# *The Development of a Robotic Urban Search and Rescue System*

A thesis presented in partial fulfilment of the requirements for the degree of

**Masters of Engineering**

In

Mechatronics

At

Massey University,

Palmerston North, New Zealand

**Brendon Rhys Le Comte**

2012

# Abstract

This thesis presents the research, mechanical, electronic, and software design and development of an urban search and rescue system. In the long term this research will help provide the communications infrastructure to allow a team of robots to perform a wide range of tasks in an urban search and rescue operation. These tasks will include search for survivors using small form factors and varied sensors to rapidly and reliably detect people. The tasks are not limited to searching for survivors, as with different sensors the robots will be able to detect unseen hazards, such as gas leaks, and inform rescuers of potential dangers as they occur. These tasks are very dangerous and shifting the work to robots will help minimize the risks to human rescuers and minimise further casualties.

The aim of this research is to develop the communication network by which the robot system will communicate. This network will be an ad hoc network capable of changing in both structure and number of nodes at any point in time. The network relies on the ZigBee protocol and utilises the flexibility and strength inherent in the protocol. The system has been built and tests undertaken to test the range and reliability of the network when acting in this ad hoc manner. The research has also lead to the development of prototypes for two of the robots outlined in the proposed system. These robots have demonstrated the basic functions of the robots and allowed testing to be simpler and easier.

The system has been developed to mirror the proposed urban search and rescue system as much as possible. However, the research lends itself to a huge variety of applications. The overall system is essentially a wireless sensor network and the current work has shown the potential for using a mobile robot to deploy these sensors. This can be leveraged to work in any industry that requires sensor based monitoring to be distributed over large areas.

# Acknowledgements

I would like to acknowledge and thank all the people who have helped me in any way with this research. I would like to express gratitude to some people in particular. Firstly my supervisor, Dr Gourab Sen Gupta, who has guided me through this process and encouraged me since I first expressed interest in postgraduate research. Prof. Dale Carnegie from Victoria University of Wellington was also instrumental in this work. Dale was the mastermind behind the original concept of the search and rescue system. His passion for the field, and his expertise helped shape this research. Thanks must be given to all of the workshop staff at Massey University who have put up with endless questions and requests: Ken Mercer, for his keen eye checking schematics and designs; Clive Bardell, for fabricating mechanical parts whenever asked; Ian Thomas, for answering every mechanical question I could come up with; and Kerry Griffiths, for taking the time to help me with machining the large parts of the research. Lastly I would like to thank my partner, Kezia, for her support and unending patience with me during this long and stressful time.

"You miss 100% of the shots you don't take"

**- Wayne Gretzky**

# Contents

# List of Figures

# List of Tables

# 1. Introduction

Urban Search and Rescue, USAR, refers to the rescue teams who are specifically trained to locate and rescue of survivors trapped in an urban environment. The primary role for a USAR team is the rescue of survivors. This task is incredibly time critical with the chances of survival rapidly decreases after 72 hours [1]. However, to achieve this goal the team will have to enter some of the most unstable, dangerous environments. Urban Search and Rescue in New Zealand is a recently established service. The official New Zealand Urban Search and Rescue group was only founded in 2000 [2] and has only seen service in one major disaster; in Christchurch 2011 [3].

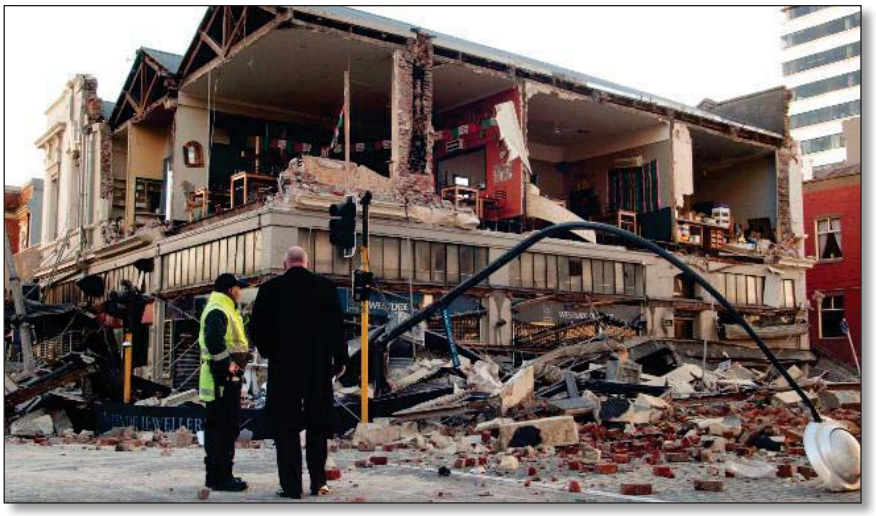

**Figure 1-1 A photo of the damage caused by the Christchurch earthquake of 2011**

In the past ten years there have been numerous catastrophes in which the loss of human life has been staggering. Fatal earthquakes, in the past two years alone, have struck Japan, New Zealand, China, Mexico, Chile, Haiti, with an estimated total loss of life of over 340,000 [4]. The increasing size of cities and population of these cities means that if a disaster were to strike, there would be a need for trained USAR teams to react as soon as possible to try and minimize the loss of life.

The USAR team may face any number of hazards during their work. Ranging from pollutants being released, to explosions, to the complete collapse of the structures they are working in. The hazards are numerous and all potentially fatal; to both the rescuers and the survivors. It is because of these dangers that robots are being developed to aid rescuers and minimize the risks faced in their job. Robots can be utilized in USAR for multiple reasons. Firstly they can navigate terrain and

environments which humans cannot. Small crevasses and tight gaps can be overcome by robots. They ability to enter an environment that a human rescuer cannot provides insight as to the status of the infrastructure of a building. Be that the condition of the buildings supports or checking for flooding or other dangers. This ability allows rescuers to rapidly determine if the environment is safe to begun rescue efforts. Secondly, robots can survive in environments that humans cannot. Gas leaks, radiations, thick smoke, all of these are hazardous to humans, but the right robot can navigate them. For these two reasons robots should be the most important tool in the USAR arsenal, but finally they are advantageous because robots are expendable. The loss of a robot is acceptable, whereas the loss of a human is not. Although the benefits of robots in these situations seems so obvious, the first time a robot was used for search and rescue purposes was the September 11$^{th}$ attacks in 2002 [5].

Research into robots for USAR applications has increased dramatically since September 11. There are hundreds of papers being published over a wide range of topics related to this field. The most numerous are focusing on the mechanical design and the locomotion of the robots. This is possibly one of the most significant challenges faced in the field. Most robots are designed for very specific conditions and USAR is undertaken in some of the worst conditions possible, creating a need for versatile and robust robots. Other research focuses on control and intelligence of the robot, or the mapping of new environments.

This project does not focus on the mechanical side of the problem. The research will focus on the design and prototype development of the overall system and communications. It is part of a greater proposal for a search and rescue robotic system and aims to prove the capabilities of some of the technology.

