move next, in this case aiming to simply follow the left wall. Otherwise, the robot would attempt to move to the co-ordinate specified by the base station.

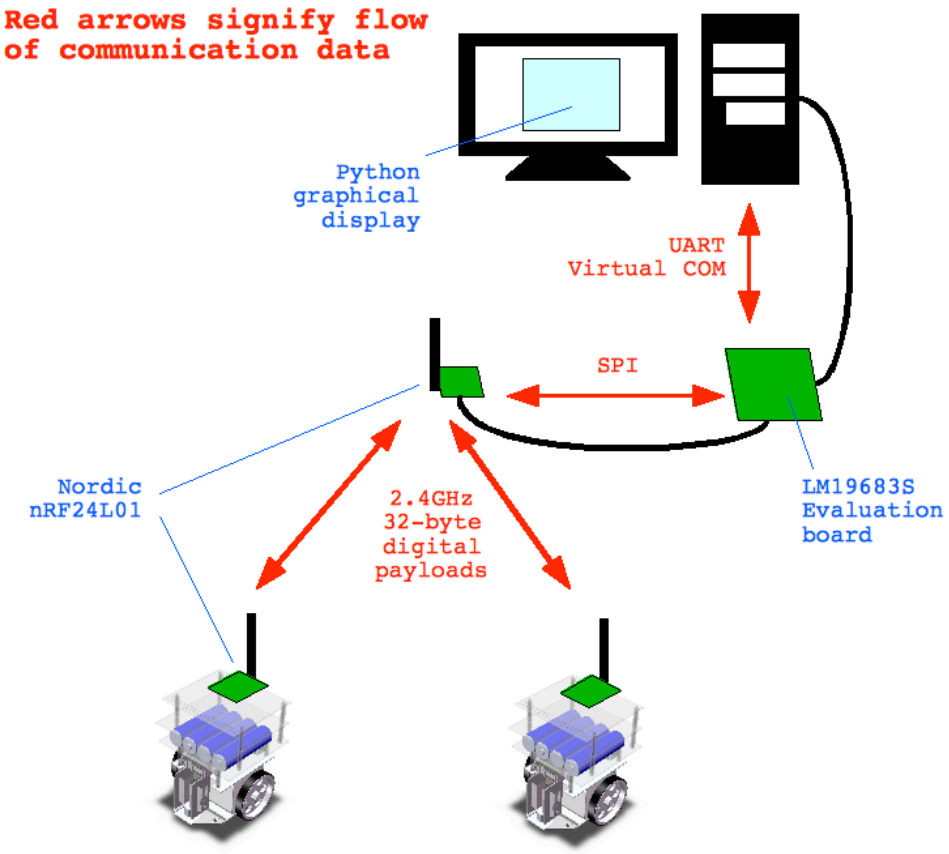


Figure 3.1: Overview of communication between robots and base station computer.

3.1.3 Communication

It was decided that there would be no direct communication between any two robots. A protocol was designed which allowed communication only through the base station. This greatly reduced the complexity of the robot's code and simplified debugging as all decisions were made in the base station.

If all robots had kept a full local copy of the mapping data, synchronisation could have been a problem, especially if packet loss occurred. There would also be a limit to the size and complexity of an area being mapped as the robots ran out of internal memory. Storing the map on an external computer meant there was more computational power available for route finding algorithms and other tasks.

Figure 3.1 illustrates the communication path between the robots and the computer.

3.2 Motor Controller

3.2.1 Requirements for driving stepper motors

Each stepper motor was controlled by four digital outputs which had to be asserted in a specific sequence and at varying delays depending on rotational speed required. In half-stepping mode, the motors operate in the region of 100 to 500 steps per sec-

ond which translates to delays of 2 to 10ms. The requirements for the stepper motor software interface were as listed below.

- Independently step both motors at periods of 2 to 10ms.
- Operate within 100 to 500 steps per second.
- Maintain a low rate of change of acceleration (jerk).
- Allow the processor to carry out other tasks while moving.

3.2.2 Generating step schedules

The rotational speed of stepper motors is determined by the frequency of the excitation sequence being applied to the windings. Hardware timers were used to increment through the waveform after a given delay. For any movement, such as moving forward 180mm, a sequence of these delays had to be calculated.

A function called generateStepSchedule(left_steps, right_steps) was created which generated two *step schedules*, or arrays containing delays based on the number of steps required from each motor. The function was programmed with a builtin acceleration profile and a deceleration profile. These were 100-long floating point arrays containing delays in seconds which had been precalculated using a spreadsheet. The profiles were based on maximum and minimum speeds of 500steps/s and 100steps/s respectively.

For a large number of steps (over 200), a step schedule would comprise a 100-step acceleration part, a constant velocity part and a 100-step deceleration part. A small number of steps would generate a step schedule with a constant velocity of 100 steps/s, the minimum speed of the motor.

Figure 3.2 illustrates the process of generating step schedules.

The acceleration and deceleration profiles used sinusoidal changes in delays to minimise jerk and reduce subsequent vibration. The angled corners of the linear velocity profile represents an instantaneous change of acceleration which signifies infinite jerk.

The left and right step schedules were eventually converted from floating-point delays in seconds to integer delays in clock cycles. These were appended to a global array which was limited to a total of 10,000 steps. After generating multiple step schedules, movement was initiated by running the function runStepSchedule(). This configured and activated two hardware timer modules.

3.2.3 Stepping with interrupt routines

Two decrementing 24-bit timers[1] were configured for one-shot use, one for each motor. When the step schedule was run, each timer was initially loaded with the first delay

[1] Technially, 16-bit timers were used with an 8-bit prescaler set to its maximum value. For this purpose the 16-bit timer was effectively equivalent to a 24-bit timer.

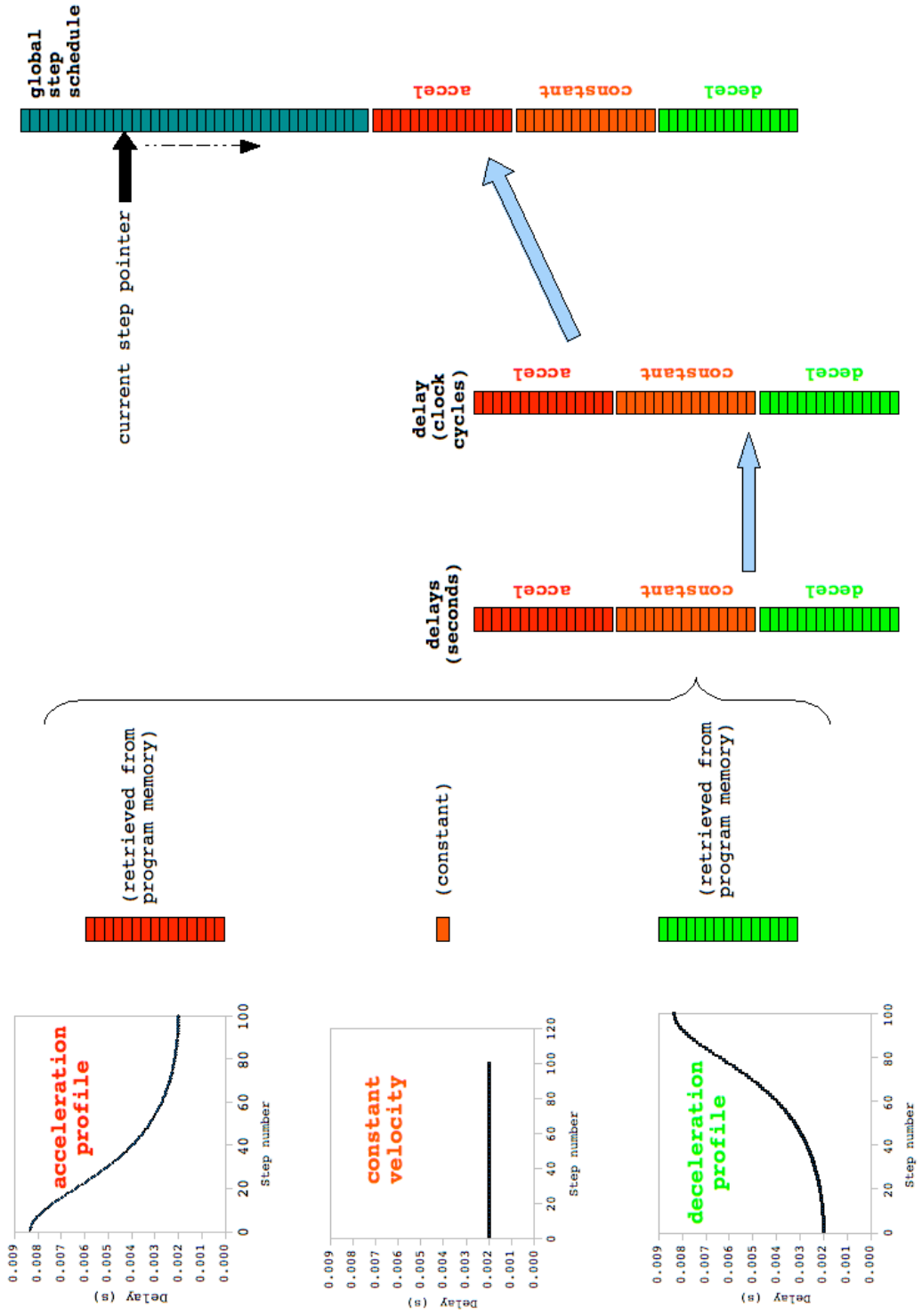


Figure 3.2: Generating step schedules from two numerical inputs.

from the corresponding left or right step schedule. The timers were configured to trigger an interrupt on reaching zero, signifying that the motor should be stepped. The interrupt routine loads the relevant timer with the next delay in the step schedule, re-enables it, then increments through the stepper motor output waveform and asserts the new value to the motor digital outputs.

Additional global variables were used to track the position of each motor through its step schedule. When completed, the corresponding timer is disabled, all motor windings are de-energised and the step schedule is reset.

3.3 Nordic nRF24L01 interface

3.3.1 SPI interface

The Nordic nRF24L01 was accessed by use of an SPI bus. The LM3S1968 had hardware support for three types synchronous serial interface: SPI, MICROWIRE and Texas Instruments. It was necessary to configure the hardware for SPI as well as specifying the clock polarity and phase control bits, as documented in the Nordic datasheet.

A low level function called SPI_tx() was built which took a single byte as an argument, sent it over the SPI bus and returned the response byte. This would become the foundation of several other functions.

3.3.2 Higher level functions

The Nordic transceiver was configured by loading a number of configuration registers, most of which were a single byte wide. This was achieved by sending a 3-bit WRITE_REGISTER command, OR'd with the 5-bit address of the register, followed by a byte representing the new register value. Reading a register was performed in a similar way but using a dummy byte[2] after the READ_REGISTER command.

A number of functions were created to abstract the process of reading and writing registers. On top of these were higher level routines which manipulated single bits in a register, for example setting transmit/receive mode, power on/off mode etc.

Sending and receiving packets was achieved by means of READ_PAYLOAD and WRITE_PAYLOAD commands, followed by 32 dummy or data bytes depending on the operation. Two functions were written to handle these operations.

The interface was designed with reusability as a primary goal. The code needed to be able to run on the robot as well as the communications relay board described in Section 3.8.1 on page 47, so applications-specific features needed to be located elsewhere.

[2]The dummy byte is required because for every byte sent, a single byte is received. In order for the slave device to respond to the command byte, it needs a second byte during which to do so.

3.3.3 Interrupt routines

Several conditions caused the radio transceiver to set its IRQ (Interrupt Request) pin low, indicating that it needed attention. The IRQ pin was connected to a digital input on the microcontroller and an interrupt routine was configured to trigger on a falling edge. Interrupt conditions were identified by a bit being set in the *status* register.

When one or more data packets were received by the radio transceiver an interrupt was triggered. The *receive payload* function was repeatedly used until the FIFO_STATUS register indicated that the receive buffer was empty. Each packet was then passed to an *external* function in the command handler, allowing application-specific processing of the packet.

The robot's handling of packets is explained further in Section 3.6 on page 45.

3.4 Distance measurement

3.4.1 Noise

It was discovered that the voltage regulator providing power to the sensors had introduced high frequency noise. The cause of this noise remains unknown but implementing a simple software averaging routine was adequate to produce very useable measurements. The final implementation made use of both builtin 64x hardware oversampling and a 10 point average.

The sampling sequence is illustrated in Figure 3.3.

3.4.2 Sample & Average timers

A 16-bit periodic timer was configured to trigger an interrupt at a rate of 50Hz. This interrupt routine initiated the ADC sampling sequence. The ADC hardware itself was configured to take 64 samples of five channels, each connected to a sensor and call an interrupt at the end of the sequence. At this point, the 10-bit ADC values were read from the ADC and entered into an array.

A second 16-bit timer was configured to run at 5Hz and trigger an interrupt routine. This routine took the mean of the 10 values in the array and used a function to convert the value to millimetres.

3.4.3 Conversion to millimetres

The output voltage from the Sharp infrared ranger varied with distance in a non-linear function as documented in the component datasheet. Tabulating a series of points on the graph and calculating a power (or log-log) regression produced the function $f(x)$ below, relating 10-bit ADC value to a distance in millimetres.

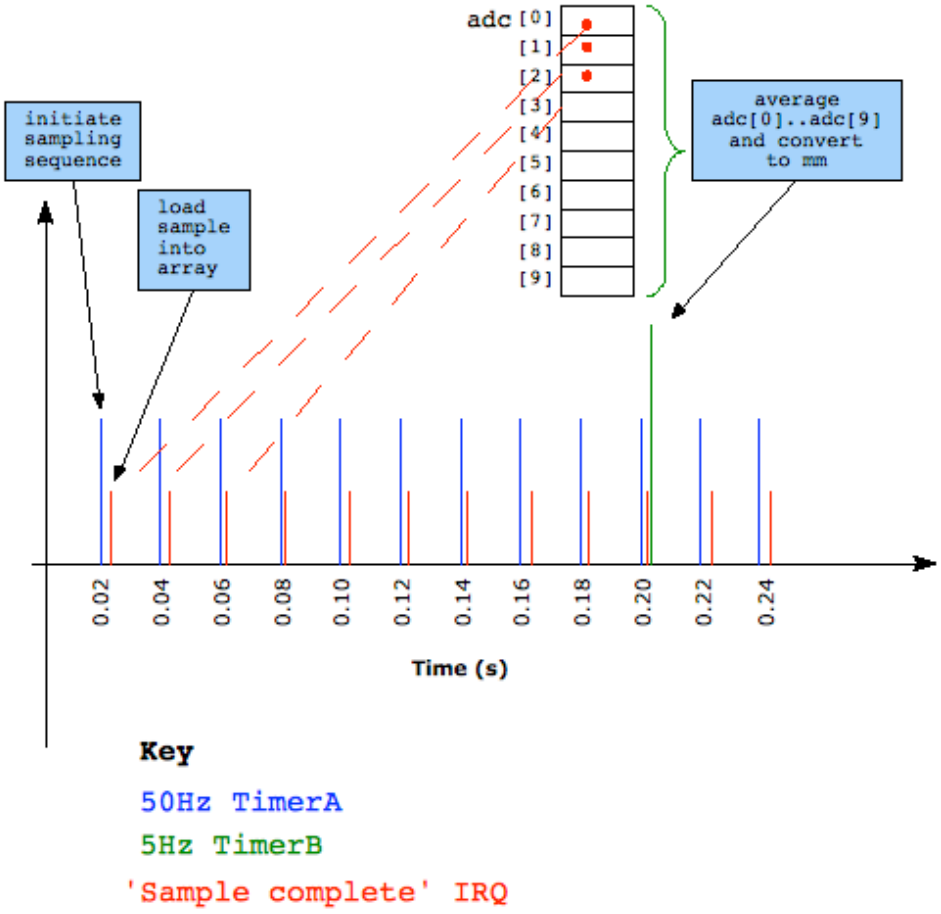


Figure 3.3: Sampling sequence of the four ADC channels.

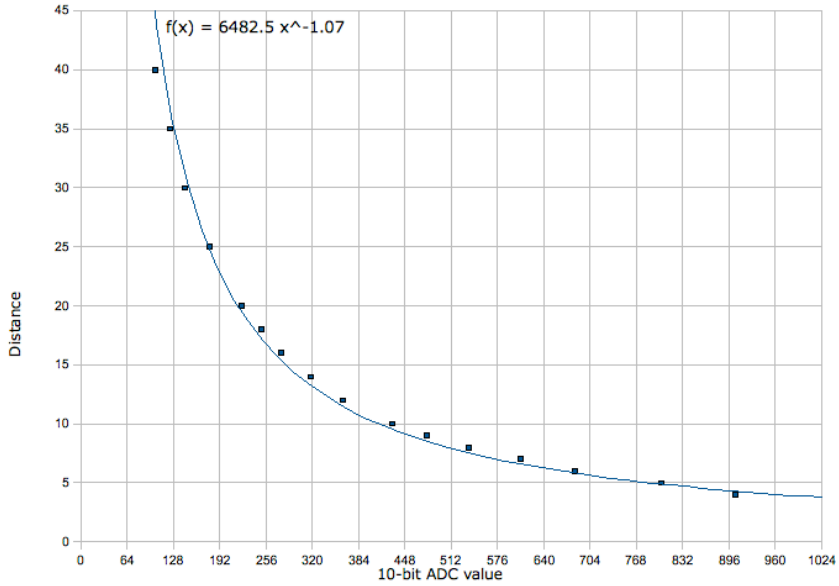


Figure 3.4: Lookup table to convert ADC values to distances in millimetres.

$$f(x) = 6482.5x^{-1.07}$$

This was used to produce a lookup table which mapped ADC value to an integer number of millimetres. The 10-bit precision of the ADC was greater than the resolution of the sensor so only the eight most significant bits of the ADC reading were used to index the lookup table. This also allowed the distance in millimetres to be stored as an eight-bit *unsigned char* (0-255).

The final lookup table was included in the program and was indexed with the most significant eight bits of the ADC value. Figure 3.4 shows $f(x)$ plotted against the values taken from the datasheet.

3.5 Movement between squares

3.5.1 Single-square approach

The robot was programmed with four routines for moving between squares: move forward, move right, move left, move backward. Functions were written to derive a direction from an absolute square co-ordinates allowing the base station to send a MOVE_TO command in i, j and leave the robot to work out how to achieve the movement.

Distance measurements were taken on entering a square and movement instructions were based only on the *instantanous snapshot*. For example, if a robot were to enter a dead-end square, the walls would be immediately identified then a motor step-

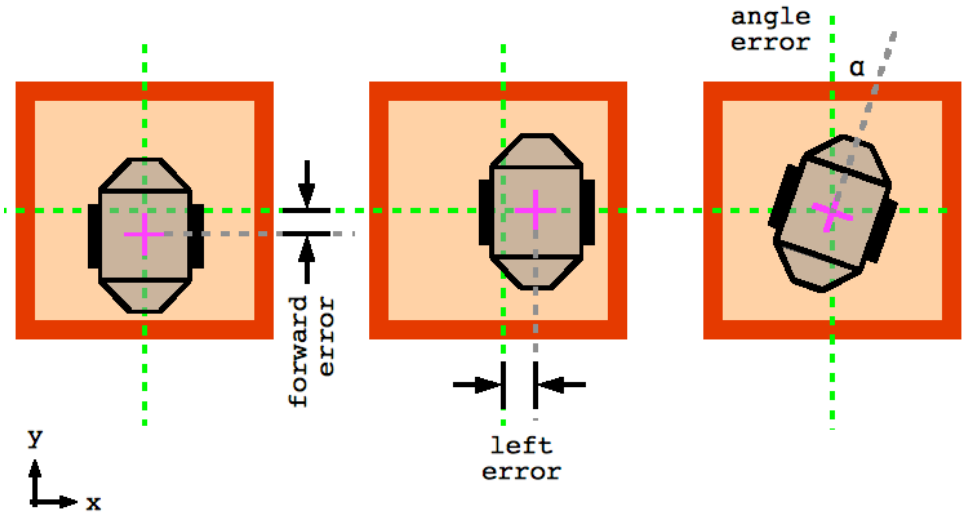


Figure 3.5: Position errors as seen by the robot in a square.

schedule would be generated to reverse the robot back out of the square and perform a rotation.

3.5.2 Measuring position errors

Although stepper motors are able to move through a controlled angle, factors described in Appendix A.4 can reduce the accuracy and lead to a build up of errors. The possible errors x, y and α were expressed relative to the robot as movement was determined in this way. The three types of error are illustrated in Figure 3.5.

Forward error in which the robot was off-centre in the forward/reverse direction. On turning 90° this would lead to the robot being too close to the wall of the next square.

Left right error in which the robot was too close to the left or right wall. This error could cause the robot to crash into a wall when turning on the spot.

Angle error in which the robot was not parallel to the adjacent walls. This would cause the robot's left/right error to build up which could result in a direct crash into the wall.