## 1.1. URBAN SEARCH AND RESCUE ROBOTIC SYSTEM CONCEPT

The broad concept of this research is a low cost, fully autonomous robotic system that can be utilized in a variety of urban search and rescue environments to maximise the detection of survivors and minimise the time taken to successfully rescue them. The system was originally conceived by Dr Dale Carnegie, of Victoria University of Wellington [6]. His vision was for a three tiered system that would be able to rapidly and efficiently search for survivors within the hazardous terrain and environments of a post disaster urban landscape. The system will consist of a large control robot, named Grandmother; medium sized deployable mobile robots, named Mothers; and small

disposable robots, named Daughters. The system itself is designed as a marsupial system, which is analogous to that of marsupials in nature. Marsupial design uses larger robots to carry and distribute a smaller robot [7]. Some of the current marsupial robots only carry one robot at a time [7]. Carnegie's system uses the marsupial functionality, and maximizes the distribution possibilities by loading the Mother robot with a large number of the smaller daughter robots, more like the Ranger and Scouts system [8]. This allows the mother robot to deliver and distribute daughter robots to large area without returning to a central hub. This system provides a very flexible system where daughter robots can be varied by functionality and used where appropriate.

The system that was designed and built during this research consists of three major systems: the Mother, the Daughter, and the User Base Station. Each of these systems consists of a range of sub-systems and components that allow the system to function. This system design is an adaptation of Dr Carnegie's concept and plans to prove some of the functionality of the system so that it can be incorporated into the larger concept.

The main goal of this research was to develop a complete system that will provide a research platform for further research and, most importantly, investigate the viability of a wireless communications channel that consists of a number of wireless relay nodes.

The key idea behind this system is that the Mother robot will be able to carry a number of the Daughter nodes and deploy these as necessary. Each of these daughters will act as a wireless sensor and a wireless relay. By creating a wireless sensor network consisting of these Daughter nodes the system will have the ability to measure specific environmental factors and provide the user with an accurate representation of these factors throughout the operating area. This information will be incredibly valuable to rescuers as it will allow them to make decisions quickly and with the best information possible. This will prevent rescuers entering a dangerous environment or allow them to plan a rescue attempt to avoid further disasters, such as explosions.

This network will also provide the communications channel between the user and the mother robot. This communications channel will provide the ability to send control data to the mother and remotely control the Mother robot. Figure 1-2 shows a general representation of how this network will work.

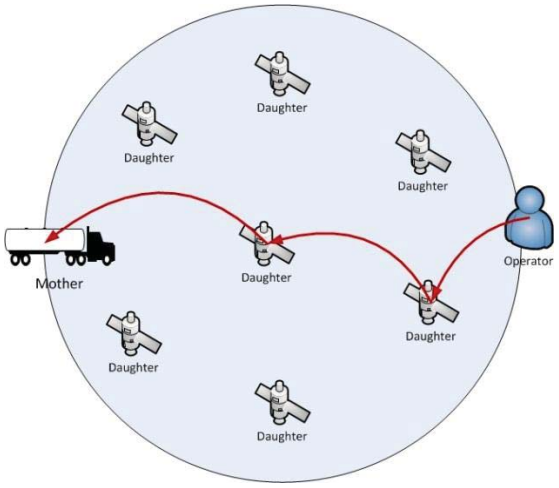


**Figure 1-2 Simplified representation of the operation of the USAR system**

The circle represents the operating area of the system. This area is increased as daughter nodes are added. The Mother has its own operating range, and when close enough to the User the control data is able to be transmitted directly between the two systems. This provides the fastest and most direct form of communications. However, if the mother is out of range of the User, as figure 1-2 shows, the communication between User and Mother can be relayed via the daughter nodes. This concept allows the range of the system to increase, with the limiting factors being the delay of communication or the number of daughters available for deployment to extend the communication range.

## 1.2. RESEARCH GOALS

There are three major research goals to be achieved within this project. The overall goal is to show the operation of the Mother and Daughter robots and how the system can be used

The primary goal of this research is to produce a prototype for the Mother robot. This prototype will prove the concept and viability of the wireless communications link and other important subsystems. The other functionality built in will provide a suitable platform for further research and development. The platform will become the focus of Massey University's research carried out as part of the collaboration with Victoria University of Wellington.

The secondary research goal is to complete the basic functionality of the system. This functionality will include the ability to move around a simple controlled environment as well as provide communications and a basic control platform and user interface. The movement will be controlled by a user operating the robot via a control

interface. The Mother robot will be designed to handle flat surfaces, and safe terrain. The ability to manoeuvre difficult terrain and hazardous environments will be excluded until the research into the systems has been completed. The user interface will be a simply display of the data received from the Mother and Daughter Robots. This user interface will be designed for development and debug purposes. Work to develop a user friendly operating interface will be left till a later time.

The last major research goal is to complete a basic prototype of the Daughter robot. This prototype will provide the wireless relay points for the communications link and show the ability of the Daughter to monitor the environment using sensors. The focus for the Daughter robots sensors will be that of a fire fighting situation. The sensors will be aimed at detecting changes in volatile gas levels and changes in temperature. This has been chosen because it provides an everyday challenge that we can work towards providing a solution for. By achieving this goal we can show that the system is functional and gain funding to expand the system functionality towards other areas.

## 1.3. PROJECT REQUIREMENTS

The requirements for the Mother Robot are as follow:

**Mechanical**

- It shall be a sturdy, durable platform to carry the system components and facilitate further research.
- It shall be simple and cost effective in design, and have a straight forward assembly.

**Functionality**

- The Mother Robot should be capable of driving on a flat surface in a controlled environment.
- It should be wirelessly controlled with the ability to extend the wireless range through the deployment of Daughter robots.
- It should be able to communicate sensor data to the user via the wireless link.
- It should be able to transmit video data to the user.
- It should be able to operate for approximately one hour.

**Dimensions**

- The total size and weight must be within a range that an average adult can carry the Mother Robot.
- The width of the Mother Robot must be less than 610mm. This is the minimum width of a door frame in New Zealand and the robot should be able to manoeuvre through this space.

**Control**

- The Mother should be controlled via a handheld controller or computer interface.
- It shall be simple, easy to use controls. Utilising common control methods (such as joysticks).

The requirements for the Daughter robot are as follow:

**Dimensions**

- The Daughter must be as small as possible to facilitate large numbers being carried on the Mother robot
- It should weigh as little as possible as to not load the Mother robot with excessive weight.

**Functionality**

- The Daughter must be able to sense at least two environmental variables
- It shall be able to determine whether the measured values are within a safe range or increasing at a rate that could become dangerous.
- It shall be able to operate for two or more hours

**Cost**

- The Daughter shall be designed to be as low cost as possible.