The forward error could only be directly measured in the presence of a forward wall. Left/right error could be measured from either a left or right wall. Due to the left side having two sensors, the left wall was preferred as a distance measurement could be taken as the average of the two sensors. Angle errors could only be calculated by comparing the front-left and rear-left sensors so a left wall was required to measure angle error.

Angle error measurement was calculated by the formula below, as illustrated in Figure 3.6.

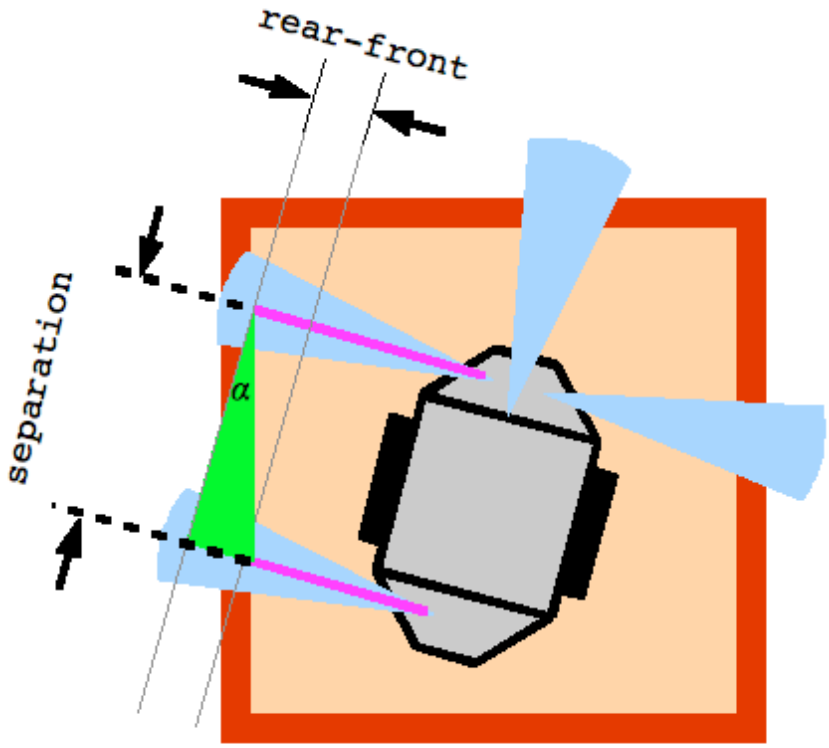


Figure 3.6: Calculating angle measurement based on front and rear sensors.

$$\alpha = tan^{-1}\left(\frac{rear - front}{separation}\right)$$

3.5.3 Compensating for position errors

Having measured the three types of error, the step schedules needed to include some compensation to aim to arrive in the centre of the next square.

Angle compensation was applied before any further movement. In the case of right/left turns it was combined with the 90° rotation of the turn sequence.

Left/right compensation was simple for right and left turns as the left/right error would become forward error after turning through 90°. It could therefore be subtracted from the forward movement into the next square.

It was discovered that runs of multiple forward movements could quickly lead to the left error building up so a left-compensate routine was developed for forward movements. Rather than simply moving forward, the routine incorporated a small rotation of angle β, a move forward then a small rotation in the opposite direction. This angle was calculated to aim the robot directly at the *centre* of the next square and was calculated by the formula below. This is illustrated in Figure 3.7.

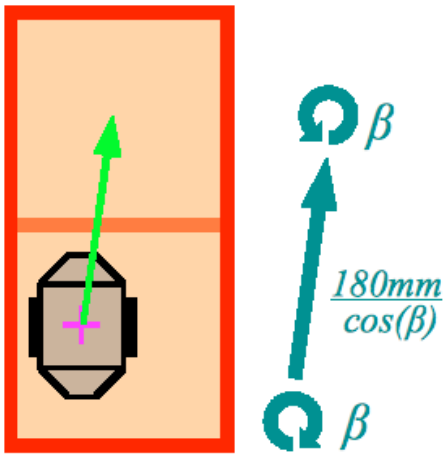


Figure 3.7: Forward movement to compensate left errors.

$$\beta = tan^{-1}\left(\frac{leftError}{180}\right)$$

Because the forward movement was no longer parallel to the walls, the distance had to be increased to reflect the length of the hypotenuse of the triangle. The distance was given by $\frac{180}{cos\beta}$.

With compensation included the step schedules were generated from the following high level instructions.

- **Move forward**
  - rotate clockwise $(\beta - \alpha)^\circ$
  - move forward $\frac{180}{cos\beta}$mm
  - rotate anticlockwise β°
- **Move right**
  - rotate clockwise $(90 - \alpha)^\circ$
  - move forward (180mm - leftError)
- **Move left**
  - rotate anticlockwise $(90 - \alpha)^\circ$
  - move forward (180mm - leftError)
- **Move backward**
  - rotate clockwise $-\alpha^\circ$
  - move backward (180mm - forwardError)
  - rotate 180° clockwise

3.6 Command handling

Most commands required a response command to be sent. To give the receiving robot time to switch back to receive mode it was necessary to pause between receiving and sending a reply. This time delay meant parsing commands was not well suited for implementation in an interrupt routine.

Initially payloads passed from the RF interrupt routine to the command handler were placed in a buffer. These were then parsed as part of the main program loop. Because multiple packets could arrive before any command parsing occurred, the buffer could easily be overwritten, losing the first packet. This was overcome by making a *command stack*.

3.6.1 Command stack

When a new packet or payload is received by the Nordic RF chip it uses an active-low IRQ pin to signify that it requires attention. An interrupt was configured to trigger on this signal becoming low level.

As the interrupt routine received the packets from the RF chip it had to store them somewhere for later inspection and parsing by the main program loop. It was essential that multiple commands could be stored as the command parsing would not happen instantly and it was important not to overwrite unprocessed commands with new ones.

A command stack was created which contained 10 'slots' of one payload width each, and 10 status bytes to signify whether the command in each slot had been processed or not. When new commands were received from the interrupt service routine they were placed into the last available slot and their status byte was changed to signify that the command was new, or unprocessed.

3.6.2 Parsing commands

Parsing commands and sending replies was a potentially time consuming process so was implemented as part of the program's main control loop. New commands were read from the command stack, sent to a command parsing function then marked as processed, freeing up the slot for a new command. Due to the way it was implemented, the length of the stack could be easily expanded in the future.

The first byte of any command represented the name of the command itself. A *switch* statement was used to execute instructions specific to the command received. As an example, the *case* for the PING command simply paused for 200us then sent a PING_REPLY command in response.

As described in Section 3.7, the second and third bytes of most commands (with the exception of initialisation commands) specified the ID number of the robot to which the

Base station to Robot commands	Robot to Base station commands
ASSIGN_ID	REQUEST_ID
PING	PING_REPLY
GRANT_DENY_ENTRY	REPORT
FREEZE	REQUEST_ENTRY
SET_WALL_STATUS	FREEZE_ACK
UNFREEZE	DEBUG
MOVE_TO	CMD_UNKNOWN
STEP_MOTORS	ASSIGN_ID_ACK

Table 3.1: Summary of the robot to base station command protocol.

command is addressed. Rejection of commands which were not destined for a particular robot was carried out at the command parsing stage. This means the command stack could potentially need to be very large to accommodate multiple packets destined for other robots. If this were to become problematic, packets could be dropped at any earlier stage before entering the command stack.

3.7 Communication protocol

3.7.1 Assignment of identification numbers

On reset, the robot was programmed to send a command REQUEST_ID which was received by the base station. The command included a 'random' *token* which was generated from the least significant bit of the eight ADC channels. In response, the base station generated a random identification number and sent it with the token in an ASSIGN_ID command. Upon successful receipt of the latter command, the robot set its identification number and responded with a ASSIGN_ID_ACK command.

Once an identification number had been assigned, the robot treated it as an address and ignored further commands whose ID didn't match. This allowed the base station to talk individually with each robot.

A special broadcast address byte was defined as 0xFF. This was designed for future use in which the base station may want to issue one-way commands to all robots such as for synchronisation.

3.7.2 Command, response

After the ID assignment sequence, all subsequent commands would be of the format *command, response* or *'speak only when spoken to'*. This limits the potential for collisions in the system as the packets being sent should be fairly predictable from the base station. Table 3.1 shows a summary of the commands exchanged between the base station and robot. See Appendix F.1 on page 97 for a description of all the bytes of each command.

3.8 Base station

3.8.1 Communications relay board (RF to UART)

The software on the base station computer needed to be able to send and receive packets using the nRF24L01. For this purpose, an RF to UART interface board was built using another identical LM3S1968 evaluation board connected to a transceiver chip.

Using the Virtual COM Port driver provided by Luminary Micro, the computer's COM6 port was configured to communicate with the microprocessor's UART. The RF interface code from the robot was used to communicate with the Nordic transceiver.

The communications relay board implemented its own *command handler* module to handle data from the RF chip. In contract to the command handler of the robot, the base station simply passed 32-byte packets between the UART and the RF transceiver.

For debugging purposes the OLED screen was used to display the number of packets sent, packets received and some counters.

3.8.2 Python

It was decided that the open source, interpreted language Python would be used for the base station software on the computer. Python is distributed with a vast library of modules which provide useful high-level functionality such as dynamic lists, dictionaries and various graphical functions. For this reason python was attractive for the rapid development of a graphical demonstration program. Python is also multiplatform which allows users more freedom to choose what system to run programs on.

3.8.3 Serial (COM) port access

For testing purposes a terminal program called RealTerm was used to send bytes to and from the LM3S1968's UART. It was able to display bytes in ASCII or hexadecimal form and could be easily used to construct packets for testing and displaying the replies received.

Once basic communication had been established with a robot via the communications relay board, a python module called pyserial was used to send commands directly from python and read back responses. A python 'command handler' class was built which provided higher level functions to construct outgoing and interpret incoming command packets.

3.8.4 Storing the maze

A 'maze handler' class was built which held a 2D array representing the status of each wall in the maze. It provided functions like *setWall(i, j, stat)* and *queryWall(i, j, stat)* to

interface with other code.

3.8.5 Graphical output

A python package called pygame was used to display the walls of the maze as well as the position of each robot within it. The package describes itself as a highly-portable set of python modules designed for writing games.

This made it highly suitable for displaying simple graphics required to visualise the progress of robots in a maze. Pygame provided simple transformations like rotation and drawing basic geometic objects or importing images.

Chapter 4

Debugging

4.1 Nordic nRF24L01

4.1.1 SPI interface

The microcontroller SPI interface was configured and the outputs SSIClk, SSIFs, SSIRx and SSITx were connected to the RF transceiver pins CLK, CSN, MISO and MOSI respectively.

In a two-byte command sent to the RF transceiver (eg READ_REGISTER, dummy), two bytes should be received: a status byte followed by the result of the command. A problem was encountered in which *all* bytes received from the transceiver were the status byte.

It was discovered that the SSIFss pin controlled by the microcontroller hardware behaved in a way which was incompatible with the transceiver. As illustrated in Figure A.2 on page 73, the active-low SSIFss (slave select) signal went to logic low at the end of each byte. This caused the RF transceiver to reset its command handling routine so every subsequent byte was interpreted as a new command. This meant the chip kept sending status bytes as it prepared to respond to a command, but was never able to.

The solution to this problem was to ensure the CSN pin stayed logic high for the duration of an SPI transaction. This was achieved by reconfiguring the SSIFss pin as a software controlled digital output and asserting the correct logic level before and after each SPI transaction.

4.1.2 RF packet not sent

The procedure for sending a packet the nRF24L01 was implemented as described below.

1. Switch to transmit mode by writing 0 to bit PRIM_RX of register CONFIG.
2. Load a packet into the transmit buffer.

3. Bring CE pin high for 10us to instruct the transceiver to send the payload.
4. Switch back to receive mode.
5. Prepare for IRQ_DS signifying data sent.

Rather than receiving the 'data sent' interrupt, after three packets an interrupt was being generated signifying that the transmit FIFO was full. To aid debugging, another board was set up and the value of the 'carrier detect' register was observed. This board was able to see data being sent but never received packets, suggesting that they were somehow truncated.

It was discovered that the reason for the data failing to send was that there was insufficient time between steps 3 and 4, resulting in the chip being switched back to receive mode during the transmission of the packet. This was solved by introducing the following sequence between steps 3 and 4.

- Wait for 100us.
- Check the TX_FIFO_EMPTY bit in the FIFO_STATUS register.
- If not empty:
  - Go to 3.

To protect from infinite execution in the event of a problem with the chip, the loop was limited to a finite number of iterations. If this limit was reached, a warning message was displayed on the OLED screen and execution was halted. This meant problems with the RF chip were detected quickly which proved useful when debugging the problem described in Section 4.1.3.

4.1.3 RFSend Interrupts

In a controlled environment the base station and the robot were able to communicate and interpret each other's commands with no apparent problem. However, when the amount of RF traffic increased, something was causing the communication to irrecoverably break down between the two points.

Initial observations pointed to a bug in the motor step scheduling code as the robot tended to crash as it was about to start a maneuver. Occasionally, however, the robot continued to run but the computer failed to send commands, which raised suspicions about the communications relay board. It was difficult to know when the relay board had crashed because it only updated its OLED screen as a result of sent and received packets. A software counter was implemented as a simple way of testing if the board was still running and this revealed that it was crashing in the same way as the robot. This narrowed the problem down as the only code common to both applications was the RF transceiver and SPI interface.

Eventually it was discovered that the cause of the problem was that packets being received would trigger an interrupt, and the interrupt routine would request the status register from the RF transceiver. This request was sent using the SPI interface. If a packet was received *during* the sending of an outgoing packet (which took a relatively long time), *both* the RFSend() routine *and* the interrupt routine would simultaneously attempt to use the SPI interface, often resulting in crashing the RF transceiver. This in turn would cause the chip to report unexpected values when requesting registers, some of which the main program relied on in order to continue.

The problem was fixed simply by disabling the interrupt when starting any SPI transaction and re-enabling it afterwards. The interrupt trigger was changed from FALLING_EDGE to LOW_LEVEL to prevent the possibility of missing an event during the period that the interrupt was disabled.

Chapter 5

Performance Evaluation

5.1 Motor control

A number of tests were devised to measure the accuracy of the movement of the robot. For each test the robot was placed inside a square drawn on a flat surface and the test sequence of movements aimed to return the robot to the exact same start position. Performance was evaluated by measuring the error in x, y and angle. The three tests used are described below and illustrated in Figure 5.1. Each test was performed five times and the maximum errors recorded are shown in Table 5.1.

Forward, reverse involved ten repetitions of 180mm forward movement followed by 180mm in reverse.

Turn, turn back involved ten repetitions of rotating 180 right followed by 180° left.

Right-angled corner involved moving forward then rotating right and moving forward, then performing a 180° turn on the spot, moving forward, turning left and moving back toward the start position, then turning on the spot to finish.

It was observed that the robot often 'jolted' into action at the very start of a moving sequence, often causing an angular error. This effect was caused by the stepper motor aligning to the nearest tooth on the rotor when powered up. In the right-angled corner test this angular error translated to errors in x and y.

Another source of error was the angular resolution of the motors. Using a half-step waveform each step signified a rotation of 0.9° meaning there were 400 steps per revolution, translating to a distance of $\frac{52\pi}{400} = 0.41$mm per step. Additionally, floating point to integer conversion of steps meant that the number of steps was always rounded *down* so remainder steps would always be truncated. This was remedied by always adding 0.5 to a floating point number before performing the integer conversion.

It was clear from the tests that the robot would have been unable to navigate through the maze had it not used position adjustments as it went. As an example, an angular error of 4° would become a translation error of $180sin(4^\circ)$ =12.6mm *per square* which could quickly cause the robot to crash into a wall.

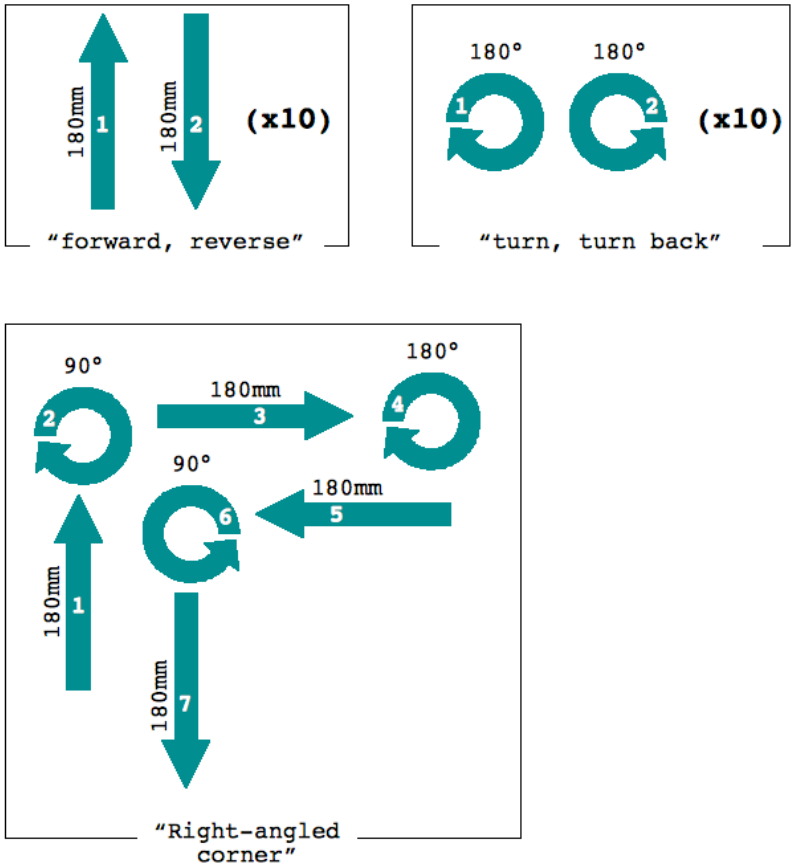


Figure 5.1: Motor control tests.

Test	Max error in x (mm)	Max error in y (mm)	Max error in angle (°)
Forward, reverse (10x)	8	9	4
Rotate right, rotate left (10x)	10	8	4
Right-angled corner	13	12	1

Table 5.1: Motor control test results - maximum errors.

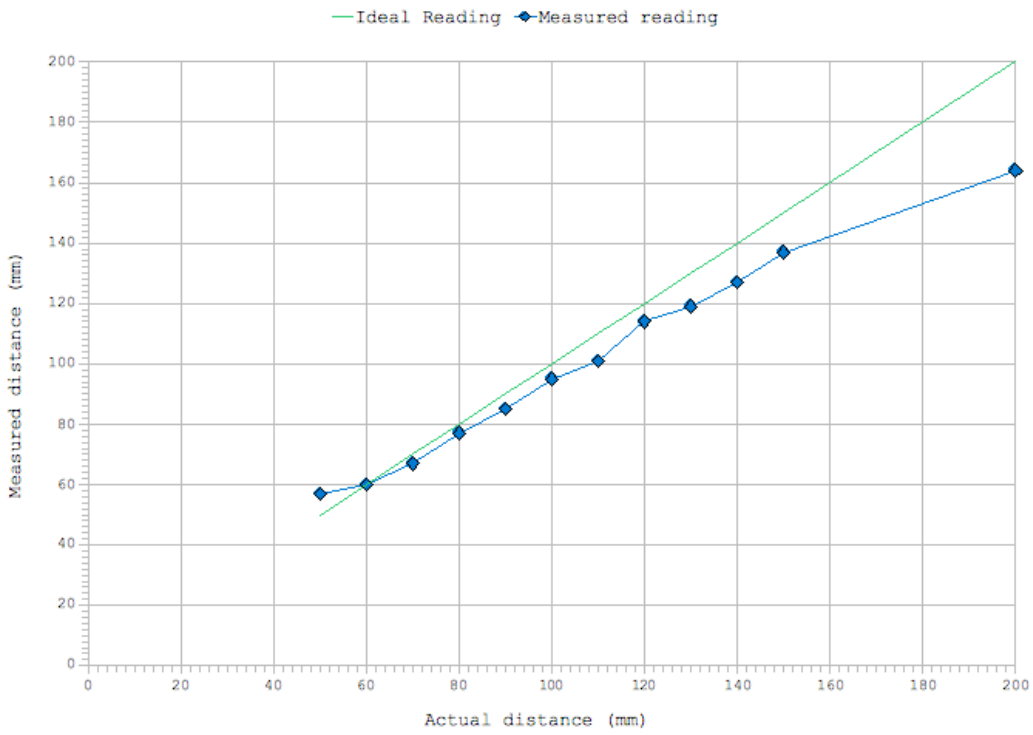


Figure 5.2: Graph showing distance readings from sensor against actual distance.

5.2 Sensors

5.2.1 Distance measurement

The sensors were positioned at a known distance and the reported value in millimetres was recorded every two seconds for a total of 20 samples. The mean distance was recorded and plotted against the real distance as shown in Figure 5.2. The measurements clearly follow a non-linear trend which implies that the regression derived from the datasheet was incorrect. This function was used to produce a lookup table to convert between ADC readings and distances in millimetres as described in Section 3.4.

To improve the correctness of the readings the lookup table would need to be replaced with one derived from actual readings. In hindsight, it would have been better to create the lookup table from actual readings rather than relying on the datasheet readings.

5.2.2 Accuracy

To measure the consistency of the distance measurement a sensor was arranged at five distances in the range of 50mm to 200mm. Samples were recorded every two seconds for a total of 100 samples and from this data the mean and standard deviation were calculated. From Table 5.2 it is clear that measurements at close range (<150mm) had good consistency with a standard deviation of under 2%. In real terms, an average

Mean distance (mm)	Range (mm)	Standard deviation (mm)	Standard deviation (%)
56	2	0.49	0.88%
99	5	1.26	1.27%
125	12	2.25	1.80%
144	12	2.62	1.82%
202	54	11.42	5.65%

Table 5.2: Consistency of distance measurement.

error of 2.62mm at a distance of 144mm was well within the requirements of detecting walls within the maze where each square was 180mm.

The low standard deviation at very short range (50mm) meant the measurements were adequate for to calculating the angular adjustments described in Section 3.5.2.

5.3 Mapping

5.3.1 Navigating through the maze

Without adjusting their position to compensate for errors, the performance was very variable. A robot was able to get through an 8x4 maze very occasionally but usually collided with a wall and lost track of its position.

Introducing position adjustments allowed the robots to able to follow a wall through the maze without becoming stuck or losing track of their square co-ordinate or direction. On the whole position adjustments were inperceptible but occasionally large angle or translation errors occurred which warranted more severe adjustments. To test the effectiveness of the position adjustments, the robot was placed in a straight 'corridor' with an angle error of 30° and left error of 40mm. When activated, it immediately turned to face the centre of the next square, moved towards it then turned back slightly to end up parallel with the walls.

As a test of robustness, one robot was left in a closed 8x4 maze and it successfully mapped the entire area five times before being deactivated.

5.3.2 Communicating with the base station

Assignment of ID numbers worked well with robots usually receiving a reply on the first of five requests. It is unclear how well this would perform as the number of robots, and corresponding packets, increased. A newly activated robot needs to get an ID as a priority but request packets could collide with packets being sent between the other robots in the area. If this were to become a problem the robots and base station could make use of the nRF24L01's five-byte hardware addresses. Each transceiver can simultaneously receive on up to six addresses and can transmit on any address. A dedicated

address could be used for transmitting ID requests and only the communications relay board would have to listen on that address. This would not reduce actual collisions but the robots' uninterested RF transceivers would ignore other packets at an earlier stage instead of passing them to the robot's command handler.

The base station sent regular POLL requests one robot at a time and the robot responded with a REPORT command. This worked well, with the python code updating local robot objects as reports were received from robots. This in turn was reflected on the graphical maze representation.

Several counters were implemented in the base station to compare the number of sent POLL commands with the number of received REPORT commands. The difference between the two gave an indication of how many packets were being lost. After fixing the problems discussed previously, no lost packets were witnessed over a period of five minutes when using two robots.

5.3.3 System in action

Appendix B shows photographs and screenshots of a successful mapping run of a 4x10 maze using two robots. The first snapshot was taken immediately after both robots had received an identification number. The second snapshot was approximately 30 seconds into a mapping run.

Chapter 6

Discussion

6.1 Results

A mobile robot platform was designed which was capable of moving itself around on flat terrain, keeping track of its position by dead-reckoning navigation and knowledge of the environment. It was able to take distance measurements of its immediate surroundings and communicate them to a base station using a widely available radio transceiver. Finally, it could make simple decisions about where to move next as well as interpreting and acting on commands received from the base station. The base station was able to display a realtime view of multiple robots in their environment.

6.2 Relevance

The system demonstrated the use of co-operating mobile robots to perform the useful task of exploring an unknown environment. This forms the basis of more advanced systems which are developing in the fields of reconnaissance, search and rescue, industrial automation and various military uses. The principles of operation were similar to those used by teams Stellar and Swarm in the MOD Grand Challenge, co-ordinating relatively simple mobile robots by using a central base station.

The system was designed with scalability as a key requirement. The command protocol was designed to be expandable and the microcontrollers were under-utilised, spending the majority of their execution time checking for new commands.

6.3 Shortcomings & Improvements

The robots occasionally encountered physical problems like slipping on uneven surfaces and clipping walls. In these situations they were not good at keeping track of their position, getting lost and reporting incorrect co-ordinates to the base station. This could be improved by monitoring the sensors during movement and comparing the

observed movement with an expected movement, aborting if the wheels appear to be slipping.

The graphical display was very simple and position updates were only displayed in complete squares. This could be improved by displaying more of the robot's reported data, for example the absolute position in millimetres rather than simply the co-ordinate. Similarly, it would be useful if an area of the screen was designated to graphically displaying a robot's sensor values and an indicator of current velocity and status.

The navigation approach was short-sighted, simply opting to follow a left wall. This was a wasteful method of mapping as dead-ends would be entered even if their adjacent squares had been previously explored. This could be improved either by allowing the robot to store the status of several adjacent squares or making the base station more sophisticated, issuing *move to* commands rather than allowing the robot to always wall-follow. In a similar way, speed improvements could be achieved by moving away from the single-square approach. Rather than stopping after entering every square, the base station could issue instructions to carry on without stopping at uninteresting squares.

Further improvements could be made to the PCB such as moving the header pins for the sensors and battery to prevent them from protruding from the side of the robot. Optimisation and size reduction may be possible as components were selected which were over-specified for the task.

Chapter 7

Conclusions

Overall the project was a great success. A robust mobile robot platform was designed and built which could be used for further research, teaching purposes or as a basis for a more advanced system.

Stepper motors were selected which were able to move the robot effectively and sensors were chosen to operate at an appropriate distance range. A chassis was purchased and modelled in CAD software and extra hardware was designed and manufactured to mount the sensors and PCBs. Additional electronics were designed to connect the microcontroller to the motors, sensors and RF transceiver and a printed circuit board was designed and made for this purpose.

A Luminary Micro microcontroller was selected and software was developed to control the stepper motors, calculate a distance value from infrared triangulation rangers, display information on an OLED screen and communicate using a widely available 2.4GHz radiofrequency transceiver. After designing a command protocol, the base station and robots were programmed to parse and respond to commands sent over the radio channel.

The robots were able to interpret their immediate environment and calculate movement routines to navigate around a simple maze environment. They tracked their position and reported their findings to a base station which showed the maze being constructed on a real-time graphical display.

7.1 Further work

One area for further work could be researching more sophisticated navigation algorithms. Simple wall-following does not guarantee that central squares will ever be entered and it often leads into dead-ends which may be unnecessary to map. There are a number of algorithms such as *flood fill* which are designed to find the shortest path to a certain point on a map. It may be possible to apply these to data from two robots and see performance increases.

The system could be easily developed for teaching about robotics, microcontroller

programming, motor control and radio communications. The existing code would allow students to achieve a basic level of functionality in a short time and then develop this further with different algorithms or approaches.

Breaking out of the uniform micromouse maze environment would greatly widen the applications of the system. The existing hardware and much of the software could be used to develop robots which can plot the floor plan of completely unfamiliar terrain. This would quickly start to resemble existing systems for search and rescue, reconnaissance and other tasks.

Bibliography

[1] *Overview*.
Available at: http://www.darpa.mil/grandchallenge05/overview.html
[Accessed January 6, 2009].

[2] *Stanford Racing :: Home*. Available at: http://cs.stanford.edu/group/roadrunner/old/techno
[Accessed January 6, 2009].

[3] *Welcome*. Available at: http://www.darpa.mil/grandchallenge/index.asp
[Accessed January 6, 2009].

[4] *Subsea Saviours,* The Engineer, 10 November 2008

[5] *The Challenge.*
Available at: http://www.science.mod.uk/engagement/grand_challenge/01_the_challenge
[Accessed January 6, 2009].

[6] *U.S. Army Expanding PackBot Contract With iRobot – Robots – InformationWeek.*
Available at: http://www.informationweek.com/news/security/government/showArticle
[Accessed January 7, 2009].

[7] *Aethon - Automated Hospital Delivery and Asset Management Solutions.*
Available at: http://www.aethon.com/
[Accessed January 6, 2009].

[8] *In It For the Large Haul,* The Engineer, 10 September 2008

[9] *Intelligent Systems Division - Research Areas - Projects - Industrial Autonomous Vehicle.*
Available at: http://www.isd.mel.nist.gov/projects/iav/
[Accessed January 12, 2009].

[10] *BAA: Heathrow transit system a world first.*
Available at: http://www.baa.com/portal/page/Corporate%5EAll+Press+Releases/4a1d6a
[Accessed January 7, 2009].

[11] Plisner, P *Transport That's Personal*, Rail Professional, December 2005, pp20-21

[12] *Safety Squared: Collision Avoidance Package.*
Available at: http://new.volvocars.com/enewsletter/07/winter/p04.html
[Accessed January 6, 2009].

[13] *EETimes.com - Self-Navigating Vehicle.*
Available at: http://www.eetimes.com/consumer/showArticle.jhtml?articleID=51200442
[Accessed January 5, 2009].

[14] NATIONAL DEFENSE AUTHORIZATION, FISCAL YEAR 2001, Section 220, page 40

[15] Sharkey, N. *Robot wars are a reality,* The Guardian, 18 August 2006

[16] Committee on Autonomous Vehicles in Support of Naval Operations, National Research Council, 2005.
Autonomous Vehicles in Support of Naval Operations, 2005, Page 4

[17] *Unmanned Systems Roadmap 2007-2032*
Available at: http://www.jointrobots.com/library02.php
[Accessed January 8, 2009]

[18] *IPTO :: Programs :: Learning Applied to Ground Robots (LAGR).*
Available at: http://www.darpa.mil/IPTO/programs/lagr/lagr.asp
[Accessed January 7, 2009].

[19] Oklobdzija, V.G., 2002. The computer engineering handbook,

[20] UK Frequency Allocation Table 2008 | Ofcom. Available at: http://www.ofcom.org.uk/radiocomms/isu/ukfat/ [Accessed April 23, 2009].

[21] File:Stepper motor 1.png - Wikipedia, the free encyclopedia.
Available at: http://en.wikipedia.org/wiki/File:Stepper_motor_1.png
[Accessed April 23, 2009].

Appendix A

Relevant Technologies

A.1 Sensors

There are a variety of sensors used in self-navigating machines, occupying a broad range of cost, complexity and specialism. As the resolution and amount of information collected by a sensor increases, so does the complexity of the signal processing required and speed of interface needed to connect to the sensor.

Using a combination of sensors can help build an even better image of the environment. Some laser scanners are equipped with conventional cameras and use software to apply the corresponding image textures to the final 3D model. These have been used in crime scene investigation for recording an accurate, human-friendly virtual model of a scene.

A.1.1 Camera systems

Digital cameras use a lens to focus light onto the photoactive region of an image sensor chip. An array of capacitors builds up charge proportionally to the intensity of light falling on a specific location. The charge on each capacitor is ultimately converted to a sequence of sampled voltages which represent light intensity values so the output of a camera is a two-dimensional array of light intensities which can be transformed into pixel values.

Image processing software is required to interpret the two-dimensional array of intensity values and identify shapes and objects. Stereo camera systems use two or more lenses to produce depth perception by stereopsis. Stereopsis is fairly computationally expensive so the computer hardware required is likely to be fairly substantial[1].

[1]Stereo cameras have the additional benefit of providing a highly natural human interface; the two images may be presented separately to each eye, allowing the brain to perform its own stereopsis and see the robot's view in 3D.

A.1.2 Laser scanners,

Light Detection and Ranging (LIDAR) systems, measure the time taken for a laser pulse to reflect from a surface and calculate a corresponding distance. They use rotating mirrors or prisms to scan the laser light across the field of view, building a three-dimensional 'point cloud'. Software 'reconstruction' algorithms are used to convert the point cloud into a mesh of small, 3D surfaces. These scanners can build a detailed, 3D map of the surrounding environment which can be processed by route finding algorithms.

A.1.3 Radar systems

Radar systems emit beams of radio or microwaves and measure the reflection to determine range and velocity of an object. Air traffic control systems operate at around 1-2GHz with a wavelength of 15-30cm. The longer wavelength means the range of radar systems is greater than that of laser scanner but at the expense of resolution. In general they are suitable for long range tracking of objects.

A.1.4 GPS and Inertial Measurement Units (IMUs)

GPS and IMUs are used to provide an absolute position within an environment. Global Positioning System (GPS) uses multiple orbiting satellites to pinpoint the receiver to a precise location on Earth. Coverage is severely limited by the urban environment as the ~1.5GHz signal is attenuated by buildings.

Inertial Measurement Units (IMUs) use a 3-axis gyroscope and 3-axis accelerometer to track movement in a technique called 'dead reckoning'; basing the position on a known last position plus a known movement. Position is derived by double integrating acceleration over time so errors can build up and therefore regular recalibration is necessary.

Using a combination of GPS and IMUs can overcome the problem of unreliable GPS reception and buildup of errors in an IMU. When the GPS signal is present the IMU can be constantly calibrated to prevent buildup of errors. When the GPS signal is lost, the IMU can be used as a replacement until the signal is regained.

A.1.5 One-dimensional rangers

Infrared and ultrasonic rangers are simple, one-dimensional sensors which can be positioned at intervals around a robot in order to produce a two-dimensional view of the surroundings. Individually these sensors detect only the presence of objects and are blind to colour, texture and shape. This makes them suitable for well defined, predictable environments with a simple, known floor plan.

Ultrasonic rangers sensors emit an ultrasonic pulse and measure the time to reflection. They include some calibration electronics which provide a digital interface so the output can be used directly as distance. The minimum range is limited by residual vibration in the transducer. As an example, a residual vibration lasting 100us and taking the speed of sound in air to be 343m/s (at 20degrees C) would equate to a distance of 34mm. Accuracy is a function of the wavelength, the ability of the sensor to distinguish between returning wavefronts and accuracy of digital timing electronics. The sensor measures the time to the first *detected* returning wavefront which may not be the first. If the sensor emitted n cycles during its 'ping' and if the first $n-1$ were undetected the reading would be incorrect by $\lambda \times n$. Further accuracy reductions arise from echoes from a non-uniform or large surface whereby every returning wavefront may be observed at multiple times, limiting the accuracy to a single wavelength.

Infrared intensity sensors use an infrared LED and phototransistor to measure the intensity of reflected light from an object. The output is a voltage level from the phototransistor so distance has to be derived as some function of the voltage level. The intensity of light reflection is affected by other factors such as ambient light, reflectivity of the surface and angle.

Infrared triangulation sensors from Sharp emit an infrared beam and use a small linear CCD array to triangulate the point and produce range information. The output is non-linear and has a strict minimum measuring distance but the sensitivity is high and there is a detailed datasheet. With a suitable calibration function these sensors could provide a very useful output.

A.2 Computer Systems

Computer systems used in self-navigating machines vary from small microprocessors to multi-kilowatt x86 clusters. The selection of computing resources largely depends on the number and complexity of sensors.

A.2.1 Microcontrollers (MCUs)

Microcontrollers (MCUs) are small processor systems are suited for sensors which are simple to interface with and do not require too much processing. They are usually low power devices, suitable for mobile applications. A microcontroller is an integrated circuit which usually contains at least the following features.

- Central Processing Unit - for executing instructions, handling interrupts etc.
- Oscillator - for providing a clock input to the CPU and other peripherals.

- RAM - volatile memory for storing code and data during execution.
- ROM - or some other kind of non-volatile memory for storing programs.
- General Purpose Input/Outputs (GPIOs) - for controlling or reading logic signals.
- Timers - for performing actions periodically or after a specified delay.

Other common features include Analogue to Digital Converters, serial interfaces, pulse width modulators and JTAG debugging interface.

Most manufacturers release a large number of similar microcontrollers with various different memory sizes and additional peripherals.

A.2.1.1 Microcontroller architectures

There are a number of notable competing microcontroller architectures from different manufacturers such as MSP430, ARM, AVR and PIC. Each architecture defines its own instruction set and therefore programs are written in processor-specific machine code code. Fortunately there exist free compilers, notably the GNU Compiler Collection, which allow the programmer to access most, if not all, functionality using a high level language such as C.

A.2.1.2 Microcontroller boards

Microcontrollers are usually purchased as standalone chips and are integrated into an application-specific PCB design. Some manufacturers sell evaluation boards which incorporate the microcontroller and provide a range of hardware for demonstrating the capabilities of the chip. These usually have breakout IO pads which can connect directly to the processor pins. Commonly found on evaluation boards are switches, LEDs, LCD screens, speakers, USB interface chips. Evaluation boards are usually sold at cost to show off the microcontroller and entice manufacturers to design the chip into their own system.

It is also possible to buy a simpler breakout board which provides a breakout pad to connect to the pins of the microcontroller. This enables prototyping of chips which would otherwise be imposible due to the use of surface mount technology.

A.2.2 x86 Architecture

The x86 architecture, originally the Intel 8086, is the most common instruction set architecture in personal computing and can be found in a variety of higher performance self-navigating applications. Because of their widespread use in personal computing (especially gaming) a driving design requirement is speed and performance.

Installed in motherboards of various form factors, x86 CPUs have 32 or 64 bit address spaces and high speed data buses such as IEEE1394 (FireWire) for connection to high data-rate hardware devices.

They are higher performance than microprocessors in terms of clock speed and memory size which allows them to run computationally expensive image processing and route calculation software. Processors come in multiple core variations and systems can be networked to form a cluster, allowing calculation to be distributed across multiple processors.

A.2.3 Field Programmable Gate Arrays (FPGA)

FPGAs are programmable logic devices containing many millions of logic elements which can be configured using a Hardware Description Language. Their complexity is such that entire x86 CPUs can be emulated in a section of an FPGA. Because of their large number of programmable gates they can also be used to implement a number of identical hardware devices which can work in parallel and operate many times faster than equivalent software in a single CPU. This could be used to speed up specific tasks such as processing data from cameras or laser scanners.

FPGAs can load their configuration from an EEPROM or Flash memory when powered up. It is therefore possible to modify and recompile the hardware description and upgrade the firmware on the FPGA without the need for replacing any components.

Because power is required to maintain the state of the programmable gates, an FPGA has a higher power consumption than an application specific chip which makes it less attractive for battery powered designs.

A.2.4 Interrupt Routines

Interrupt service routines (ISRs) are subroutines which are usually executed as the result of a hardware condition such as a falling edge of a digital input. Normal program execution is *interrupted* in order to execute the ISR.

Interrupt routines are useful for handling events from external hardware devices. An example would be a communication device with a limited internal buffer. An interrupt routine could be triggered by the device to transfer the buffer into memory immediately after data is received to prevent the buffer from overflowing and potentially losing data.

Interrupts also allow a processor to only execute code when that code is required and save valuable CPU cycles. Implementing the above example without interrupts would mean frequently checking for data in the buffer and acting if some is found. If receiving data was rare then many cycles would be wasted by inspecting an empty buffer. Additionally, if the checking was not frequent enough then the buffer could still overflow between checks.

It is normally desirable to make interrupt service routines as short and fast as possible. During the execution of an ISR, other interrupts may be triggered and these are unable to run until the first has completed[2]. This may lead to data being missed as in the case of the communication device. Interrupts are therefore not suitable for functions which have necessary timing delays or are otherwise slow to complete.

A.2.4.1 Considerations

Because interrupt service routines are executed independently of the main program code, all communication with the main program must be done through global variables. When dealing with these variables in the main program code it is important to remember that they may change in between instructions. Where multiple variables are used for a related purpose, they must be changed in an order which is 'safe' if an interrupt routine runs in between setting the variables. For example, if the main program dynamically increased the size of an array and set a variable to signify the new size of the array, the size should be set *after* the array has been lengthened. In the reverse order, an interrupt routine may read the size and index the array past the end of its allocated memory.

Debugging is made more complicated by interrupts because the state of the program is harder to predict. When precise timing is involved interrupts can alter the timing in a sporadic way which is very difficult to debug.

A.3 Communication hardware

The choice of communication hardware largely depends on the operating range and bandwidth requirements. A system which uses a high-altitude UAV to transmit high resolution photographs could have very different requirements from a swarm of tiny robots used for assembling circuit boards.

A.3.1 Infrared communication

Infrared communication uses an infrared light-emitting diode and silicon photodiode to send data serially using a protocol such such as IrDA. A common application is in remote control handsets for controlling televisions and other equipment. It is a low power technology which suits mobile applications but it relies on line-of-sight and is less effective outdoors with the sun's bright infrared interference.

[2]Some processors offer support for prioritisation of interrupt service routines.