## 1.4. THESIS STRUCTURE

Chapter 2 covers the literature review undertaken for this project. It shows research relating to the difficulties and challenges of designing, building, and operating an urban search and rescue robot. There is a review of locomotion types that could be used for the Mother robot discussed in this chapter. There is also a section in this chapter that covers some important existing systems. These systems show either functional urban search and rescue systems, or subsystems or functionality that can be

adapted and used in an urban search and rescue robot. The final part of the literature review is a review of communications protocols and a summary of all the research.

Chapter 3 covers the ZigBee protocol. The operation of the protocol, the function of the protocol and the different modes it can operate in. There is also a review of how this protocol is being used in a wide range of systems within current research.

Chapter 4 covers the concept and design of the Mother robot. This chapter will outline what subsystems the Mother consists of, how they were designed and developed, and the software that controls the Mother.

Chapter 5 covers the Daughter node system. It will describe in detail the system, hardware, and the software that controls it. The Daughter comprises a major component of the system and this chapter will explain the reasons for the current design.

Chapter 6 describes the User Base Station, how it was put together and how it is operated. This chapter will focus on the functionality and the user interface, explaining the choices made that lead to the development of the current interface.

Chapter 7 covers the network that is the heart of the system. The network allows the user to control the Mother robot and gather environmental data from the Daughters. This chapter will detail the functions of the system as well as show the results from testing done as part of this research.

Chapter 8 details the conclusions drawn from this work. It will explain the successes and failures that arose and outline how these will influence the future work that needs to be completed to make this system functional in any real world situation

During the course of this research, 3 papers have been published. The first at the 5$^{th}$ International conference on Automation, Robotics, and Applications 2011 in Wellington, New Zealand The second at the Electronics New Zealand conference 2011 in Palmerston North, New Zealand and the third at the 2012 IEEE International Instrumentation and Measurement Technology Conference in Graz, Austria. The paper abstracts follow:

[9] B. Le Comte and G. Sen Gupta, “Characterisation of a Motor Driver with Over-Current Protection and Speed Feedback,” in *5th International Conference on Automation, Robotics, and Applications*, 2011, pg. 127 - 132.

**Abstract:**

Urban search and rescue robots are becoming more prolific and widely used. However, these robots are still being developed for specific environments and situations. The research being undertaken at Massey University is part of a larger collaboration to produce a multi-tiered robotic system for urban search and rescue. The first major hurdle of a mobile robotic system is mobility. While this research will ignore the difficulty of hazardous terrain, stable and controllable manoeuvrability is still required. To achieve this, a motor controller system has been designed and developed. Each controller will control a wheel of the robot and maintain a speed determined by a central computer. The motor controller requires the ability to control the motor speed and check that the current drawn is not too high as to damage the motors or other hardware. This paper outlines the structure of this motor controller system as well as the testing undertaken to determine the PID constant for the control loop. The testing has resulted in a set of PID constants that control the system, within pre-defined metrics, very effectively.

[10] B. Le Comte and G. Sen Gupta, “Two-Tier Wireless Robotic System for Urban Search and Rescue,” in Electronics New Zealand Conference, 2011, pp. 8 - 12.

**Abstract:**

Urban search and rescue is a dangerous task. However, it is incredibly important in a world where the cities are getting larger. Robots can be designed and developed to help minimize the risks associated with this task as well as improving the likelihood of success for any urban search and rescue operation. Massey University is currently working on a prototype system that will prove the concept of a tether-less system that utilizes smaller deployable robots for maintaining a communications link. This prototype will eventually be incorporated into the three-tiered robotic system being developed by Victoria University of Wellington. This paper outlines the overall research goals at Massey University. The structure of the overall system and a description of how each section will work are covered. A description of both function and architecture is covered for both tiers of the system as well as specific implementation of the lower tier robot.

[11] B. Le Comte and G. Sen Gupta, “Distributed Sensors for Hazard Detection in an Urban Search and Rescue Operation,” in IEEE International Instrumentation and Measurement Technology Conference, 2012, pg. 2385 - 2390.

**Abstract:**

Urban search and rescue is a dangerous and difficult task. The amount of information required to mount a rescue is staggering. This coupled with the time constraints of a rescue operation, mean that decisions are sometimes made without all the facts. Gas leaks or structural weaknesses put everyone in danger and can be difficult to detect without being in the environment itself. Robots can penetrate these disaster zones much more easily than humans or dogs and can detect unseen hazards like gas or radiation or can relay images of structural supports. By using robots we can quickly increase the amount of information available to first responders and help them make informed decisions that see more lives saved. The research at Massey University aims to build a system that will provide, amongst other features, a distributed sensor network that can monitor environmental factors and report them to the rescue team. This paper outlines the concept of these sensors, or Daughters, and the results of the tests on the sensors implemented.

# 2. Literature Review

Robots are being used in Search and Rescue applications all over the world. The field has been rapidly growing in the past decade. Robots have seen use in the aftermath of the September 11 attacks [5], [12], the devastation of the Christchurch quakes [13], and the Pike River mine tragedy [14]. Robots are being more widely applied to rescue efforts and post disaster activities. This increase in use has led to a drastic increase in the types and functionality available in existing systems.

While the majority of research still appears focused on solving the mobility issue, there is still a lot of work being done to solve the other obstacles involved with this type of work. This work has produced a wide range of robots and robotic systems that can be applied to the field.

## 2.1. Major Research Challenges

Robotics for urban search and rescue has challenges that make it interesting from a research point of view. These challenges also make it a very difficult field commercially. The issues which USAR robots face are so numerous that the development of robots to do this task is so expensive that they are not commercially or practically viable. Issues such as the incredible variation of terrain that the robots need to navigate have created problems that are not easily overcome. The solutions to the challenges are also what make an urban search and rescue robot unique when compared to more generic systems.

Gurvinder Virk believes that the major issues holding back this field are Locomotion, Mapping, and Human machine interaction [15]. Another point of issue is the communications systems used. The communication systems used dictates how the robot can be used and in what situations it can be deployed. The communications generally manifest as tethered or untethered systems, which provide other advantages and disadvantages.

### 2.1.1. Locomotion

The issue of locomotion is an obvious one. Locomotion in an urban landscape after a natural or man-made disaster is incredibly varied. The terrain can vary in size, from large chunks of concrete, to dust. Leaks from water and sewage pipes add extra difficulty by making surfaces slippery and reduce traction. This environment is not just difficult to maintain traction and move across, but the non-uniform nature of the

environment means that a robots size may stop it from penetrating further. These problems are only exasperated by the fact that the environment can continue to change during a rescue operation, from issues such as building collapse or aftershocks etc. To help prepare researchers for these difficult environments, USAR robotics competitions have been started that utilise simulated rubble and situations to test and evaluate systems [16]. Research is vast and varied in this specific area, from robots that use manipulators to move themselves [17], to systems that utilise a continuous membrane [18], to more common forms of locomotion like tracks or wheels [19]. Reviews have been presented both on robot locomotion [20] and on locomotion for search and rescue specifically [21] but still the research community cannot build a robot that can conquer all environments. Locomotion, mechanical design in the broader sense, will be a limiting factor in research and prove to be a major challenge at developing a robotic system that can handle any and all terrain.