A.3.2 Radio frequency transceivers

Radio-frequency (RF) transceivers are input/output devices which can send and receive data at a certain frequency, or range of frequencies. They come in a huge variety of forms. In the simplest, transmission is achieved by 'bit bashing' or manipulating a digital input and bytes are sent over air as they are received by the chip.

Other chips are sold on a board with a microcontroller such as a PIC which can be programmed to act as an interface between the RF chip and another microprocessor. This can provide features such as acknowledgement and automatic retransmit.

A.3.3 Operating frequencies

Radio frequency (RF) communication devices come in many different forms but comprise a radio transceiver and transducer which operate at an agreed frequency, or sequence of frequencies. They can operate out of line-of-sight but the electromagnetic wave is attenuated by buildings and other obstacles depending on the wavelength. The operating range is dependant on both the transmitting power and the wavelength.

There are twelve internationally agreed frequency bands which are designated for use by industrial, scientific and medical (ISM) purposes and as such there is a requirement that communication equipment must be tolerant of harmful interference in these bands. Many countries have taken the decision that since a certain level of harmful interference is ordinarily present, some bands could be made exempt of licensing restrictions. In the UK, OFCOM allows licence exempt transmission in regions of three ISM bands, centred around 915MHz and 2.45GHz. Additionally, a frequency range centred at 865MHz has been designated by OFCOM[20] as licence exempt for short range devices.

A.4 Motors

The robot had no means of detecting its own absolute position so it relied on dead-reckoning navigation. In order to explore the maze the robot needed motors which it could move through a known angle to keep track of its position. This can be achieved either through use of *feedback* or by using motors which are themselves able to move through a specified angle based on an input.

Ensuring the motors have rotated through a given angle does not offer any assurance that the robot has actually moved. The wheels may slip on the terrain or an input may be beyond the capability of the motor. It is important to consider the physical capabilities of the hardware when writing the software to control it.

A.4.1 DC Motors

DC motors move at a speed which is related to the average input voltage applied at their terminals. Their speed can be approximately controlled using a pulse-width modulator, in which the mark-space ratio of a square wave is adjustable, but accurate position control is not possible without additional hardware.

A.4.2 Servo motors

Servo motors incorporate a rotary encoder as a feedback mechanism. A servo controller accepts a position as an input and attempts to move the motor to the desired position by reducing the error reported by the feedback.

A.4.3 Stepper motors

Stepper motors have a number of internal windings arranged as shown in Figure A.1[21]. The windings are arranged to make four electromagnets around the stator with teeth facing the centre of the motor. The rotor is also toothed having geometry like a gear.

When one electromagnet is energised its teeth are attracted to the nearest teeth of the rotor, pulling the rotor into alignment with that electromagnet. The teeth of the electromagnets are offset from each other such that when the rotor is aligned to one it is not aligned to the others. Energising a different winding causes the rotor to move to realign itself with that electromagnet. By energising the four windings in sequence the rotor can be made to move continuously at a precise angular velocity or perform a single rotation of an exact number of *steps*.

A.4.3.1 Full step drive

There are a number of possible energisation sequences used for driving stepper motors. Full stepping takes advantage of the maximum torque available to the motor by always simultaneously energising two windings. The four coils could be switched by four digital outputs on a microcontroller cycling through a simple four-state output. The drive electronics can be as simple as four transistors.

A.4.3.2 Half stepping

Half stepping can use the same driver electronics as full stepping but the excitation sequence alternates between energising one and two windings. A sequence implemented on a microcontroller would have eight states rather than four. Using half stepping provides double the angular resolution, effectively halving the step size. However, alternate steps use only half the torque available to the motor.

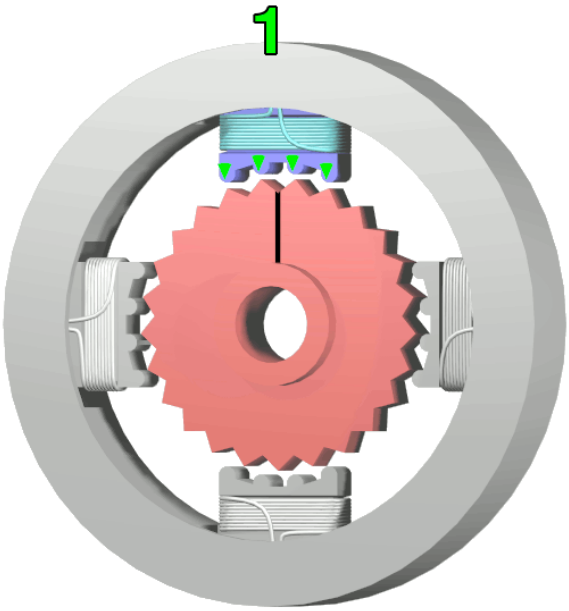


Figure A.1: Stepper motor illustration.

A.4.3.3 Microstepping

Microstepping aims to increase the angular resolution further by controlling the voltage in two adjacent electromagnets to follow an approximation of a sine-cosine waveform. A high performance microstepping driver may use 256 discrete voltage levels to approximate the sinusoidal waveform. This would increase the angular resolution by a factor of the same number which increases the smoothness of the motion and reduces the vibration at low speed.

Controlling voltage level rather than simply switching it requires more drive electronics such as operational amplifiers. If implemented in software, a 256 level microstepping sequence would require eight digital outputs per winding.

A.4.3.4 Physical considerations

At low speeds stepper motors create vibration and the rotation is not smooth[19]. This could cause slippage if used to drive wheels so there is a minimum speed at which they should be run.

They also have a maximum speed which is the result of the reduction of torque as the speed increases. For a stepper motor to produce the full rated torque the windings must be at their rated current at each step. As the speed increases, the inductance of the winding and reverse EMF induced by the rotor resists the current change and reduces the torque. The maximum speed is that at which the motor is unable to move the load torque.

Stepper motors are susceptible to an oscillatory velocity response when a high rate of change of acceleration (jerk) is applied. Linear acceleration followed by constant

velocity is an instantaneous change in acceleration so should be avoided.

Finally, stepper motors have an angular resolution which is a function of the geometry of the rotor. Typically there are 200 steps per revolution giving an angular resolution of 1.8° although this resolution can be effectively reduced by the use of half stepping and microstepping.

A.5 Inter-device Communication

A.5.1 Serial Peripheral Interface (SPI)

The Serial Peripheral Interface (SPI) is a full duplex synchronous serial data link suited for short range connection between master and slave devices. SPI is a four-wire bus in which multiple slave devices may be connected to a single master and enabled by use of a Slave Select logic signal. The four logic signals are described below.

SCLK (serial clock) which is driven by the master. This must be configured at a frequency that every slave device can support.

MOSI (master out, slave in) is used to transfer data from the master to the slave device, for example a command from a microcontroller to an analogue to digital converter.

MISO (master in, slave out) transfers data from slave to master.

SS (slave select) is used to enable a slave device. Using independent slave configuration the master must provide one slave select signal for every slave connected. Some devices support daisy-chain configuration whereby the output of the first slave is connected to the input of the next. When slave devices are not selected, most peripherals tri-state their MISO and MOSI signals to allow other peripherals to use the bus.

At the start of a data exchange the master sets one Slave Select signal to logic high, optionally waits for a period, then enables SCLK. During each clock period the master sends a single bit on the MOSI line and simultaneously receives a bit from the peripheral on the MISO line. The slave device is only able to send while the master is driving the clock so it is the responsibility of the master to ensure it sends enough clock cycles to receive all the data it requires.

A.5.2 Universal Asynchronous Receiver/Transmitter (UART)

A Universal Asynchronous Receiver/Transmitter is a hardware device which converts data between parallel and serial form. The UART is responsible for the timing, parity and frame detection of the communication but does not directly generate line voltages;

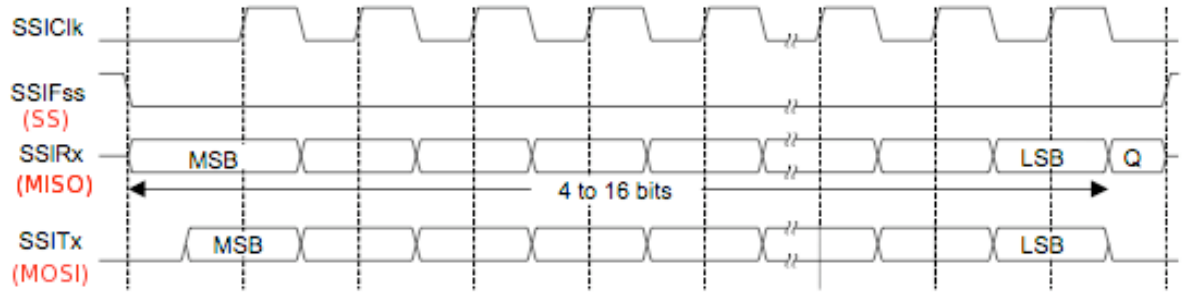


Figure A.2: SPI timing diagram.

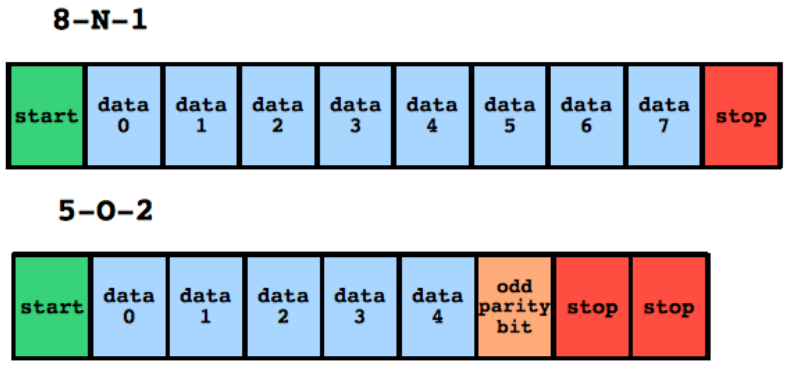


Figure A.3: UART data frames for 8-N-1 and 5-O-2

these are handled by additional line driver chips specific to a serial link such as RS-232 or Universal Serial Bus (USB).

In order to communicate, two UARTs need to agree on line speed, parity check scheme and number of *data* and *stop* bits. Data frames comprise a *start* bit, five to eight *data* bits, an optional *parity* bit then one to two *stop* bits.

A typical configuration of UART parameters is written as 115200/8-N-1 which signifies a line speed of 115200bps, eight data bits, no parity check and 1 stop bit. The line speed includes bits used for *framing* which in this case would be one start bit and one stop bit, giving a utilisation of 80% and a real bit rate of 92,160bps. Figure A.3 shows two examples of UART data frames.

A.6 Printed Circuit Board (PCB)

A.6.1 Analogue Digital Separation

Digital circuits operate at only discrete signal levels: ideal switching between a digital '1' and '0' involves instantaneous transitions between high and low level signals. A square waveform contains many harmonics which produce electromagnetic noise across a range of frequencies.

Electromagnetic noise affects analogue circuits by directly altering the voltage level.

Using an analogue output from a ranging sensor means that noise would directly corrupt the distance measurement.

When designing PCBs it is good practice to separate the analogue and digital components by placing them physically far apart.

A.6.2 Grounding

To reduce the vulnerability of a circuit to electromagnetic radiation it is often desirable to fill the unused parts of a board with a ground plane. Tracks which are known to carry noise (such as digital IO lines) can be surrounded by the ground plane which helps to eliminate the noise propagation of noise by sinking it to ground.

Ground planes also help to ensure all components are comparing voltage signals against the same reference point by providing a low impedance path. Circuit design is simplified as the ground plane is often more accessible than designing specific ground tracks.

A.6.3 Capacitors

Many power supplies including voltage regulators produce noise which can produce unwanted effects in both digital and analogue circuits. Additional sources of noise arise from rapidly switching inductive loads such as windings in stepper motors.

Decoupling capacitors are used to isolate two circuits and prevent noise from one circuit interfering with the other. Connecting a capacitor from one part of a circuit to ground acts as a low-pass filter by conducting AC to ground while leaving DC unaffected.

Appendix B

Photographs and Electronic media

B.1 Photographs

Figures B.1, B.2 and B.3 show the robot hardware, Luminary Micro LM19683S Evaluation board and the system during a mapping run.

B.2 Screenshots

Figures B.4 and B.5 show the graphical user interface shortly after activating two robots and part way through a mapping run.

B.3 Supporting CD-ROM

Enclosed is a supporting CD-ROM containing the following resources.

- An electronic copy of this report (PDF format).
- Installation files for most software used throughout the project.
- PCB schematics, track layout and Gerber files.
- SolidWorks 2007 designs of AIRAT2 chassis, sensor housing and PCB mounting plate.
- C source code for robots and communications relay board.
- Python source code for base station graphical display.
- Component data sheets.

If for any reason the CD-ROM is missing, please contact the author who will be happy to replace it.

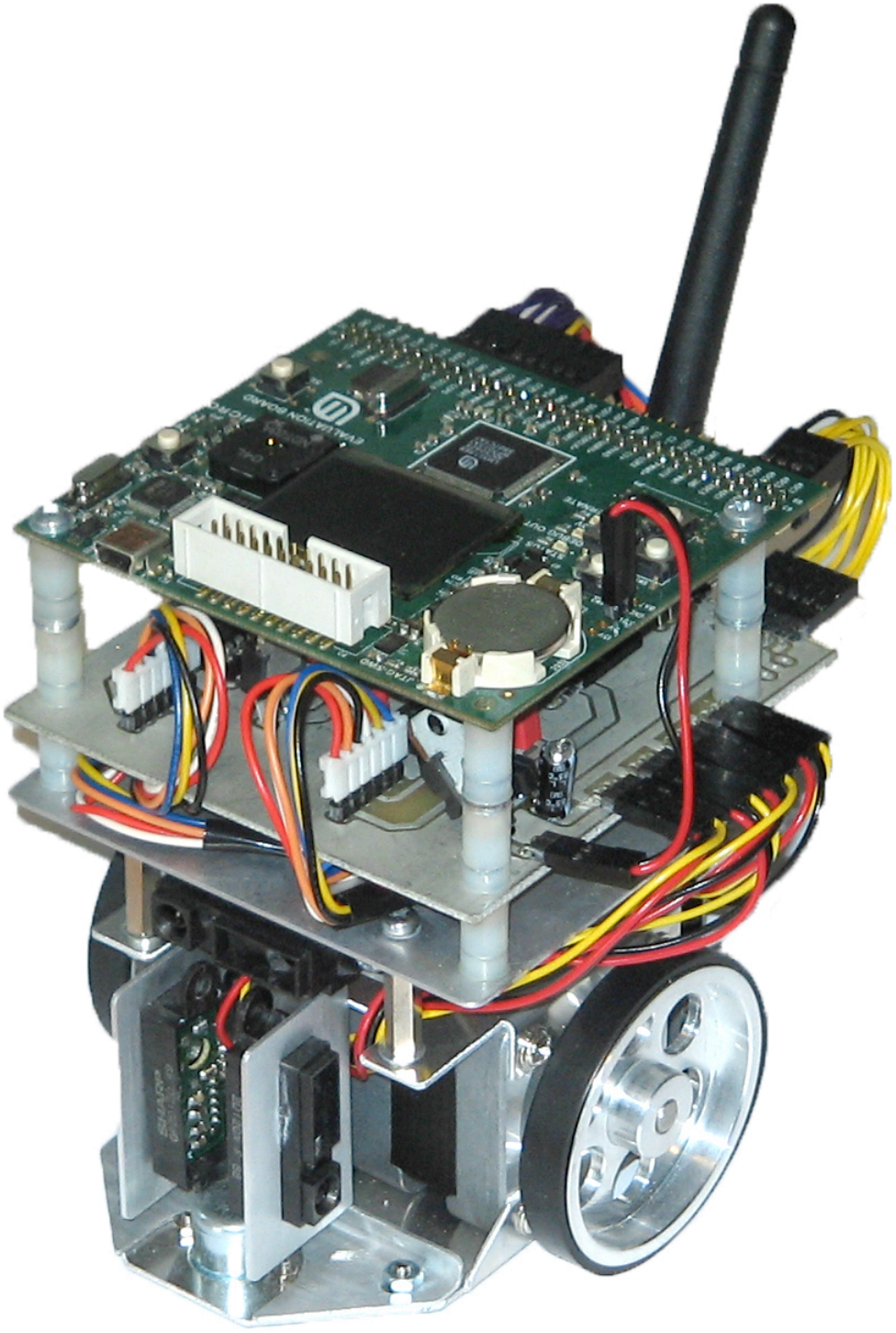


Figure B.1: Photograph of the mobile robot hardware.

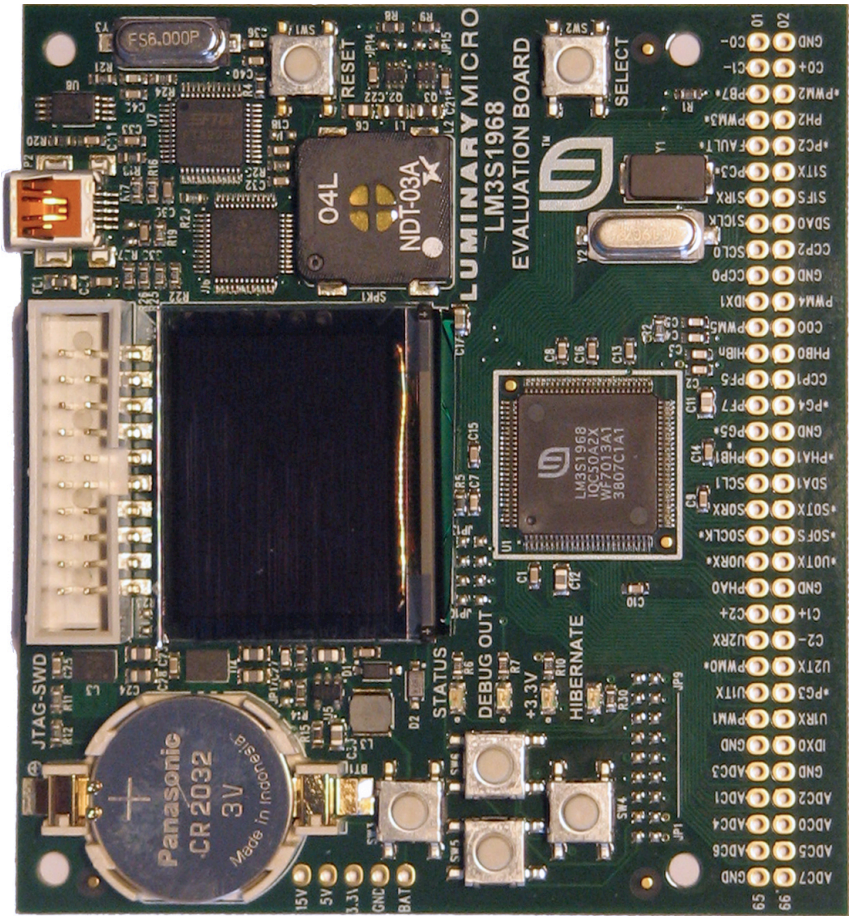


Figure B.2: Photograph of the Luminary Micro LM3S1968 Evaluation board.

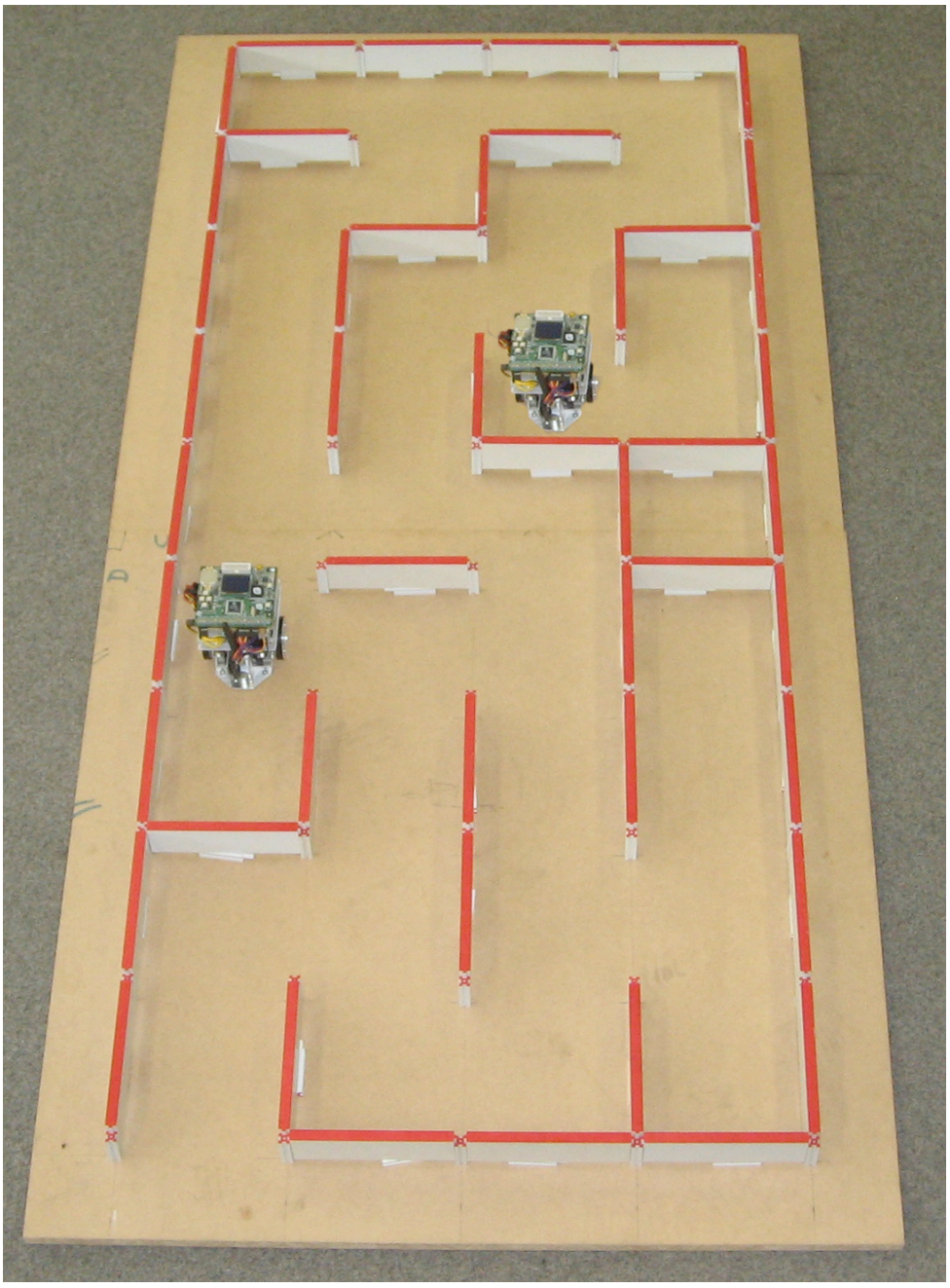


Figure B.3: Photograph of two robots mapping a 4 x 9 maze after approximately 30 seconds of operation.

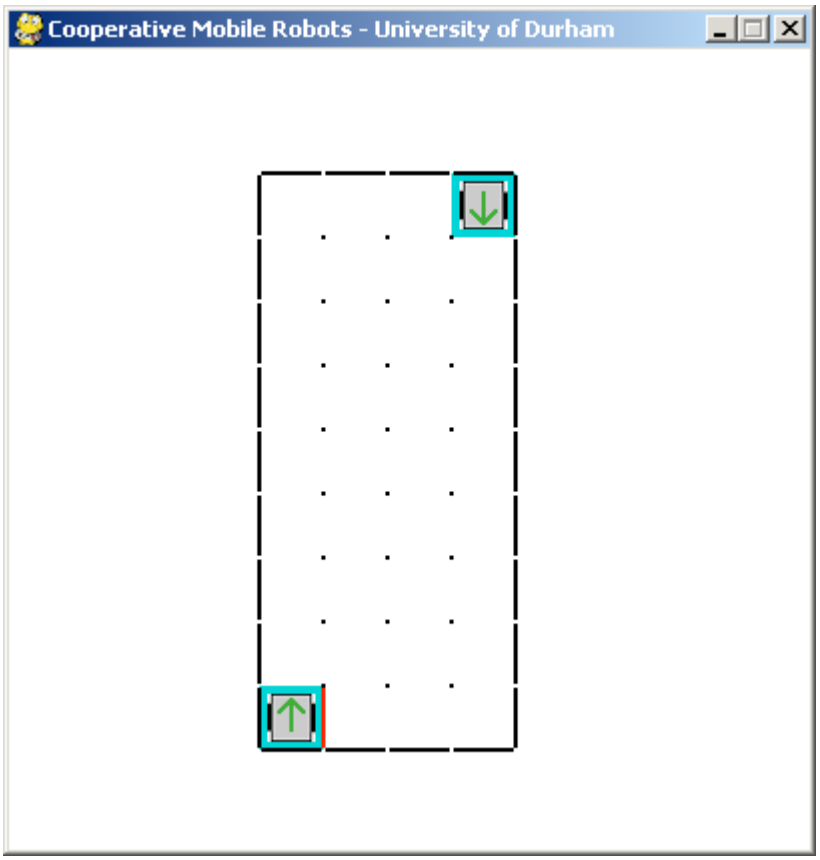


Figure B.4: Screenshot of graphical base station immediately after activating two robots.

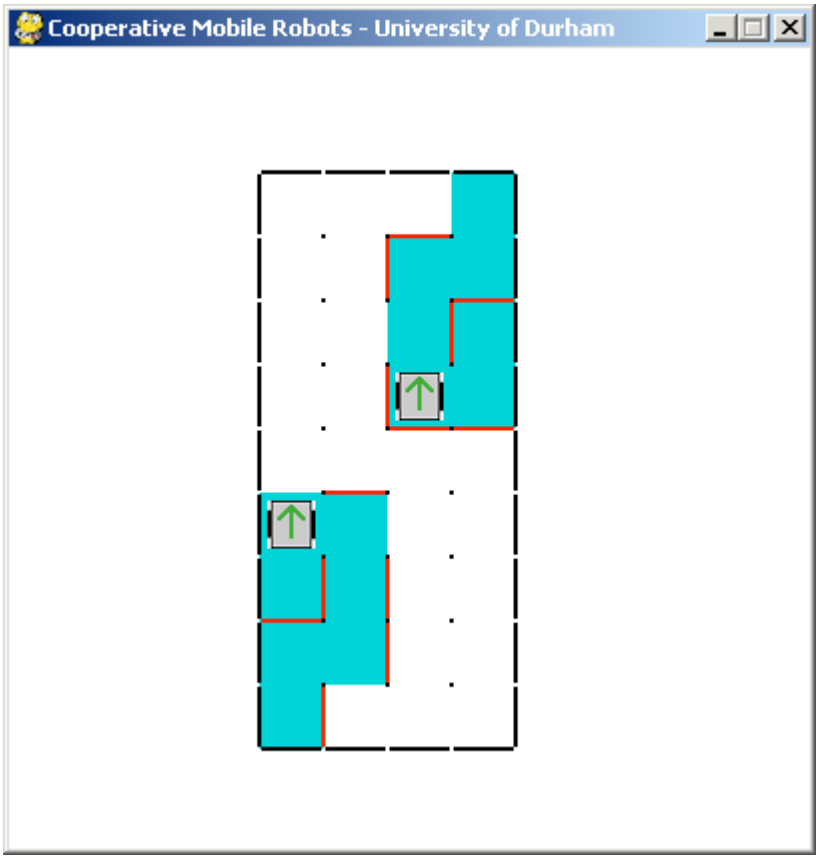


Figure B.5: Screenshot of graphical base station after approximately 30 seconds of operation.

B.4 Website

For video content of the system in action as well as details of any further work which has been carried out since the completion of the project, see the accompanying website at the following address.

`http://www.slyllama.net/mobilerobots`

Please direct any questions or suggestions to the contact details listed on the website.

Appendix C

Risk Assessment and COSHH

The following pages show the Risk Assessment and COSHH (control of substances hazardous to health) forms which were completed prior to commencing any practical work.

UNIVERSITY OF DURHAM - Health and Safety Office

RISK ASSESSMENT FORM

Summary of Assessment - Tick Box	
CATEGORY A WORK	☐
CATEGORY B WORK	☐
REFER TO HEALTH & SAFETY OFFICE	☐

FOR RESEARCH WORK
(including practical undergraduate projects)

PART 1. PERSONAL DATA AND DETAILS OF PROJECT/EXPERIMENT

Department ...Engineering..... Laboratory .....E268..... Tel. Ext. .....

Name ...Paul M Furley..... Supervisor .....Jim Swift.....

Male/~~Female~~ ~~Staff/Technician~~/Student UG/~~PG~~

Degree/~~Higher Degree Course~~ Starting Date .....13th October 2008.....

Proposed finishing date of Project/Experiment .....

Title of Project/Experiment .....Interacting Maze Solvers.....

.....

Description of Project/Experiment. Give a brief description of Project/Experiment, attaching copy of grant submission if appropriate.

.....Investigation of small robots traversing a physical maze in order.....
.....to find a solution quickly. Some electromechanical construction.....
.....will be carried out and a 'micromouse' maze will be built.....
.....
.....
.....
.....
.....
.....
.....
.....
.....

Instructions

1. This form is to be completed by the Supervisor in consultation with (any) co-workers.
2. It encompasses postgraduate research and individual undergraduate project work, and individual research. (the summary assessment is, of course, not appropriate where the researcher works individually.)
3. It concerns practical work using materials or equipment, and field work but not theoretical or computing work.
4. It must be signed by the Supervisor or senior researcher and by the co-worker (or student) to indicate that HE or SHE has read it. After completion tick the summary box on the front.
5. One copy to Department records, one copy to co-worker/student, plus one to HEALTH & SAFETY OFFICE *only* if health surveillance, air monitoring or respiratory/ear protection required.
6. The aim of the Assessment is to satisfy Regulation 3 of "The Management of Health and Safety Regulations 1992". (Risk Assessment). Should the work depart from the aims, methods or materials envisaged then a new assessment must be carried out.

PART 2. NATURE OF POSSIBLE HAZARDS

Chemical Hazards — Are chemicals, including proprietory materials, to be used — ~~Yes~~/No

← if No

Is Good Lab. Practice Sufficient (see COSHH Code of Practice) — Yes/No

← if Yes

* Is there a cyanide/fluorine, 'very toxic' hazard — ~~Yes~~/No/~~Possibly~~
* Is there a volatile material hazard — ~~Yes~~/No/~~Possibly~~
* Is there an explosive or violent reaction hazard — ~~Yes~~/No/~~Possibly~~
* Is there a carcinogenic, mutagenic or teratogenic hazard (see COSHH Code of Practice - Carcinogens) — Yes/~~No/Possibly~~
* Will hazardous dust or fume be created — Yes/~~No/Possibly~~

Ionising Radiation Hazards — Does project/experiment involve radioisotopes or radiation sources (α, β, γ, X neutron emitter/apparatus) — ~~Yes~~/No/~~Possibly~~

← if No

if Yes Submit written scheme of work to Departmental Radiation Protection Supervisor, for approval before work starts

* Is the High-Activity Cell (Cobalt 60 source) to be used — ~~Yes~~/No/~~Possibly~~

Laser Hazards

* Are class 3B or 4 lasers to be used — ~~Yes~~/No/~~Possibly~~

Non-Ionising Radiation Hazards, Noise, Vibration — Is non-ionising radiation, noise or vibration used/generated in the project/experiment. — ~~Yes~~/No/~~Possibly~~

← if No

if Yes

* Ultra-violet YES/NO, infra-red YES/NO, Microwaves YES/NO, Radio-frequency YES/NO, intense magnetic or electric fields YES/NO, Noise > 85dBA YES/NO, Vibration which reaches the hands or body YES/NO

Electrical Hazards — Is electrical equipment, other than tested 240V portable appliances connected to mains supply, to be used — ~~Yes~~/No/~~Possibly~~

← if No

* Will any electrical equipment using more than 50V a.c. or 100V d.c. be used out of doors or in wet conditions — ~~Yes~~/No/~~Possibly~~
* Are voltages > 1000V to be used — ~~Yes~~/No/~~Possibly~~
* Are there exposed circuits and terminals with voltages > 50V a.c., 100V d.c. — ~~Yes~~/No/~~Possibly~~
* Are portable generators to be used — ~~Yes~~/No/~~Possibly~~

Robotic Hazards — Is robotic equipment to be used — Yes/~~No/Possibly~~

← if No

* Are restricted areas to be entered — ~~Yes~~/No/~~Possibly~~
* Are mobile robots to be used — Yes/~~No/Possibly~~

Other Hazards

* Are there any other hazards which pose unusually high risks — ~~Yes~~/No/~~Possibly~~
* Are explosives to be used — ~~Yes~~/No/~~Possibly~~
* Does stored energy, e.g. machine rotation, present a hazard — ~~Yes~~/No/~~Possibly~~
* Does the work entail entry to cold rooms or hot rooms, and/or sealed rooms which can be locked — ~~Yes~~/No/~~Possibly~~
* Does the work entail work in the field — ~~Yes~~/No/~~Possibly~~
* Are there pressure systems or closed systems in which over-pressure or under-pressure can occur — ~~Yes~~/No/~~Possibly~~
* Is dangerous apparatus/equipment ot be used, e.g. Chainsaw, Stihl Saw, Stunning guns — ~~Yes~~/No/~~Possibly~~

Biological Hazards — Are biological materials to be used (including exposure to animals or plant allergens). — ~~Yes~~/No/~~Possibly~~

← if No

if Yes Complete BIO-COSHH form and if the answer to every other question is NO end this risk assessment. If other hazards are present, the Supervisor shall decide if this or the BIO-COSHH form, or both, is most appropriate

If any starred (*) items are marked YES or POSSIBLY then include in assessment.

PART 3. DETAILS OF ASSESSMENT

The aim of the assessment is to identify preventative and protective measures which must be provided and used so that risks to the experiment and others are controlled. In most cases existing Departmental or University procedures will be sufficient but should be referred to.

If any question marked * in part 2 is answered YES then provide written details including safety procedure below.

Use of solder - if carrying out a prolonged session then use portable ventilation, otherwise carry out in a well ventilated area.

The robots used are approximately 6 inches cubed and travel relatively slowly. Only powered by small batteries

GENERAL SAFETY INFORMATION RELATING TO PROJECT/EXPERIMENT

Means of Containment (if applicable) to be used when appropriate	Fume Cupboard	Yes/No	.....
	Biological Safety Cabinet	Yes/No	Grade
	Total Enclosure	Yes/No	
	Local ventilation	Yes/No	
	Other	Yes/No	Specify .....
Personal Protection, to be worn when necessary	Lab Coat	Coveralls	Coveralls and Hood
	Hand Protection	Yes/No	Type .....
	§ Respiratory Protection	Yes/No	Type .....
	§ Ear Protection	Yes/No	Type .....
	Eye Protection	Yes/No	Type .....
	Foot Protection	Yes/No	
	Other		Specify .....

§ refer to Health & Safety Office.

PART 4. WASTE DISPOSAL

Will Hazardous wastes arise? ~~Yes~~/No

Waste Disposal (Departmental) practice .....

.....

.....

.....

PART 5. MONITORING

Is air monitoring required? (Materials .....) ~~Yes~~/No

If YES, refer to Health & Safety Office.

Is health monitoring required? ~~Yes~~/No

Compulsory, ionising radiations classified worker, exposure to animals/plant allergens, exposure to certain chemicals (see COSHH).

Voluntary, 3B/4 laser use. noise and vibration exposure, regular exposure to toxic chemical, field work.

Supervisor to submit names to Health & Safety Office (for referral to Occupational Health Adviser) with this form.

PART 6. EMERGENCIES

Special action to be taken after Accident. Indicate special measures needed for the work described in the assessment, e.g. first aid, major spillage, electricty/water failure, work in remote or hazardous environments.

No special actions required.

.....

.....

PART 7. SUMMARY

Supervisors should identify areas of work within the project in the categories below, which are not mutually exclusive. Category B is the expected initial assessment as all practical work requires the supervisor's approval. Following demonstration and training, the supervisor may re-categorise some work to A. The Supervisor must give instructions and advice on safeguards, determine the extent and frequency of supervision, and state the name of any others used in supervision.

A - NOT to be undertaken without direct supervision.

B - NOT to be started without supervisor's approval.

WORK	CATEGORY	SUPERVISOR
All work on this project	B	JSS

Signature of Supervisor ..... Date 14/10/08

Signature of Co-worker/student having read the assessment Paul Furley Date 14/10/2008

UNIVERSITY OF DURHAM – SCHOOL OF ENGINEERING

COSHH ASSESSMENT

Assessor (please print) PAUL FURLEY **Procedure** SOLDERING COMPONENTS ON PCBS **From** 14/10/08 **To** 31/03/09

Material	Label	Risk Nos.	Safety Nos.	Likely Route of Exposure	Protection Measures To Be Used	First Aid	Fire Fighting	Spillage	Waste Disposal	Other Properties
SOLDER		R20 R25	S23, S51 S38	DURING SOLDERING OF MICROCONTROLLER BOARD	PORTABLE VENTILATION IF PROLONGED USE	—	—	—	—	

Assessor Signature Paul Furley **Date** 14/10/08 **Supervisor Signature** [signature] **Date** 14/10/08