### 2.1.2. Mapping and Localisation

The issue of mapping and localisation are very important to this field. Mapping and localisation refers to the generation of a complete map of the environment and the ability to state the position of the robot relative to a landmark or the user. This is incredibly important as it will allow the rescuers to see the environment and create a rescue plan that is the safest possible. The mapping gives the rescuers all the details of size, and placement of obstacles. From this map a trained professional will be able to see what objects are holding the structure up and where potential weak points are. This information is incredibly valuable in protecting the rescuers and not causing any further harm to the victims. This information, however, is useless unless the rescue team knows where it is relative to themselves or a landmark. Having all of this information but no idea of where it represents means the rescue team will have to blindly search to find the area. This ‘stab in the dark’ approach is dangerous as they can inadvertently cause more harm than good. Research in this field has utilised a wide range of sensors for detecting the environment. Cameras are widely used, with research in single camera [22] and with multiple cameras [23]. Laser range finders have also been used in this field [24] as have a wide range of other sensors. Research also focuses on the algorithms required to make this intensive mapping computationally viable and work in real time [24] [25]. The advent of cheaper, more powerful processors will help make this field of research easier to incorporate into a mobile robot.

### 2.1.3. HUMAN MACHINE INTERACTION

Human machine interaction is the final major obstacle. This is only considered an obstacle as artificial intelligence is still too immature to allow robots to autonomously co-exist with humans with reliable safety and task performance [15]. Because we cannot rely on the robots to operate autonomously there is the requirement that a human operator make decisions and interpret data. To do this effectively, the first requirement is an interface that can be easily understood and operated. The interface must show all the relevant data, while maintaining a clear and concise layout. Because an USAR robot will be monitoring so many variables, both internal and external to the robot, there is a huge amount of data to consider. However, Human Machine Interaction does not stop at the screen. In the case of systems which are teleoperated, such as the proposed system, the user must control the robot through an interface. Much research is being done into the types of interfaces that can be used. Some researchers are attempting to use current generation gaming technologies, such as the Xbox 360's Kinect, to provide a video interface [26].This research may not be directly aimed at robotics but it has applications in the control of robots. Other researchers, primarily in bionics research, have been working to pioneer brain-computer interfaces [27]. This research provides people with disabilities to control computers using only their thoughts. Recent research has even show a women controlling a robotic arm with one of these devices [28]. While research into these exciting interfaces continues there more common interfaces are still very widely used. An example of a more traditional interface is a joystick. The da Vinci surgical system utilises physical operated controls, much like a joystick, to perform minimally invasive surgeries, sometimes from remote locations [29]. While the research continues to push the bounds of human machine interaction, it still focuses on a case by case basis. The proposed system will require a simple to use, easy to move, interface system. It must work is hazardous, cluttered environments and be reliable in these conditions.

### 2.1.4. COMMUNICATIONS

Communications used can be categorised into two major groups: Wired and Wireless. Each category has advantages and disadvantages. Wired communications or a tethered robot provides the ability to make lighter robots, as there is no requirement for on board power. This is beneficial as in unstructured environments the weight of the robot can cause slippage of rubble, or in an extreme case the collapse of the

structure. A tethered robot also provides a reliable communications channel, with large bandwidth. This allows the robot to transmit video and data very easily. The tether provides a life line by which the robot can be retrieved. Robin Murphy, an expert in the field of robots for urban search and rescue, is a major supporter of tethered systems [30].

Wireless communications, or tether less systems, have the advantage as they are not restricted by a cable. This means they can enter environments without the fear of a tether becoming caught on rubble. While tethered robots have a reliable communications, they also have the disadvantage of the tether is also a vulnerability for the robot. If the tether is damaged or destroyed the robot is lost. With wireless system the communications the robot is only restricted by the connection to the user.

From an implementation point of view, tethered is the easiest and most reliable system for an urban search and rescue robot. However, a tether less system would provide far better functionality for the application. There is division amongst researchers over the best type of communications for these robots.

### *A. Wi-Fi*

Wi-Fi is a common wireless communications system. It is utilised within most modern day laptops and most modern smart phones. This pervasive communications protocol is defined by an array of standards under the heading of 802.11. There are currently four commercially available Wi-Fi technologies under the 802.11 heading; these are listed in table 2-1

| Wi-Fi Technology | Frequency Band | Bandwidth or maximum data rate |
|---|---|---|
| **802.11a** | 5 GHz | 54 Mbps |
| **802.11b** | 2.4 GHz | 11 Mbps |
| **802.11g** | 2.4 GHz | 54 Mbps |
| **802.11n** | 2.4 GHz, 5 GHz, 2.4 or 5 GHz (selectable), or 2.4 and 5 GHz (concurrent) | 450 Mbps |

**Table 2-1. Wi-Fi technologies under the 802.11 specification [31]**

Wi-Fi technology makes use of the 2.4GHz to 5GHz frequency spectrum for operation and can operate over long distances, up to a few hundred meters. The

technology was devised as a means for communications for mobile computers but has proliferated and is common in a vast array of consumer devices.

### *B. Bluetooth*

Bluetooth was devised as a short range wireless replacement for serial communications. It would allow devices to communicate using the same software but wirelessly. It operates, similar to Wi-Fi, in the 2.GHz range. However, Bluetooth only extends as high as 2.485GHz. Bluetooth has 3 classes of transmitters [32] . These are broken down based on the range of the transmitter. Table 2-2 shows the 3 classes and there expected ranges.

| Class | Range |
|---|---|
| Class 3 Radios | have a range of up to 1 meter or 3 feet |
| Class 2 Radios | most commonly found in mobile devices – have a range of 10 meters or 33 feet |
| Class 1 Radios | used primarily in industrial use cases – have a range of 100 meters or 300 feet |

**Table 2-2. Bluetooth Radio Classes [32]**

Bluetooth was also intended as a low power method of wireless communication, with each subsequent generation striving to provide a lower power alternative to the previous. For example, version 3 consumes anywhere between ½ and 1/100 the power of the original Bluetooth technology [32]

### *C. ZigBee*

ZigBee is a wireless technology designed, much like Bluetooth, for short range wireless communications. The ZigBee technology is being developed by a group of technology manufacturers [33]. ZigBee is a low power wireless communications system intended for use in home and building automation. Due to its goal it is required to provide reliable communications up to 100m in range. A ZigBee module operates in the 2.4GHz band and adheres to the IEEE 802.15.4 specification. The ZigBee protocol includes many features such as 16 channels of operation, pairing of devices, different transmission types, AES-128 security, and more [33].