Appendix D

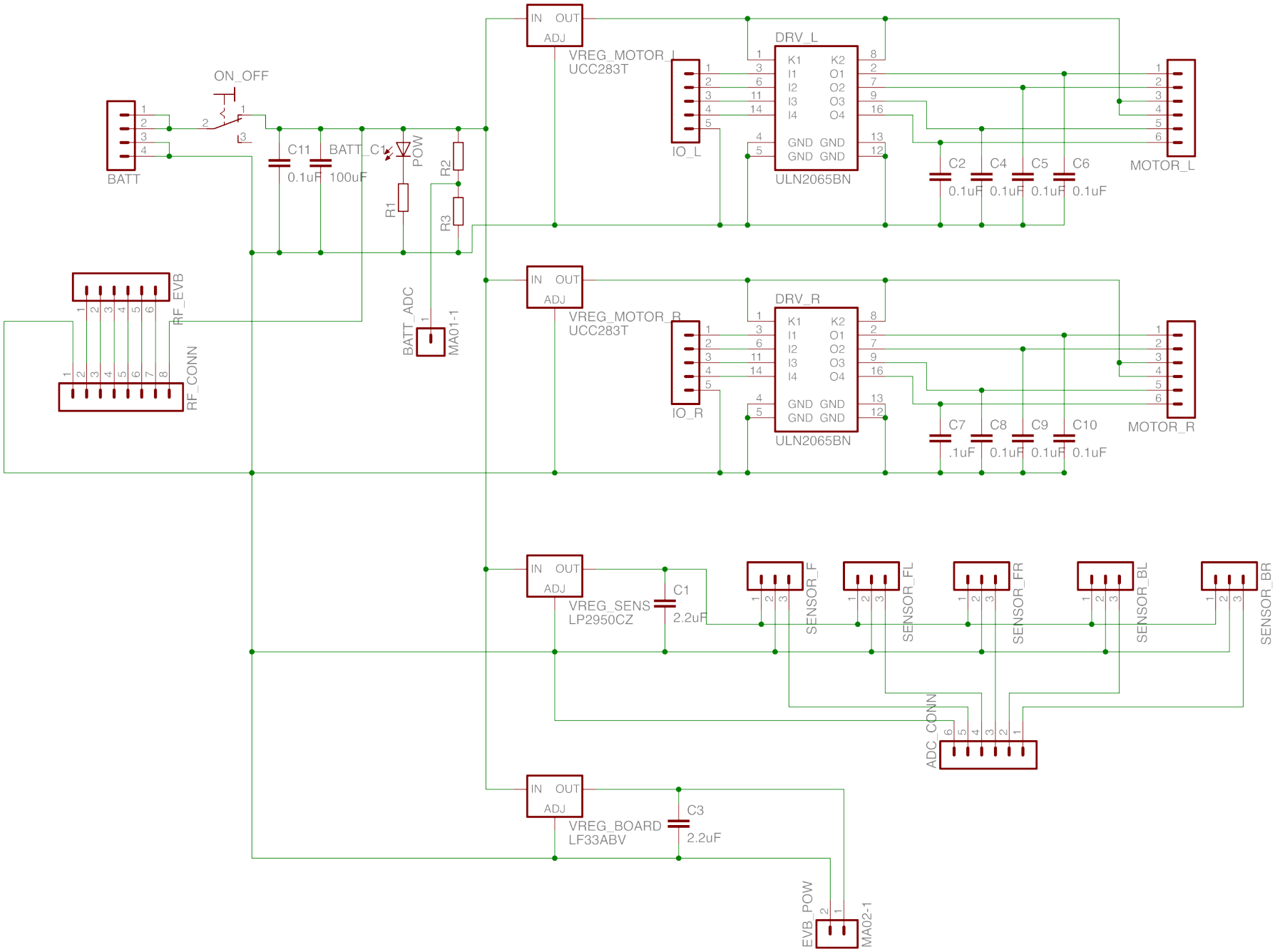
PCB Designs

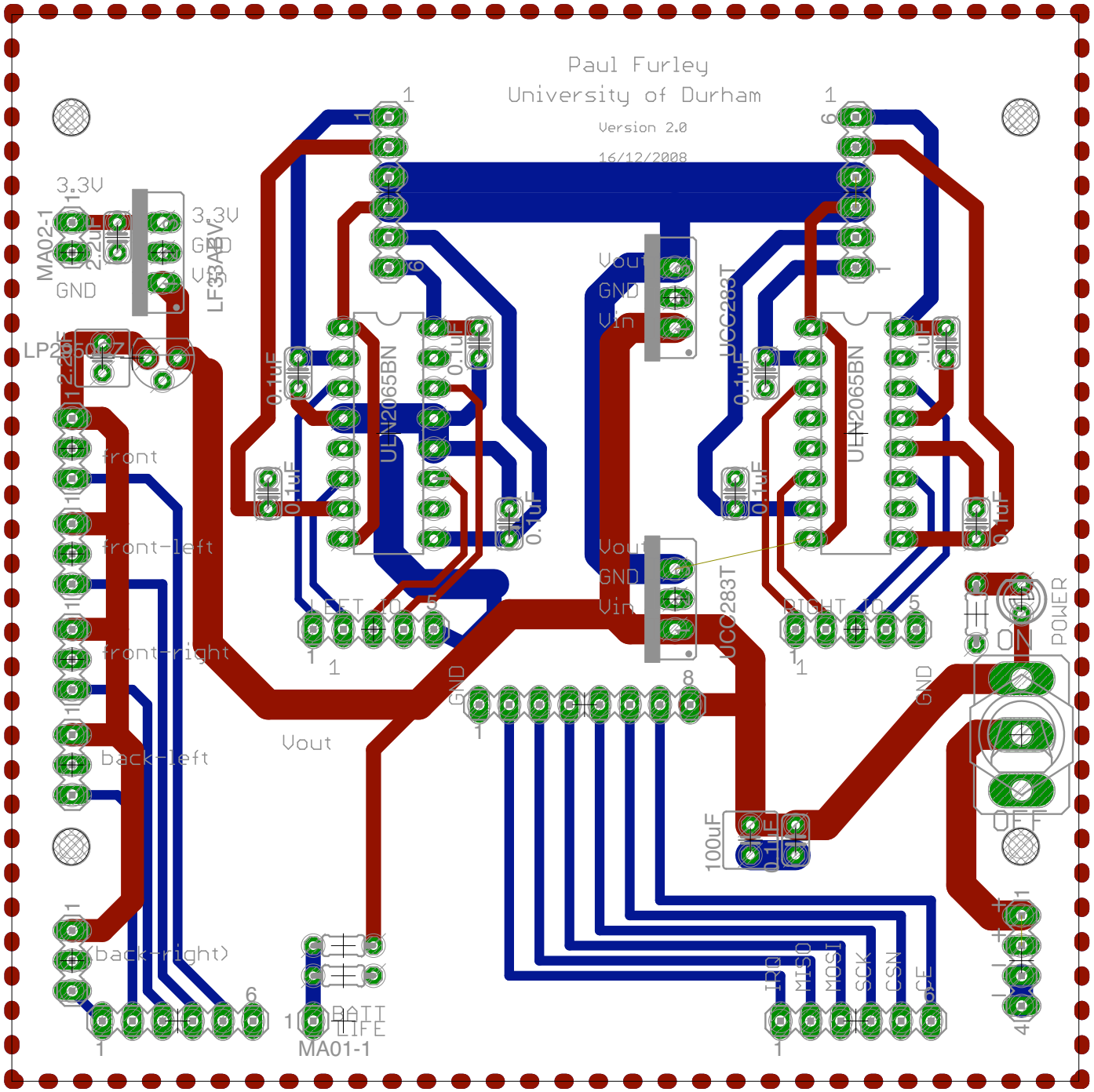
D.1 Schematic

D.2 Track layout

D.3 3D model

D.4 Gerber files


ON_OFF
BATT
C11
0.1uF
BATT_C1
100uF
POW
R1
R2
R3
BATT_ADC
MA01-1
RF_EVB
RF_CONN
VREG_MOTOR_L
UCC283T
IO_L
DRV_L
ULN2065BN
C2
C4
C5
C6
0.1uF
MOTOR_L
VREG_MOTOR_R
UCC283T
IO_R
DRV_R
ULN2065BN
C7
.1uF
C8
C9
C10
0.1uF
MOTOR_R
VREG_SENS
LP2950CZ
C1
2.2uF
SENSOR_F
SENSOR_FL
SENSOR_FR
SENSOR_BL
SENSOR_BR
ADC_CONN
VREG_BOARD
LF33ABV
C3
2.2uF
EVB_POW
MA02-1


Paul Furley
University of Durham
Version 2.0
16/12/2008
3.3V
MA02-1
2.2uF
GND
LF33ABV
LP2950Z
front
front-left
front-right
back-left
back-right
0.1uF
ULN2065BN
LEFT 10 5
RIGHT 10 5
Vout
GND
Vin
UCC283T
100uF
ON
OFF
POWER
IRQ
MISO
MOSI
SCK
CSN
CE
BAIL LIFE
MA01-1

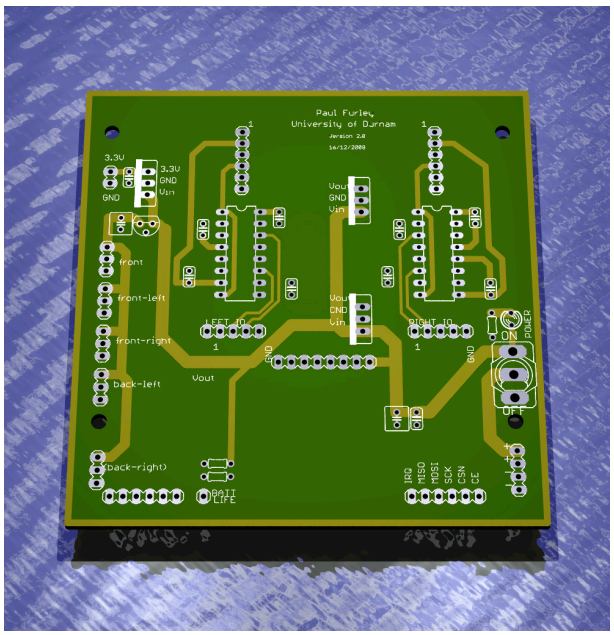


Figure D.1: Top view of a 3D render of the PCB design.

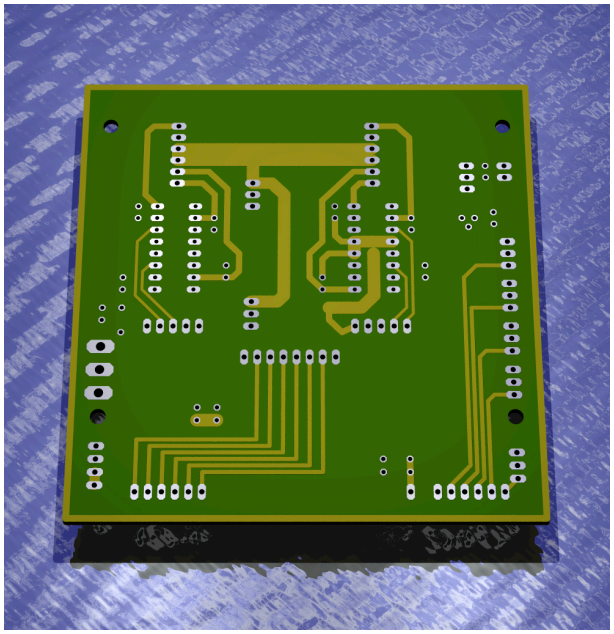
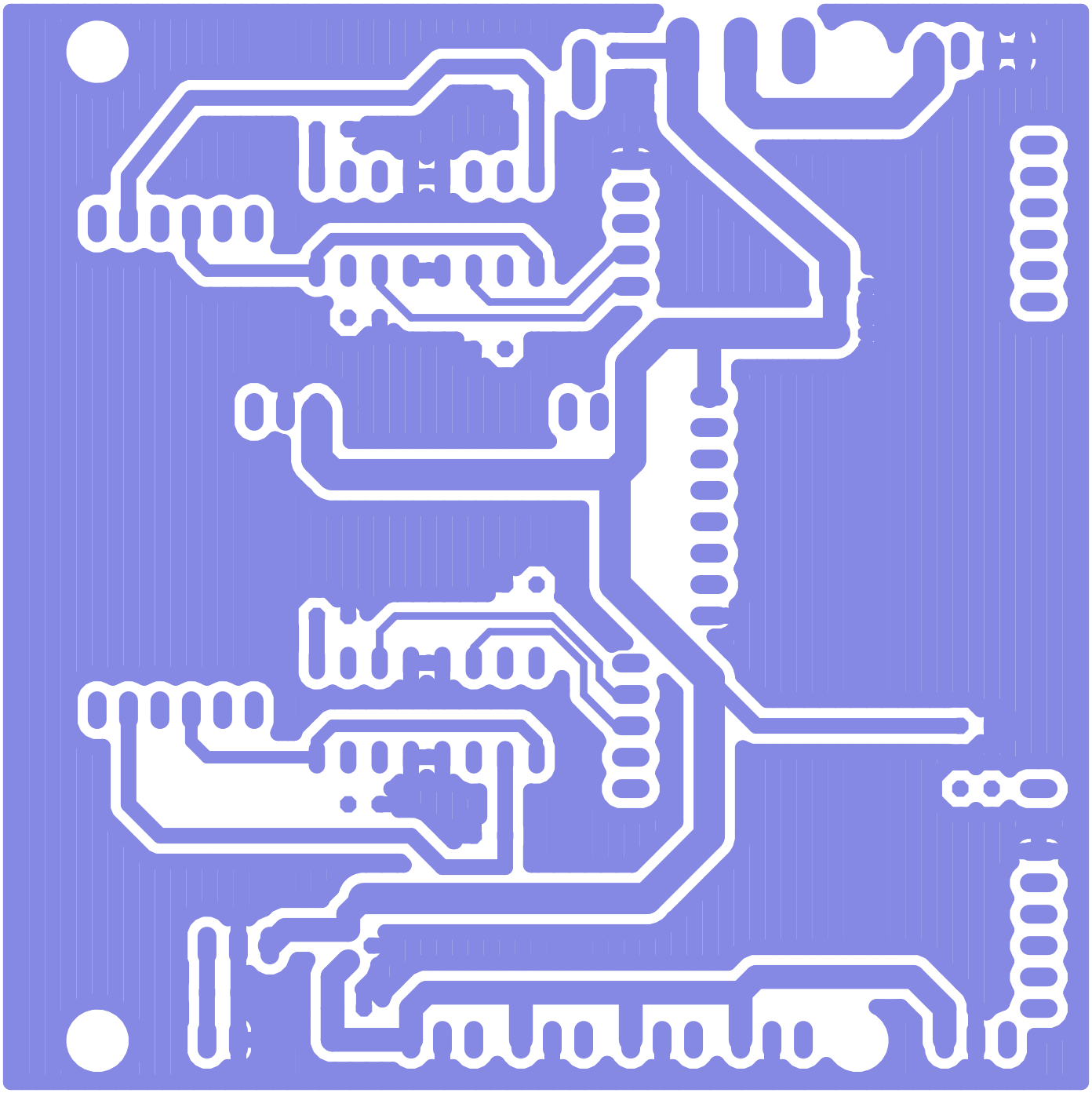
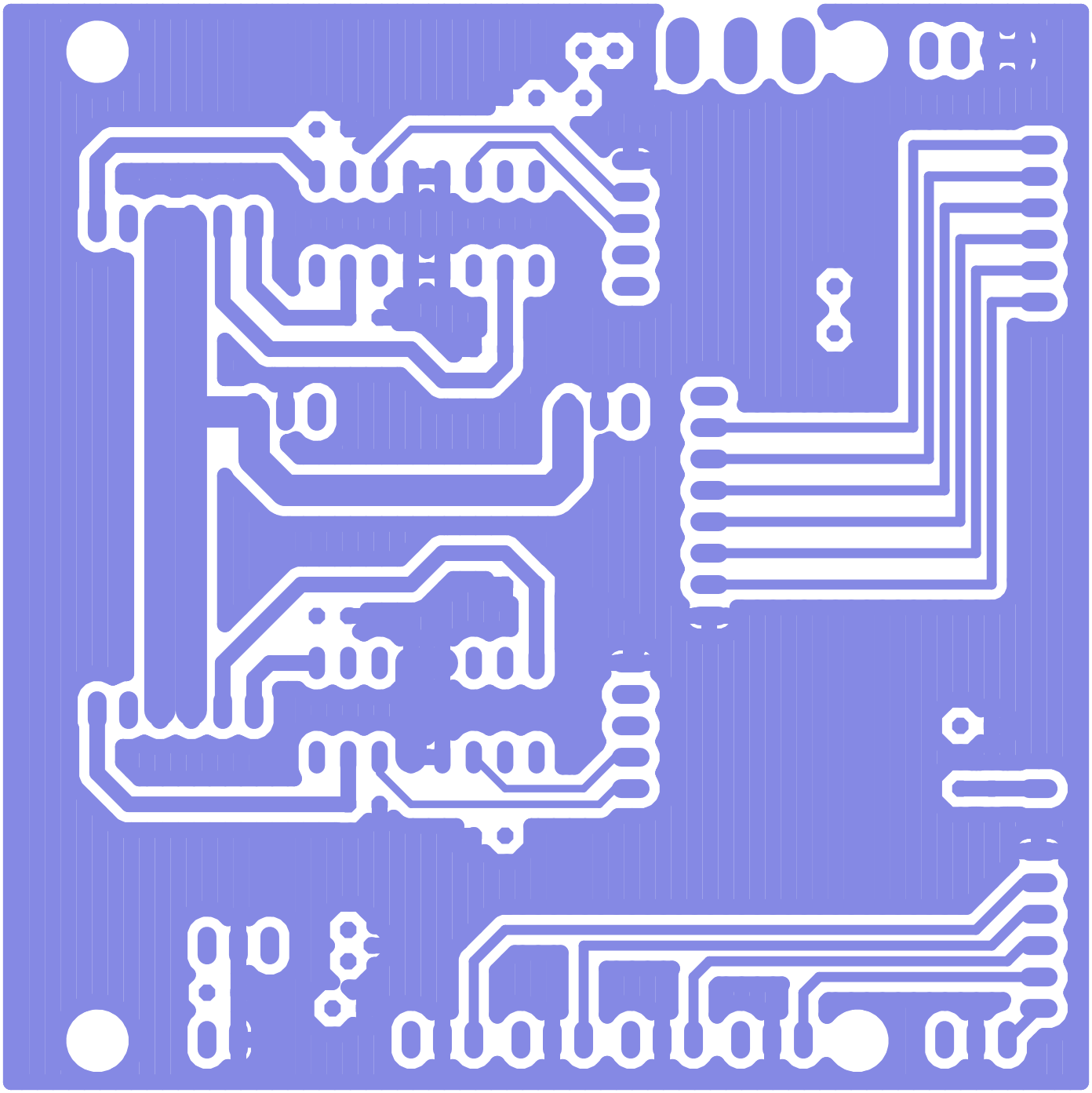
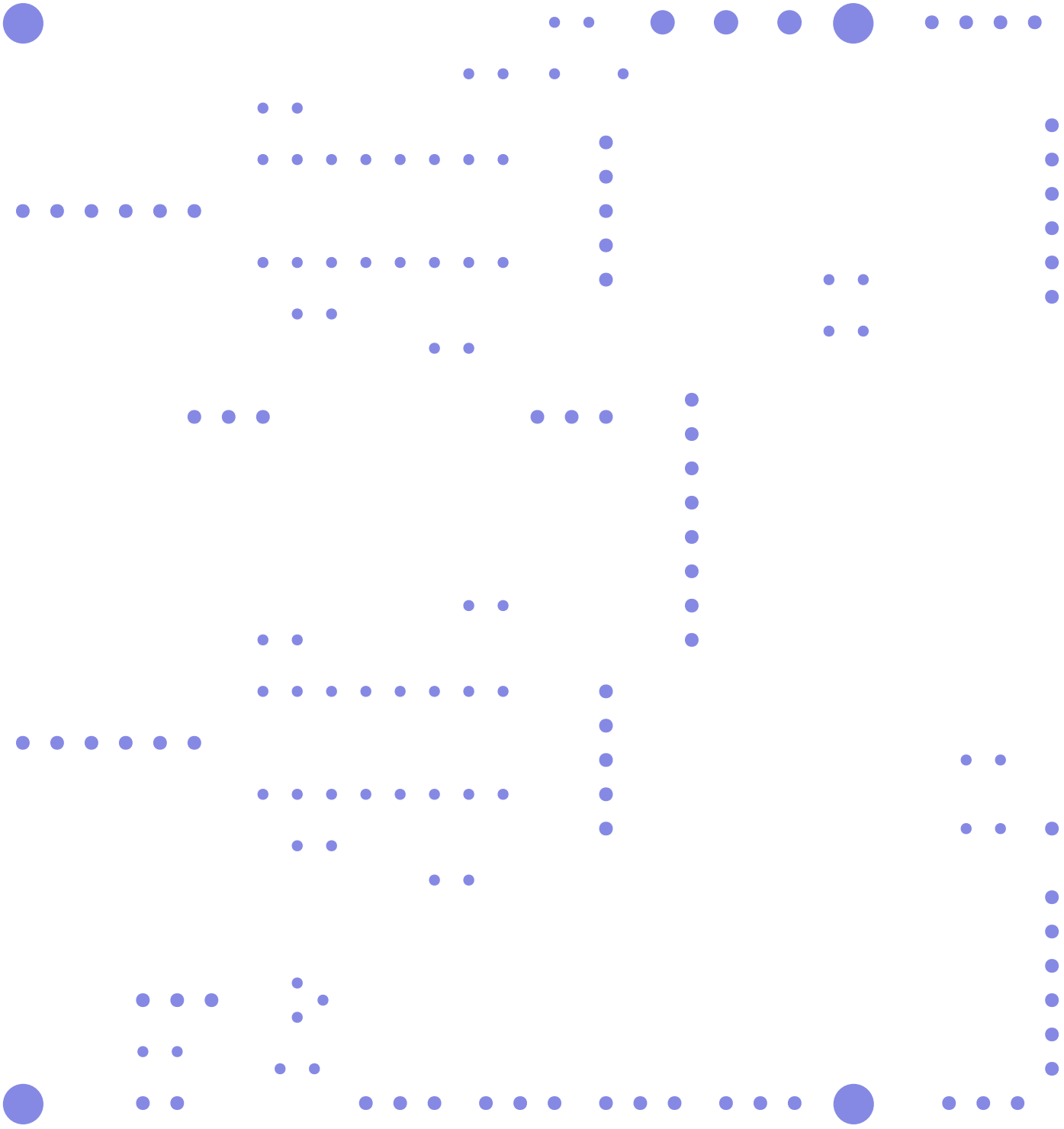


Figure D.2: Bottom view of a 3D render of the PCB design.





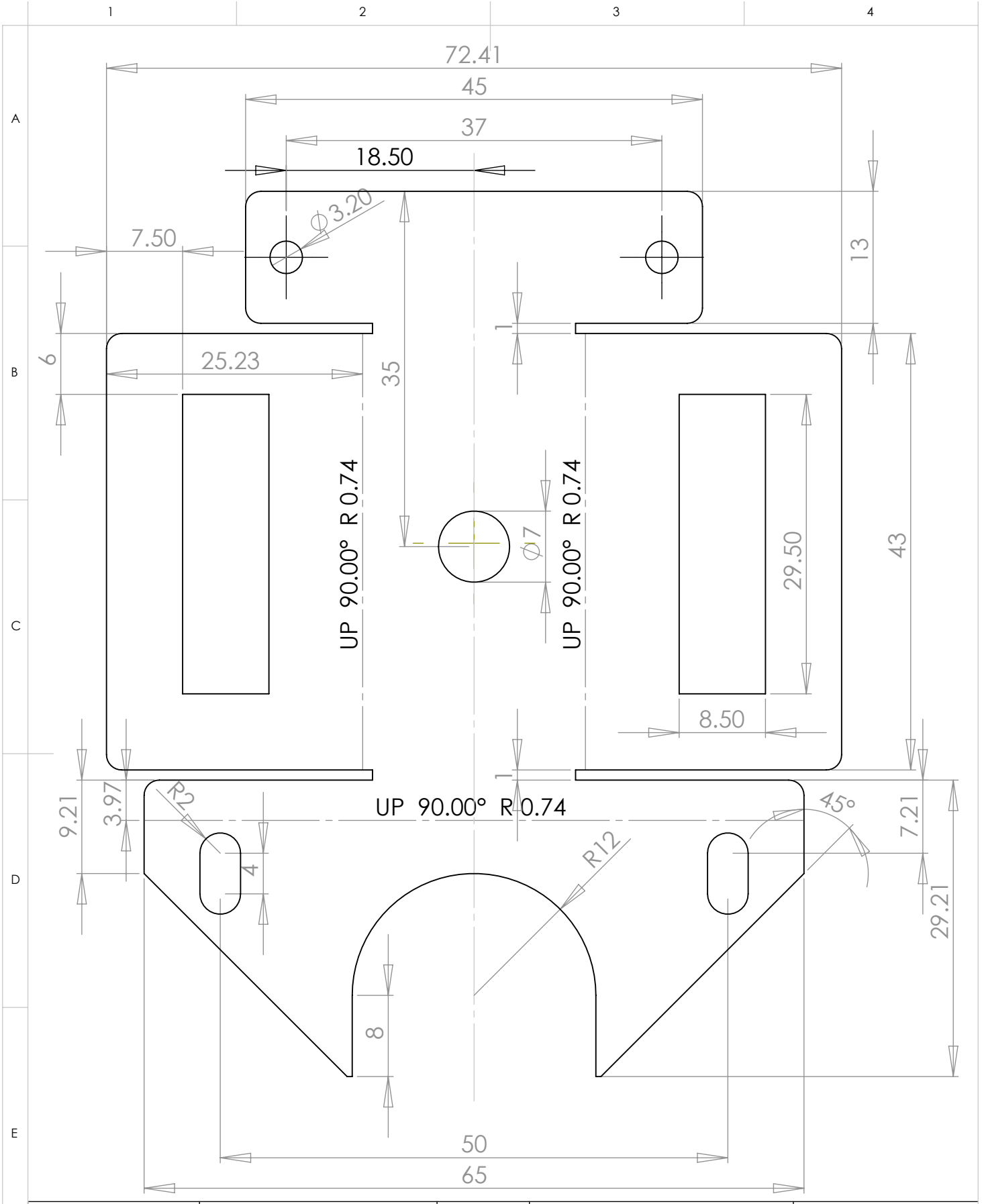


Appendix E

Mechanical Drawings

E.1 Sensor housing

E.2 PCB mounting plate



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:	FINISH:	DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION		

	NAME	SIGNATURE	DATE			
DRAWN						
CHK'D						
APPV'D						
MFG						
Q.A						

TITLE:

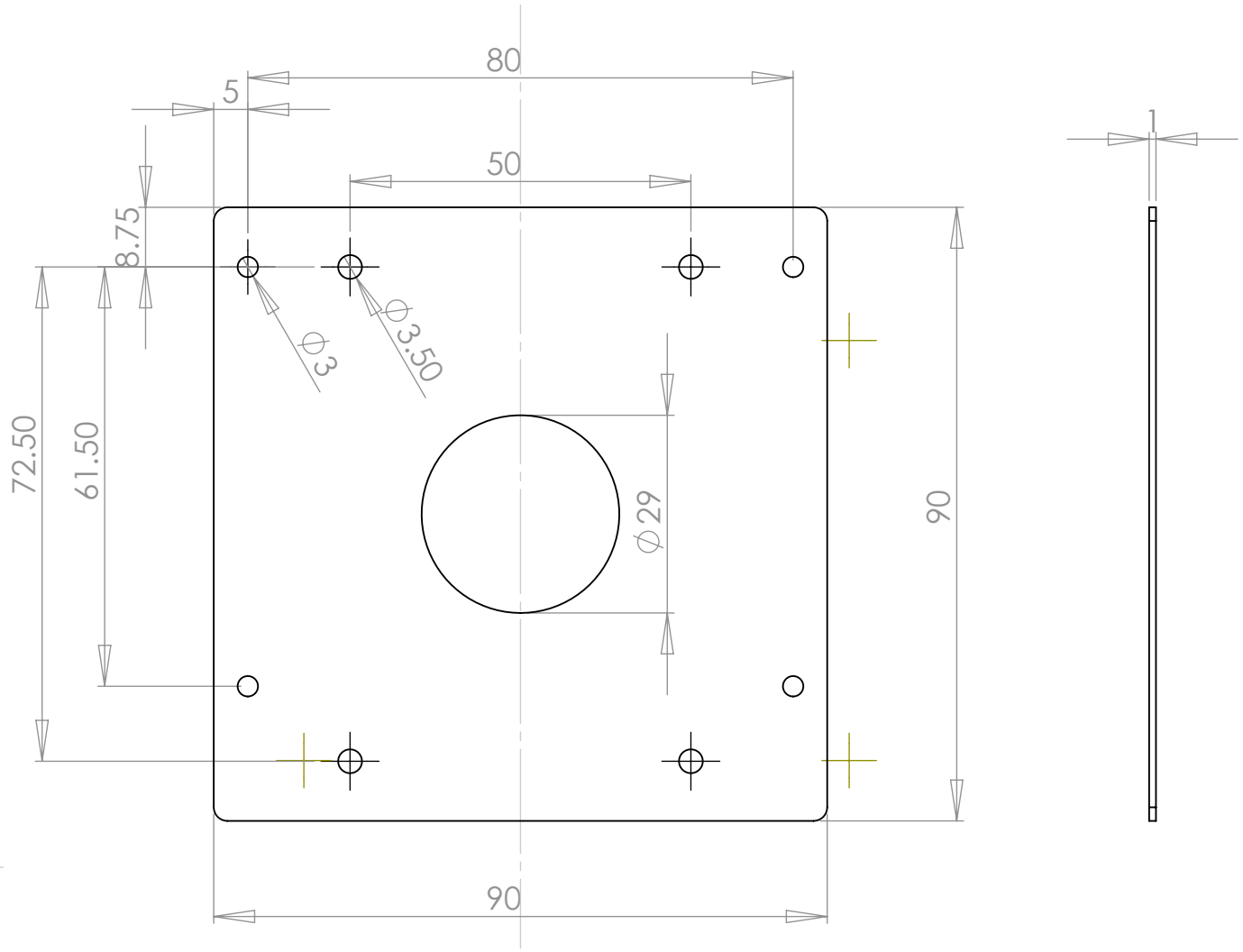
MATERIAL:

DWG NO. Sensor housing MK5 A4

WEIGHT:

SCALE:2:1

SHEET 1 OF 1



UNLESS OTHERWISE SPECIFIED: DIMENSIONS ARE IN MILLIMETERS SURFACE FINISH: TOLERANCES: LINEAR: ANGULAR:		FINISH:		DEBUR AND BREAK SHARP EDGES	DO NOT SCALE DRAWING	REVISION
	NAME	SIGNATURE	DATE			TITLE:
DRAWN						
CHK'D						
APPV'D						
MFG						
Q.A				MATERIAL:	DWG NO. PCB Mount MK2	A4
				WEIGHT:	SCALE:1:1	SHEET 1 OF 1

Appendix F

Software Design

F.1 Command protocol

F.2 Sensor lookup table

Mobile Robot command specification

Base station to Robot commands

	byte 0	byte 1	byte 2	byte 3	byte 4	byte 5	byte 6	byte 7	byte 8	byte 9	byte 10	byte 11	byte 12	byte 13	byte 14
assign id	0x01	token0	[rsrvd]	id	[rsrvd]	i	j	direction							
ping	0x02	id	[rsrvd]												
poll	0x03	id	[rsrvd]												
grant/deny entry	0x04	id	[rsrvd]	i	j	grant/deny									
freeze	0x05	id	[rsrvd]												
set wall status	0x06	id	[rsrvd]	i	j	direc	status								
unfreeze	0x07	id	[rsrvd]												
move to	0x08	id	[rsrvd]	i	j	compass									
step motors	0x09	id	[rsrvd]	L_steps	R_steps										

Robot to Base station commands

request id	0x81	token0	[rsrvd]												
ping reply	0x82	id	[rsrvd]												
report	0x83	id	[rsrvd]	i	j	x_MSB	x_LSB	y_MSB	y_LSB	angle	status	compass	wall_n	wall_s	wall_e
request entry	0x84	id	[rsrvd]	i	j										
freeze ACK	0x85	id	[rsrvd]												
debug	0x21	id	[rsrvd]	str[0]	str[1]	str[2]	str[3]	str[4]	str[5]	str[6]	str[7]	str[8]			
cmd unknown	0xFF	id	[rsrvd]	cmd											
assign id ACK	0x86	id													

byte 15	byte 16	byte 17	byte 18	byte 19	byte 20	byte 21	byte 22	byte 23	byte 24	byte 25	byte 26	byte 27	byte 28	byte 29	byte 30	byte 31
wall_w	right	front	left	b_left	b_right											

8-bit ADC to millimetre lookup table

ADC Value	Distance (mm)
0	255
1	255
2	255
3	255
4	255
5	255
6	255
7	255
8	255
9	255
10	255
11	255
12	255
13	255
14	255
15	255
16	255
17	255
18	255
19	255
20	255
21	255
22	255
23	255
24	255
25	255
26	255
27	255
28	255
29	255
30	255
31	255
32	255
33	255
34	255
35	255
36	255
37	255
38	255
39	255
40	255
41	255
42	255
43	255
44	255
45	250
46	245
47	239
48	234
49	229
50	224
51	219
52	214
53	210
54	206
55	202
56	198
57	194
58	191
59	187
60	184
61	181
62	178
63	175

ADC Value	Distance (mm)
64	172
65	169
66	166
67	164
68	161
69	158
70	156
71	154
72	151
73	149
74	147
75	145
76	143
77	141
78	139
79	137
80	135
81	133
82	132
83	130
84	128
85	127
86	125
87	124
88	122
89	121
90	119
91	118
92	116
93	115
94	114
95	113
96	111
97	110
98	109
99	108
100	107
101	105
102	104
103	103
104	102
105	101
106	100
107	99
108	98
109	97
110	96
111	95
112	94
113	93
114	93
115	92
116	91
117	90
118	89
119	88
120	88
121	87
122	86
123	85
124	85
125	84
126	83
127	83

ADC Value	Distance (mm)
128	82
129	81
130	80
131	80
132	79
133	79
134	78
135	77
136	77
137	76
138	75
139	75
140	74
141	74
142	73
143	73
144	72
145	72
146	71
147	71
148	70
149	70
150	69
151	69
152	68
153	68
154	67
155	67
156	66
157	66
158	65
159	65
160	64
161	64
162	64
163	63
164	63
165	62
166	62
167	62
168	61
169	61
170	60
171	60
172	60
173	59
174	59
175	59
176	58
177	58
178	57
179	57
180	57
181	56
182	56
183	56
184	55
185	55
186	55
187	55
188	54
189	54
190	54
191	53

ADC Value	Distance (mm)
192	53
193	53
194	52
195	52
196	52
197	52
198	51
199	51
200	51
201	50
202	50
203	50
204	50
205	49
206	49
207	49
208	49
209	48
210	48
211	48
212	48
213	47
214	47
215	47
216	47
217	47
218	46
219	46
220	46
221	46
222	45
223	45
224	45
225	45
226	45
227	44
228	44
229	44
230	44
231	43
232	43
233	43
234	43
235	43
236	43
237	42
238	42
239	42
240	42
241	42
242	41
243	41
244	41
245	41
246	41
247	40
248	40
249	40
250	40
251	40
252	40
253	39
254	39
255	39

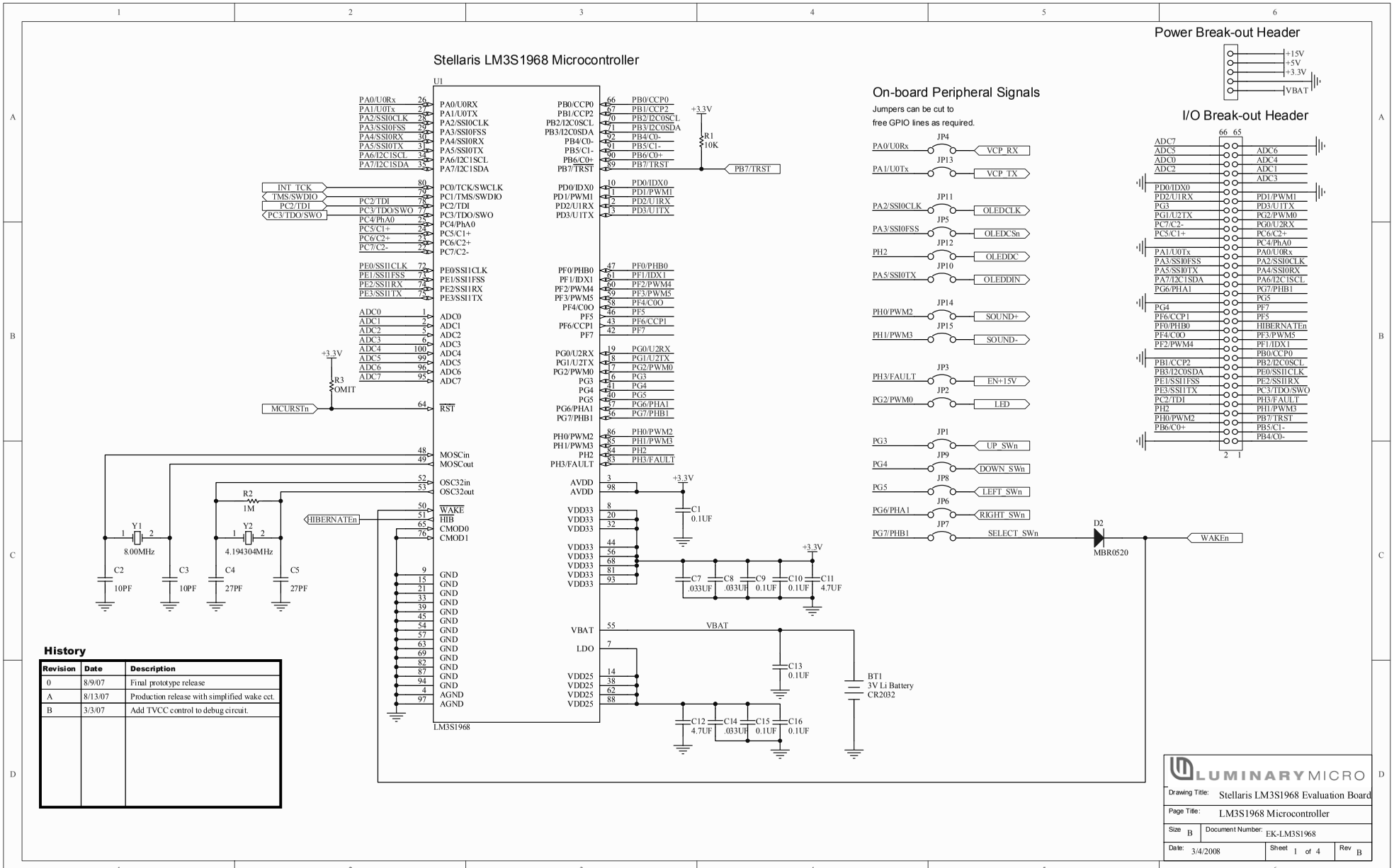
Appendix G

Schematics & Datasheet extracts

G.1 Evaluation board

G.2 Batteries

G.3 Sensors

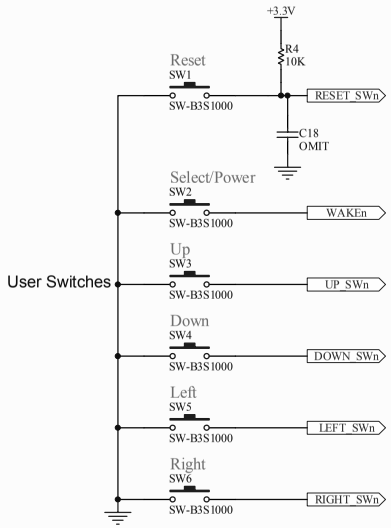


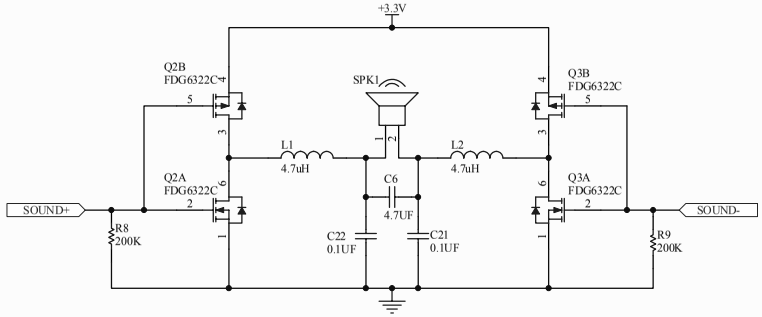
History

Revision	Date	Description
0	8/9/07	Final prototype release
A	8/13/07	Production release with simplified wake cct.
B	3/3/07	Add TVCC control to debug circuit.

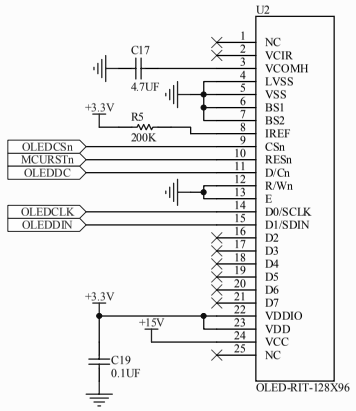
LUMINARY MICRO

Drawing Title:	Stellaris LM3S1968 Evaluation Board	
Page Title:	LM3S1968 Microcontroller	
Size: B	Document Number: EK-LM3S1968	
Date: 3/4/2008	Sheet 1 of 4	Rev B

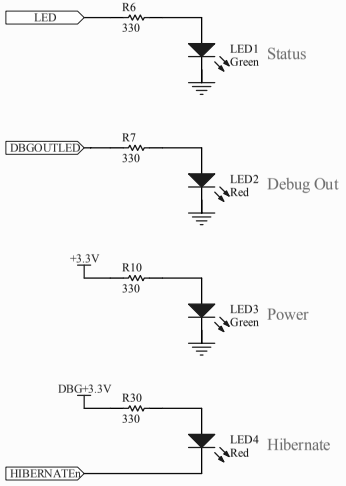




0.2W Audio Amplifier



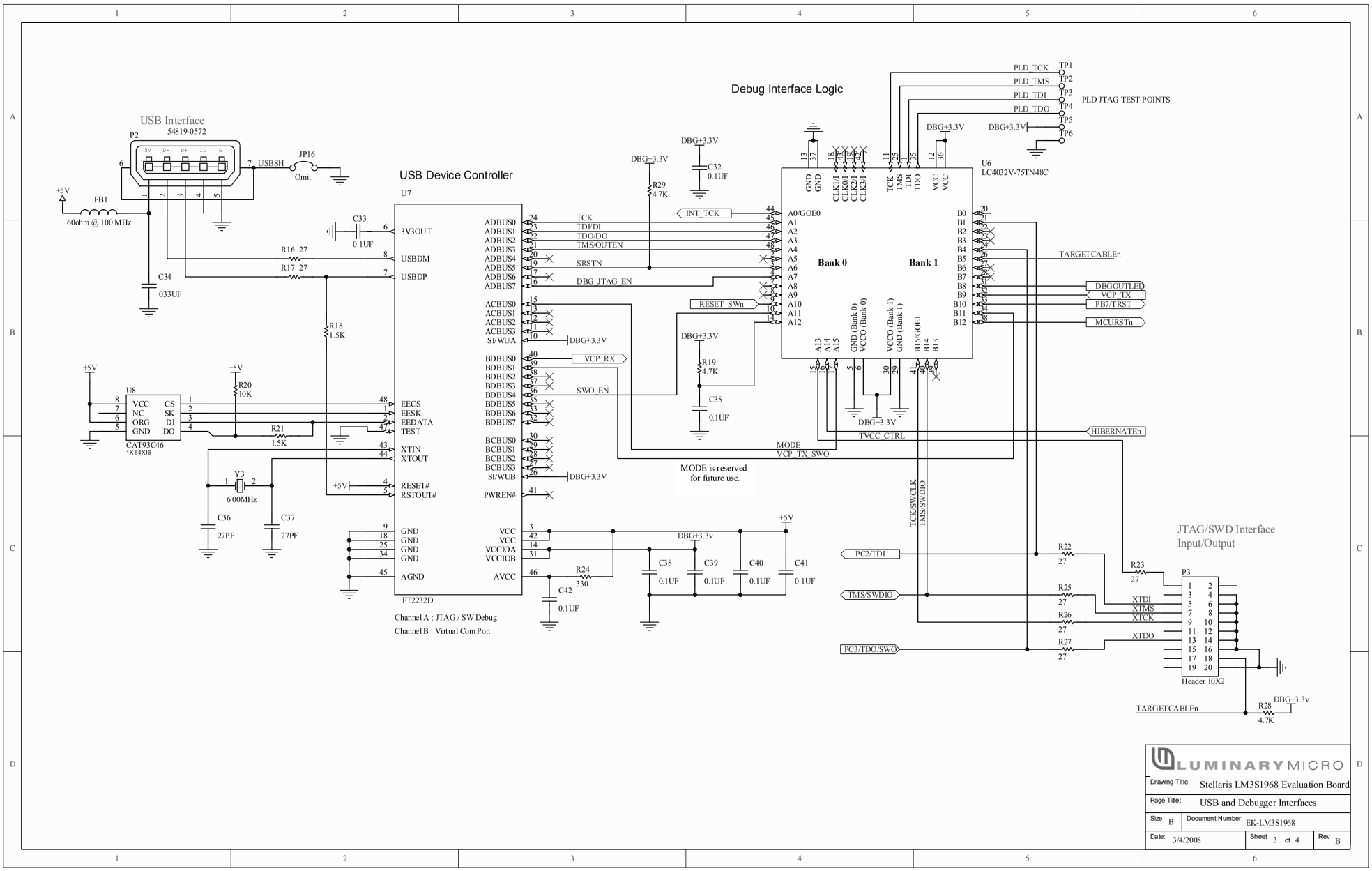
128x96 OLED Graphics Display

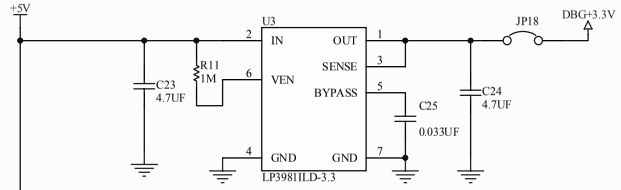


Status LEDs

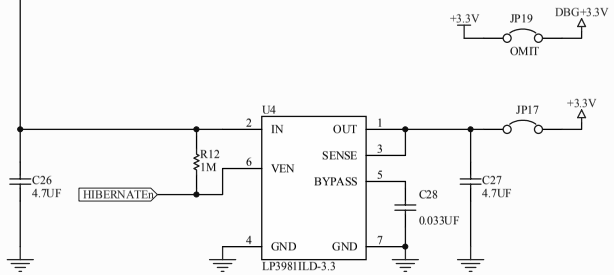
LUMINARY MICRO

Drawing Title:	Stellaris LM3S1968 Evaluation Board	
Page Title:	OLED Display, Switches and Audio	
Size: B	Document Number: EK-LM3S1968	
Date: 3/4/2008	Sheet 2 of 4	Rev B