### 2.1.5. Design Considerations Based on These Challenges

The combination of these four groups of design and development issues poses a formidable challenge for any engineer engaged in this field of research. The requirement for a system that is not only mechanically robust but also has the highest levels of reliability from a system standpoint is a challenge that cannot be taken lightly. However, due to the broad scope of these challenges it is difficult for a small team, or individual, to address all of the aforementioned issues within the scope of a single project. The research outlined in this thesis focusses primarily on the communications issue. A robotic system that has limited locomotion, poor mapping, or inadequate interfacing systems can still be used in some capacity as long as the communication is functioning. In search and rescue, even an autonomous system, would be rendered functionless by the loss of communication. This is due to the system being unable to alert the rescue team to anything. A system could implement an audible noise as a backup, but in a post disaster area noises are likely to go unheard over the ambient noise. With this in mind the scope of the project was defined to focus on developing a reliable and adaptable communications infrastructure that works in a wide range of structured and unstructured environments.

## 2.2. Existing Systems

### 2.2.1. iRobot Packbot and Warrior

iRobot is a large commercial robotics company based in Massachusetts, USA. iRobot develops a range of robots for both commercial and industrial/military uses. The most famous of which is the Roomba robotic vacuum cleaner. iRobot also created the Packbot and Warrior robots. These robots provide a tracked solution to mobility with the added bonus of controllable tracked arms, as seen in figure 2-1. These arms afford the robots greater ability to navigate obstacles, such as stairs or rubble. By raising or lowering the arms the robot can gain traction or stability on slopes.

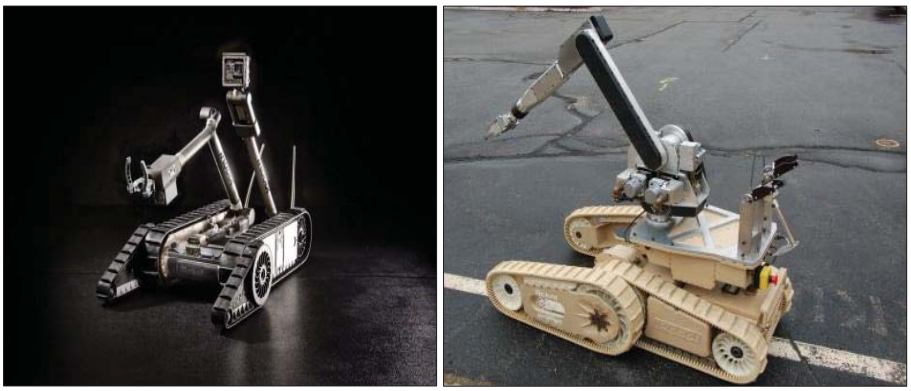

**Figure 2-1 iRobot Packbot (Left), iRobot Warrior (Right) [34], [35]**

This solution is very effective at climbing stairs [34], [35] and has been adopted by a large number of robots. The tracked design allows the robot to travel over loose surfaces, such as gravel, with relative ease. These robots allow carry a robotic arm that allows the user to interact with the environment. According to the iRobot specifications the Packbot can lift up to 30 pounds with its manipulator [34]. The end effector force is even transmitted to the user allowing them to determine whether or not an object has been grasped. These robots can be purchased commercially and can be configured in a variety of ways depending on the application. In terms of USAR, these robots have been deployed in disaster situation. These robots are being used in the clean up after the Japan earthquake and the ensuing disaster at the Fukushima Dai-ichi Nuclear Plant. These robots have been used as robotic fire fighters, pulling hoses into hot zones inside the nuclear plant to help direct the flow of cooling water [36].

### 2.2.2. GEMINI-SCOUT MINE RESCUE ROBOT

In recent years there have been numerous mining disasters. Chile, America, China, and even the Pike river tragedy here in New Zealand have reminded the world of how dangerous mining operations can be. Saving the lives of trapped miners, while not an USAR operation, is very important. The challenges of terrain, environment and access are as important in this area as they are in USAR. The Gemini Robot was developed by Sandia National Laboratories [15]. The robot is water proof up to 18 inches [16] and electrically sealed, so not to cause explosions from methane commonly found in mines. The robot has been designed specifically for the challenges of a mine related disaster, but as the project manager Jon Salton has said that the “Gemini-Scout could easily be fitted to handle earthquake and fire scenarios, and we think this could provide real relief in currently inaccessible situations” [16].

**Figure 2-2 The Gemini-Scout mine rescue robot [19]**

### 2.2.3. QUINCE

**Figure 2-3 Quince robot shown in a testing environment [37]**

Quince is a robot developed by the Chiba Institute of Technology and Tohoku University. Like many of the other robots described Quince utilises tracks for locomotion. Quince is very similar to the design of the Packbot from iRobot, however, two main distinctions can be made between the two. The first is that quince lacks a manipulator. This means Quince cannot actively interact with the environment in the same capacity. Secondly, Quince has front and rear tracked arms. These arms are independently controlled and give Quince 6 degrees of freedom [37]. The arms also help the robot maintain stability and can be used to extend the total length of the system to bridge wider gaps [37]. Quince also provides an array of sensing technologies. Quince also has a 3-D scanner system, called a TK scanner, that utilises a 2-D laser range finder on a pan-tilt platform [38]. This TK Scanner allows Quince the ability to map the physical environment.

### 2.2.4. ACTIVE SCOPE

**Figure 2-4 Active Scope and operator (Left), Close up on the end of the Active Scope (Right)** [39]

Active scope represents a very different approach to robotics within search and rescue. This system aims to provide the search and rescue team or inspection teams the ability to enter the smallest possible holes and view them for survivors or damage. This approach shows the ability of technology to greatly improve the amount of information available to workers in a post disaster situation. It utilises ciliary vibration as its drive mechanism. This concept uses small inclined rods, or cilia, that are vibrated causing motion. The user can control the direction the robot travels by moving the head of the scope in the desired direction. Results show very good mobility through tight spaces, but the scope has poor performance in areas without walls to provide a surface to move along [39]. This system is very effective as an inspection tool, but has disadvantages in range and functionality.

### 2.2.5. SWARMANOID PROJECT

The swarmanoid project is working towards utilising large numbers of specialised robots to complete tasks. The project utilises both swarm intelligence and distributed robotic systems to achieve goals. The Swarmanoid project has three main robots being used, the Eye-bot; a flying robot that can use vision to detect obstacles or objectives, the Foot-bot; a mobile tracked robot that has a range of sensors that help to it with its objectives, and the Hand-bot; a robot with manipulators that it can use to climb vertical surfaces[40]. The swarmanoid system has shown that the robots can work together and achieve simple goals, such as retrieving a book from a book shelf [41]. These robots are not designed for USAR operations, however, the concepts of distributed robotics to solve tasks and specialised robots for tasks are very important and very much a field of interest for USAR robotics.