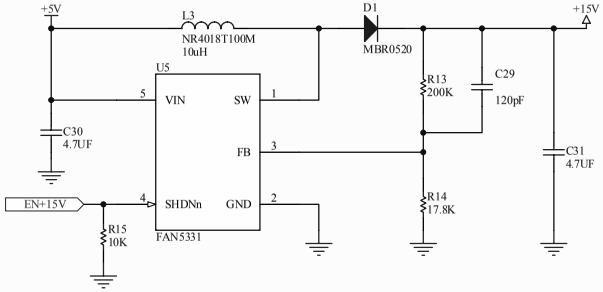

Debug Interface Logic
USB Interface
54819-0572
P2
JP16
Omit
USBSH
FB1
60ohm @ 100 MHz
C33
0.1UF
C34
.033UF
R16 27
R17 27
R18
1.5K
USB Device Controller
U7
3V3OUT
USBDM
USBDP
ADBUS0
ADBUS1
ADBUS2
ADBUS3
ADBUS4
ADBUS5
ADBUS6
ADBUS7
ACBUS0
ACBUS1
ACBUS2
ACBUS3
SI/WUA
BDBUS0
BDBUS1
BDBUS2
BDBUS3
BDBUS4
BDBUS5
BDBUS6
BDBUS7
BCBUS0
BCBUS1
BCBUS2
BCBUS3
SI/WUB
PWREN#
EECS
EESK
EEDATA
TEST
XTIN
XTOUT
RESET#
RSTOUT#
GND
AGND
VCC
VCCIOA
VCCIOB
AVCC
FT2232D
TCK
TDI/DI
TDO/DO
TMS/OUTEN
SRSTN
DBG_JTAG_EN
VCP_RX
SWO_EN
DBG+3.3V
R29
4.7K
C32
0.1UF
INT_TCK
RESET_SWn
R19
4.7K
C35
0.1UF
MODE
VCP_TX_SWO
MODE is reserved
for future use.
U8
CAT93C46
1K 64X16
R20
10K
R21
1.5K
Y3
6.00MHz
C36
27PF
C37
27PF
C38
0.1UF
C39
0.1UF
C40
0.1UF
C41
0.1UF
R24
330
C42
0.1UF
Channel A : JTAG / SW Debug
Channel B : Virtual Com Port
U6
LC4032V-75TN48C
Bank 0
Bank 1
A0/GOE0
A1
A2
A3
A4
A5
A6
A7
A8
A9
A10
A11
A12
B0
B1
B2
B3
B4
B5
B6
B7
B8
B9
B10
B11
B12
GND
CLK1/1
CLK0/1
CLK2/1
CLK3/1
TCK
TMS
TDI
TDO
VCC
GND (Bank 0)
VCCO (Bank 0)
VCCO (Bank 1)
GND (Bank 1)
B15/GOE1
B14
B13
A13
A14
A15
TVCC_CTRL
TCK/SWCLK
TMS/SWDIO
PLD_TCK
PLD_TMS
PLD_TDI
PLD_TDO
TP1
TP2
TP3
TP4
TP5
TP6
PLD JTAG TEST POINTS
TARGETCABLEn
DBGOUTLEI
VCP_TX
PB7/TRST
MCURSTn
HIBERNATEn
PC2/TDI
TMS/SWDIO
PC3/TDO/SWO
R22
R23
R25
R26
R27
27
P3
Header 10X2
XTDI
XTMS
XTCK
XTDO
JTAG/SWD Interface
Input/Output
R28
4.7K
DBG+3.3v
+5V
LUMINARY MICRO
Drawing Title: Stellaris LM3S1968 Evaluation Board
Page Title: USB and Debugger Interfaces
Size B
Document Number: EK-LM3S1968
Date: 3/4/2008
Sheet 3 of 4
Rev B



Debugger +3.3V 200mA Power Supply



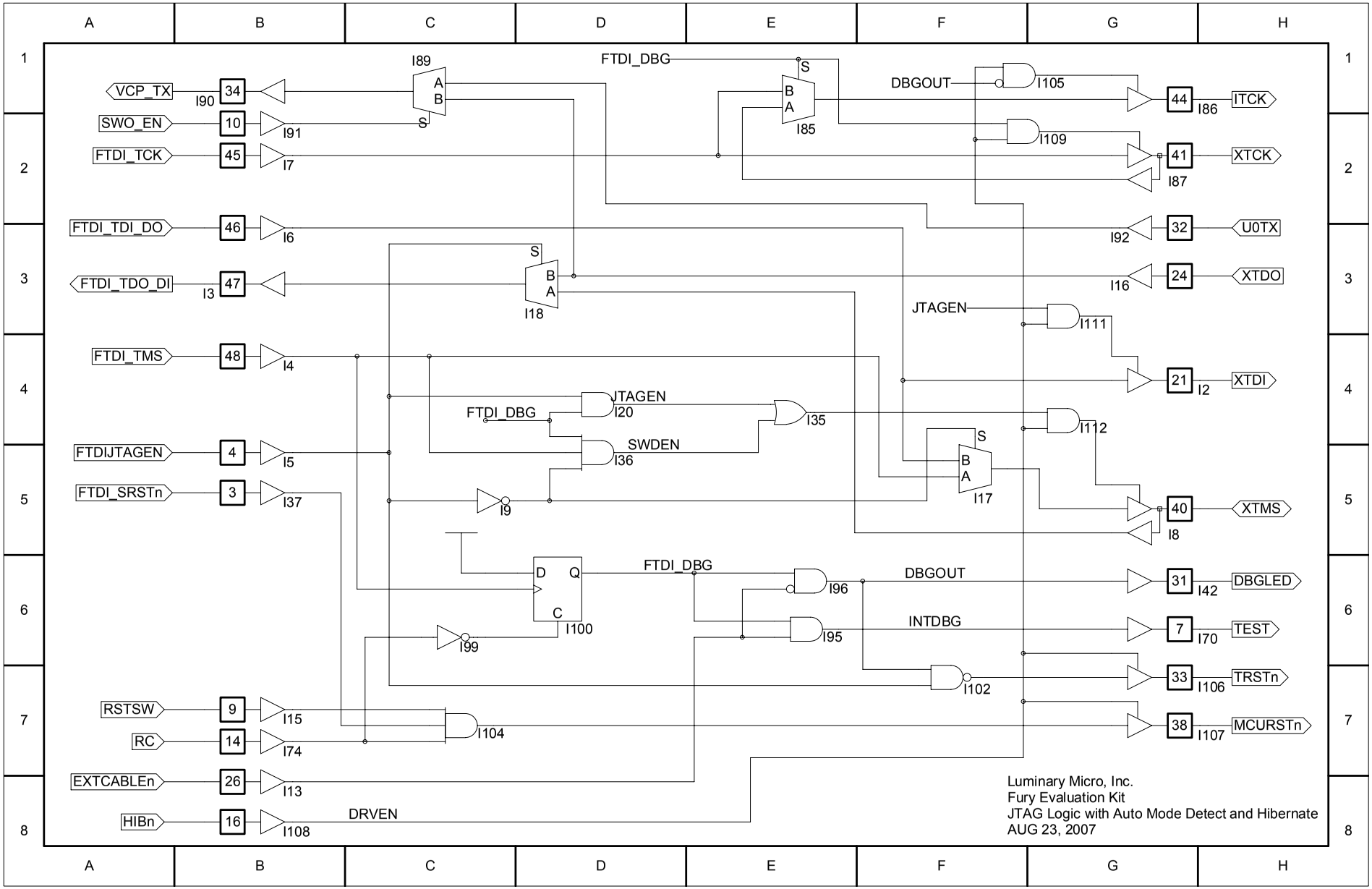
Main +3.3V 300mA Power Supply



+15V 50mA Power Supply for OLED Display

LUMINARY MICRO

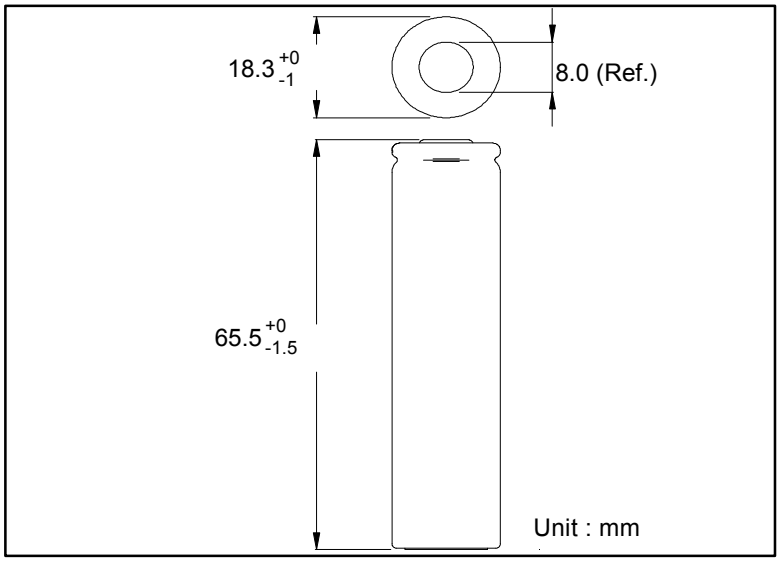
Drawing Title:	Stellaris LM3S1968 Evaluation Board		
Page Title:	USB, Debugger Interfaces and Power		
Size: B	Document Number: EK-LM3S1968		
Date: 3/4/2008	Sheet 4 of 4	Rev B	


VCP_TX
34
I90
SWO_EN
10
I91
FTDI_TCK
45
I7
I89
A
B
S
FTDI_DBG
S
B
A
I85
DBGOUT
I105
I109
44
I86
ITCK
41
XTCK
I87
FTDI_TDI_DO
46
I6
I92
32
U0TX
FTDI_TDO_DI
47
I3
S
B
A
I18
I16
24
XTDO
JTAGEN
I111
FTDI_TMS
48
I4
21
I2
XTDI
FTDI_DBG
JTAGEN
I20
I35
I112
FTDIJTAGEN
4
I5
SWDEN
I36
S
B
A
I17
FTDI_SRSTn
3
I37
I9
40
XTMS
I8
D
Q
C
I100
FTDI_DBG
I96
DBGOUT
31
I42
DBGLED
I99
I95
INTDBG
7
I70
TEST
I102
33
I106
TRSTn
RSTSW
9
I15
RC
14
I74
I104
38
I107
MCURSTn
EXTCABLEn
26
I13
HIBn
16
I108
DRVEN
Luminary Micro, Inc.
Fury Evaluation Kit
JTAG Logic with Auto Mode Detect and Hibernate
AUG 23, 2007

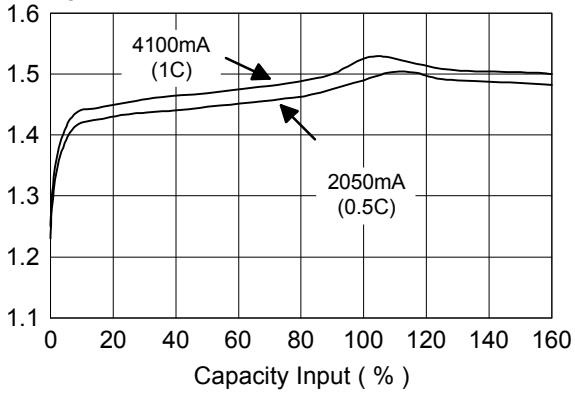
GP Batteries

Data Sheet — *Model No.: GP410LAH*

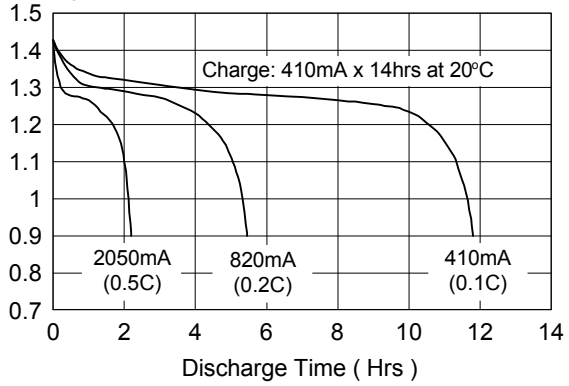
Type	:	Rechargeable Nickel Metal Hydride Cylindrical Cell
Nominal Dimension (with Sleeve)	:	Φ =18.3mm H = 65.5mm
Applications	:	Recommended discharge current 410 to 12300mA
Nominal Voltage	:	1.2V
Nominal Capacity	:	4100mAh when discharge at 820mA to 1.0V at 20°C
Charging Condition	:	410mA for 14hrs at 20°C
Fast Charge	:	2050mA to 4100mA (0.5 to 1C) charge termination control recommended control parameters: -ΔV : 0 - 5mV DT/dt : 0.8°C/min (0.5 to 0.9C) : 0.8 - 1°C/min (1C) TCO : 45 - 50°C Timer : 105% nominal input
Service Life	:	>500 cycles (IEC standard)
Continuous Overcharge	:	410mA maximum current for 1 year No conspicuous deformation and/or leakage
Weight	:	58.0g
Internal Resistance	:	Average 19mΩ upon fully charged (Range 15-25mΩ) at 1000Hz
Max. Charging Voltage	:	1.5V at 410mA charging
Ambient Temperature Range	:	Standard Charging : 0 to 45°C Fast Charging : 10 to 45°C Discharging : -20 to 50°C Storage : -20 to 35°C



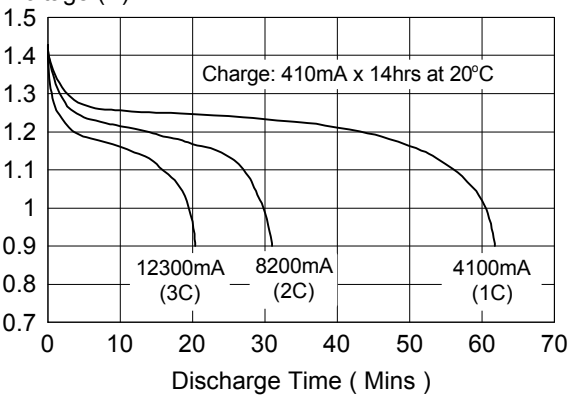
Fast Charge (Charge control required)



Low Rate Discharge



High Rate Discharge



* The information (subject to change without prior notice) contained in this document is for reference only and should not be used as a basis for product guarantee or warranty. For applications other than those described here, please consult your nearest GP Sales and Marketing Office or Distributors.

Member
Gold Peak Group

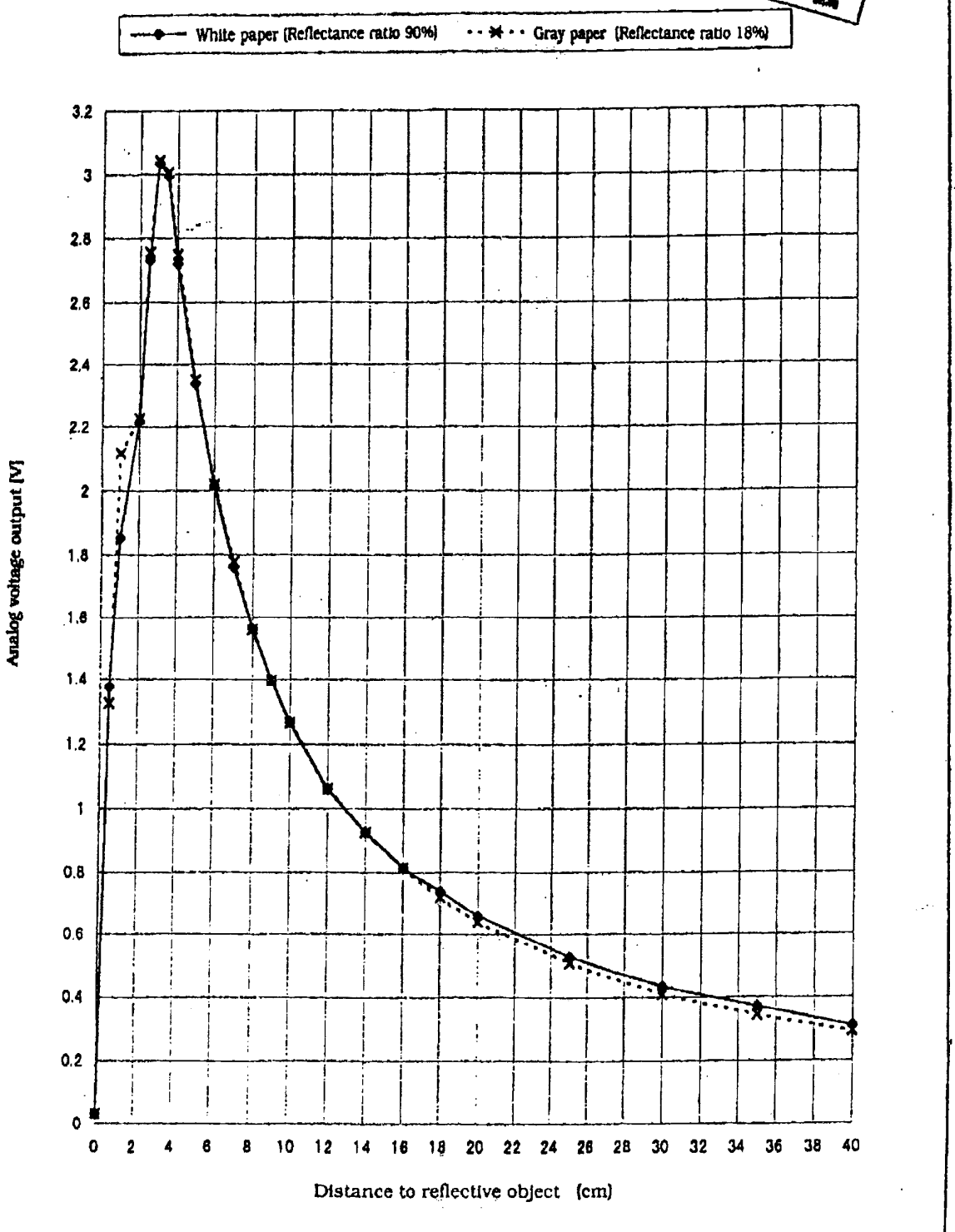
 CRS3479 Rev.00

SHARP CORPORATION

ED-99170	August 30, 1999
MODEL No. GP2D120	PAGE 8/9

REFERENCE

6-1 GP2D120 Example of Output distance characteristics



Appendix H

Software Versions

H.1 Luminary Micro

- CodeSourcery DriverLib-LM3S-1582
- FTDI driver v2.02.04
- LMFlashProgrammer-3.5

H.2 Base station software (python)

- python-2.5.4 (Microsoft Windows)
- pygame-1.8.1releasewin32-py2.5
- pywin32-213.win32-py2.5
- pyserial-2.4.win32
- RealTerm v2.0.0.57
- PortMon-3.0.2.0

H.3 Design software

- SolidWorks 2007
- Eagle Layout Editor 5.3.0 (Microsoft Windows)
- Eagle3D-1.05.27112006
- POV-Ray-3.6 (Microsoft Windows)
- gerbv-2.0.1