**Figure 2-5 Swarmanoid robots working together** [42]

### 2.2.6. RANGER AND SCOUT

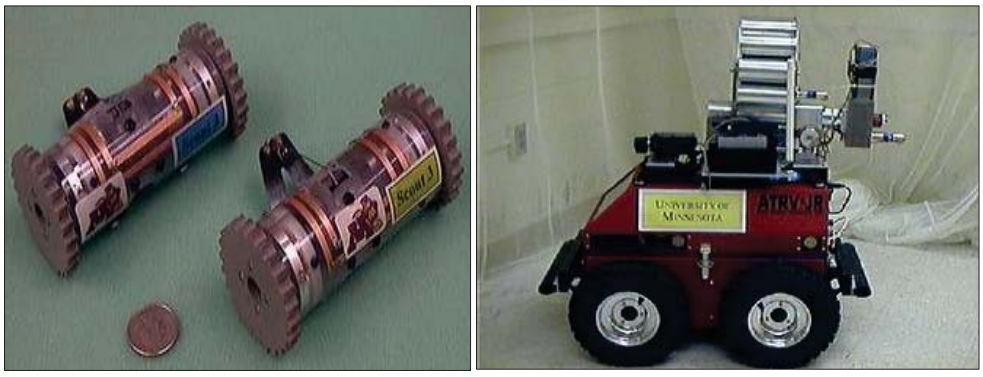


**Figure 2-6 Scout robots (Left), Ranger robot (Right) [8]**

The ranger and scout system developed by the University of Minnesota shows the use of a true marsupial system. The ranger robot, featured in the right of figure 2-6, carries the smaller scout robot, on the left of figure 2-6 and deploys them from the turret on top of it The Scouts are specialized robots that carry out low-level, usually parallel tasks aimed to meet the mission objectives [20]. The system allows multiple rangers to be deployed and dozens of scouts to be launched. The rangers communicate with each other over a 2.4 GHz frequency-hopping wireless LAN system [19]. The system is aimed at reconnaissance but could easily be adapted for USAR operations.

### 2.2.7. VICTORIA UNIVERSITY OF WELLINGTON RESEARCH

Research and development towards the three tiered system [6] has been going on for a number of years at Victoria University of Wellington and University of Waikato. This research has seen the creation of a prototype Grandmother robot , which was developed at Waikato University, and an initial Mother prototype [43]. The mother robot featured a wide range of systems and a mechanical design intended to provide the ability to navigate hazardous terrain. The Mother prototype was developed as collaboration between Victoria University of Wellington and the University of Canterbury. A fourth year engineering student from the University of Canterbury was tasked with the mechanical design and construction of the system [44]. This system was built and can been seen in figure 2-7.

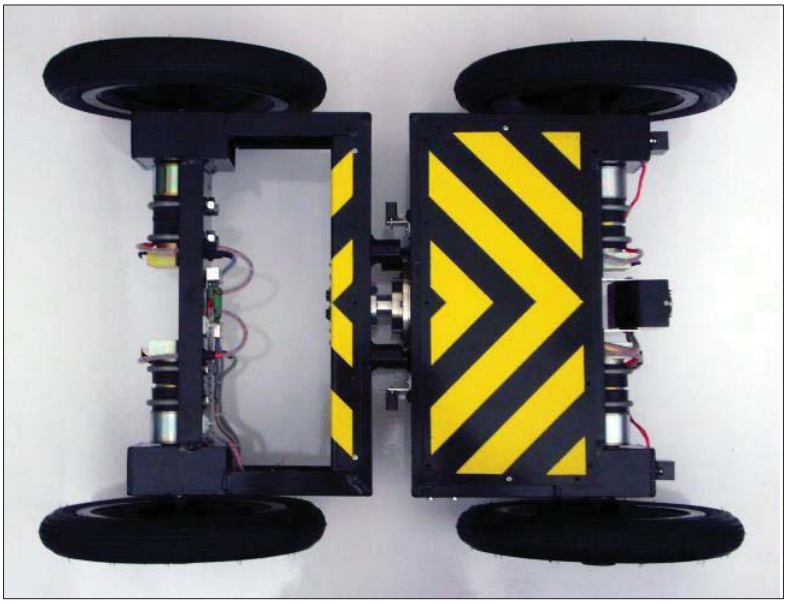

**Figure 2-7 The Mother prototype developed at Victoria University of Wellington [43]**

There were difficulties with this design when tested in a real world environment. The calculations for the required torque show a requirement of 3.8Nm per motor, however, the gearboxes were rated for 3.5Nm continuous. These oversights lead to the destruction of multiple gearboxes during testing. Due to a wide range of design decisions that were believed to conflict with the goals of this research, it was deemed necessary to start from scratch and build a new system.

## 2.3. RESEARCH SUMMARY

There is a wide range of research being conducted that all relates to, or can be utilised by, urban search and rescue robotics. The differences in locomotion techniques and locomotion control can be applied to improve the manoeuvrability and durability of a robot in a difficult environment. By incorporating the research into localisation and mapping the robot can provide much clearer maps of the environment and help rescuers pin point the location of survivors or hazards. The knowledge of exactly where the person or hazard is will minimise the response time and help save lives. Human machine interaction, while still having some subjective elements, will help rescuers and operators interact more naturally with the robot system. This will decrease training times, make data more intuitive, and potentially allow anyone to help and be involved in the rescue attempts.

# 3. ZigBee Protocol

ZigBee Wireless communications modules provide a very elegant solution to low cost, low power wireless communications. ZigBee modules are fully featured with a wide range of settings and operational capabilities and provide a solution that can be adapted for any purpose. ZigBee modules can operate in two distinct modes, transparent mode and API mode [45]. Transparent mode is a serial line replacement. Data sent to the module is transmitted wirelessly, while data received wirelessly is transmitted on the output pin. This gives a very quick and easy replacement for serial communications. API mode is more advanced and requires data to be formatted into frames. The frames allow more flexibility and access to network related functions.

## 3.1. API MODE

The API framework is a frame based system that allows the ZigBee module to complete complex tasks [45]. Using API a designer can get a ZigBee module to send data from one node to another or change a setting within the ZigBee module without having to enter a command mode. This allows flexibility of communications, network operation, and network monitoring.

A ZigBee frame can be broken down into three main parts. The first part is the header which contains the start delimiter and the length. Second is the message, which contains the message type, destination, and the data to be transmitted; and the checksum, a single byte of data used to check whether the frame has been delivered successfully. Figure 3-1shows the breakdown of an API frame.

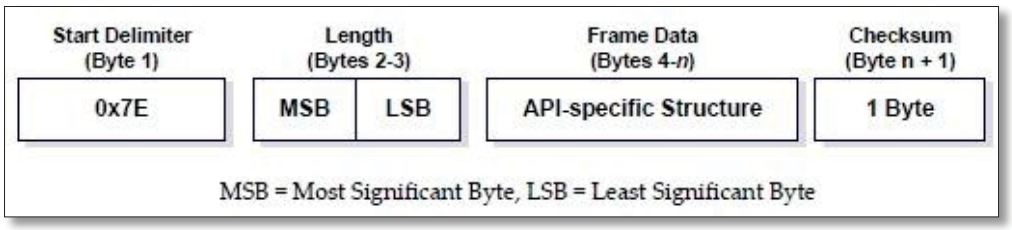


**Figure 3-1. API frame format** [45]

The start delimiter in an API frame is the tilde character, '~', and allows software to easily recognise the start of a frame. The next two bytes are the length of the packet. The length is calculated as the number of bytes within the frame data. It does not include the start delimiter, length, or checksums.

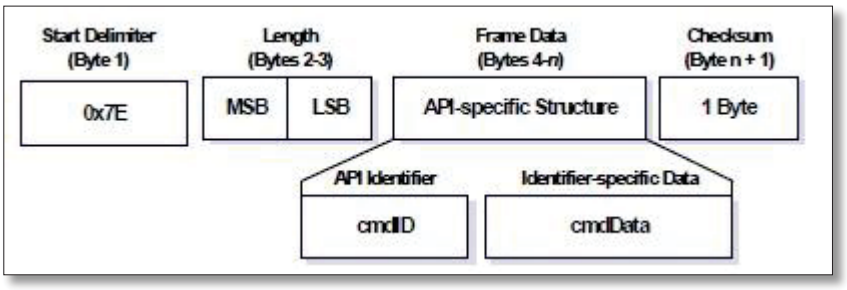


**Figure 3-2. API frame with the frame data broken down** [45]

The frame data can be broken down into two segments, the identifier and the specific data. The identifier tells the ZigBee module what the frame contains. For example, the identifier 0x10 indicates a transmit request packet. This means that the data is intended to be sent to the destination address. In this situation the destination address would be contained within the specific data. There is a wide range of API identifier codes and functions that they execute. The specific data changes depending on the function being undertaken and the software must ensure that all the data is put into the frame, or read in the case of received packets.

The checksum is the final byte. The checksum is calculated by summing all the bytes within the frame data, limited to a single byte, and then subtracted from 255. This value allows the software to check whether a received frame has been received without errors.

Using this framework and building frames to suit allows the components within our network not only to communicate with each other but change on demand to suit a situation. The API framework also allows the retrieval of useful information such as the signal strength or RSSI, or the last received packet. By checking this value we can do many things such as estimate the distance between nodes [46] or make decisions about when to deploy more nodes to maintain the communications reliability.

## 3.2. Routing Protocols

The ZigBee protocol has three defined routing methodologies. These methodologies are Ad-hoc On-Demand Distance Vectoring, or AODV, Many-to-One Routing, and Source Routing [45]. Each of these routing protocols has advantages and disadvantages for different situations.

### 3.2.1. Ad-hoc On-demand Distance Vectoring Routing

The default routing protocol of ZigBee modules is Ad-hoc On-demand Distance Vector routing, or AODV routing. AODV works by generating the frame route on demand. The source node, before sending a frame, will broadcast a route request command. The route request contains the source address, the destination, and a path cost value. This request is propagated through the network until it reaches the destination address. As each node receives the request, it will broadcast the message with an updated cost value. Once the destination node receives the request s from the neighbouring nodes, it compares the cost field. The destination node will then reply directly to the node that sent it the lowest cost value. Each node in the route will propagate this data back to the source. Once the request has been returned to the source node it can then transmit the frame along the newly acquired path. Each node, as this request propagates, will update an internal table of neighbouring nodes and store the cost value. This internal table can be used to route future packets without the need of full route requests.

### 3.2.2. Many-to-One Routing

Many-to-one routing is aimed at networks that send data to a central location or central device. While AODV may need to establish new routes for every frame transmitted, many-to-one works by establishing routes and storing the information within the internal routing table. This minimises the route discovery over head by having the central device sending a single route discovery command. Many-to-one routing is not limited to a single central collector, the network can handle multiple collector devices. This is possible by having each collector sending a routing request and the reverse routing information for each collector being stored as a different entry in the internal routing table of each node.

### 3.2.3. Source Routing

Source routing is aimed at networks where a single device is required to transmit too many remote devices. The network needs to be of a size larger than the number of routing table entries. In this network the routes established by AODV will be overwritten constantly. This would require a large number of route establishing requests and produce a large overhead and lower the speed and performance of the network. Source routing requires the application to handle storing the routes for each destination. The application stores and manages the routes that are established by

route requests. Using these routes the application can build a source routing frame that has the destination, number of hops, and the intermediate destination addresses. This allows the application to define the route to be used explicitly.

## 3.3. Device Profiles

The ZigBee protocol outlines three types of devices. These devices have different roles within the network and also have different abilities and applications. A basic description of each device type follows. A more detailed description of each device type and the operations of them can be found within the ZigBee data sheet [45].

### 3.3.1. Co-ordinator

The Co-ordinator is the first type of device in a ZigBee network. A ZigBee network requires a co-ordinator to exist, and can only support one co-ordinator per network. The co-ordinators role is to establish the network. The co-ordinators role is to find an open PAN ID, or network ID. The co-ordinator also searches for open channels, or frequencies, which can be used. Both of these searches are completed by broadcasting and waiting for replies from neighbouring networks. These replies allow the co-ordinator to determine what PAN ID's and channels are in use. The co-ordinator then establishes the network on the ID and channel. The network security policies are set from the values stored within the co-ordinator. Once the network has been established the co-ordinator can route data throughout the network and also transmit and receive data from the network.

### 3.3.2. Router

The router acts in the same manner as the co-ordinator, with the only difference being that it cannot establish ZigBee networks, only join them. Once a router has joined a network it can allow other devices to join that network, send and receive packets, and route packets within the network. When a router is turned on it performs a scan to find all nearby networks. Then using its current configuration setting it joins either a specific network or an available network.

### 3.3.3. End-device

The end device has similar functionality to the router and co-ordinator. The end device can scan for and join networks and once a member of a network can transmit and receive data. The major difference is that end devices cannot allow other devices to join the network, nor can end devices route data within the network. This is because

end devices are intended as battery powered devices that will periodically sleep to save power. An end device must have a parent device, either a router or co-ordinator, to ensure that data is always received. The parent device stores data for the end device if it is currently asleep. When the end device wakes the parent while transmit the data, of course the parent can only hold a finite amount of data and will discard the oldest data as new data arrives.

## 3.4. SETTINGS AND X-CTU

ZigBee modules can be configured to ensure they operate in the expected manner within a network. These settings can be set via network commands or set via a free application called X-CTU[47]. X-CTU provides a simple user interface that helps a programmer set the ZigBee module up in the correct way for their intended application. Figure 3-1 shows the X-CTU user interface. X-CTU categorises the variables into groups to simplifying making changes. These groups are:

- Networking
- RF Interfacing
- Serial Interfacing
- Sleep Modes
- I/O Settings
- Diagnostic Commands

Each of these groups holds variables relating to its title. These settings are not all required to be set for the ZigBee module to work as defaults are always in place. But adjusting these settings can change the way in which a ZigBee module operates, or a ZigBee network operates. These settings can be modified and stored onto the ZigBee module using the Write button. X-CTU also allows the user to read the current settings on a ZigBee module by clicking the read button. These settings vary based on the firmware version being used. The following sections outline what options are available for the firmware type used in this research.

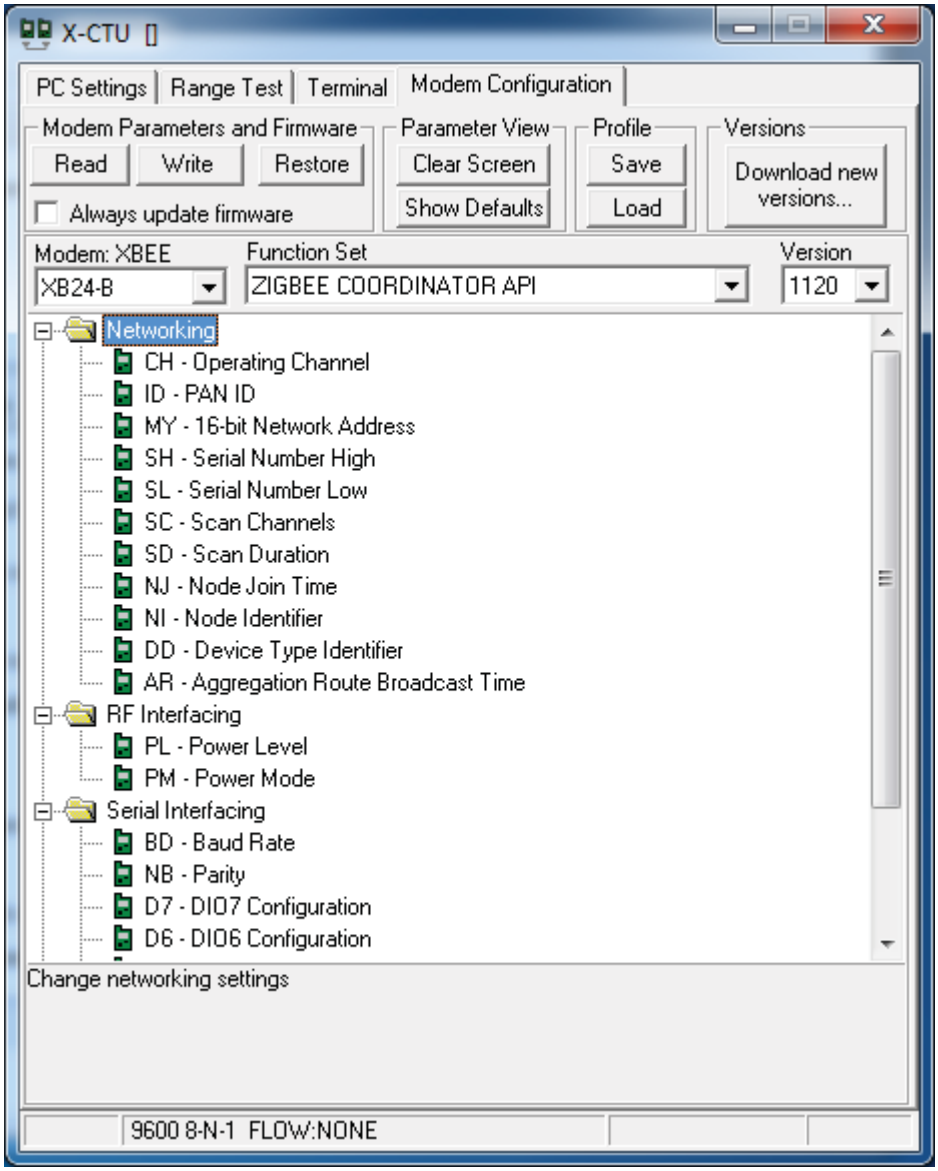


**Table 3-1. X-CTU user interface**

### 3.4.1. NETWORKING

The networking group covers settings that govern the operation of the ZigBee network. These settings define the networks that are created and the networks that can be joined by a ZigBee module. PAN ID defines an identification variable for the network. This value can either be set, or allowed to generate automatically. The co-ordinator is the only device type that can generate a PAN ID and other device types will simply store the current PAN ID value in this field. Operating channel refers to one of the 16 predefined operating channels for ZigBee modules. Much like PAN ID this is either set by a co-ordinator or read by or devices. Other settings such as Join Time change the amount of time a ZigBee module will allow nodes to join its network. This

can be useful if the network should be established early and then not allow new nodes to join. Conversely it can be set to always allow new nodes to the network, allowing a network to continue growing as it operates.

### 3.4.2. RF INTERFACING

The RF Interfacing group of settings is two values. These values refer to the power settings of the ZigBee module. Power level is a simple value that determines the output power of the module, between -7dBm and +3dBm. The other option, Power Mode, turns on or off the boost mode. Boost mode increases sensitivity and increases output power. These serve to improve the link margin and range.

### 3.4.3. SERIAL INTERFACING

Serial interfacing allows the user the ability to change the way the ZigBee module communicates between its host controller. These settings are standard Baud Rate, Parity settings for changing the rate at which the serial communications run. This group of options also allows flow control to be enabled on certain I/O pins of the ZigBee module. Outside of the serial communications this group of options allows the user to enable API mode or leave the module in AT mode. This is one of the most useful options for the ZigBee module.

### 3.4.4. SLEEP MODES

Sleep modes is a setting used to determine how long a node should buffer a message for end devices. This means that nodes will not permanently store data for an end device and will eventually discard the data if the node does not wake up. The value should match on both the parent and child for this setting.

### 3.4.5. I/O MODES

I/O mode allows the ZigBee module to use the I/O pins in a variety of ways. A pin can take on 4 different states: Analog to digital converter, digital input, digital low output, and digital high output. Each pin has hardware associated with it and X-CTU only allows the pin to take on a configuration that its hardware can handle. This means that each pin can be configured to complete a certain task, be that turning on an LED or reading a sensor signal. The final setting allows the user to change to RSSI PWM timer. This timer defines the amount of time between RSSI values being set to the pin. The RSSI value is a PWM signal that can be read to read the signal strength. The timer defines the intervals between this signal being applied to the pin.

### 3.4.6. DIAGNOSTIC COMMANDS

The final group of settings are read only. These are used to determine whether the ZigBee module is working or not. These values contain firmware version, hardware version, and supply voltage information. These can help determine if there is a node that has not been updated to the latest firmware and that is why it is not communicating correctly. These values can be read remotely using API commands and can help diagnose faults within the network.

# 4. Mother Robot Design

The Mother robot acts as a distribution system for the smaller Daughter robots. Receiving commands from the User Base Station, the Mother will enter the disaster zone and begin to drop Daughters in a manner to maximize the results for a certain task. The Mother should also be able to gather information on the environment and potentially build a map as it moves about. The overall system will not be limited to a single mother, and may have many operating in the same area to facilitate much more rapid deployment of Daughters. The Mother, in its current iteration, is a simple wheeled robot and has not been designed to handle the rigors of an urban disaster zone. The current Mother is a simple moving platform that will allow the development of the systems functionality and allow researchers to validate theories and display technologies.

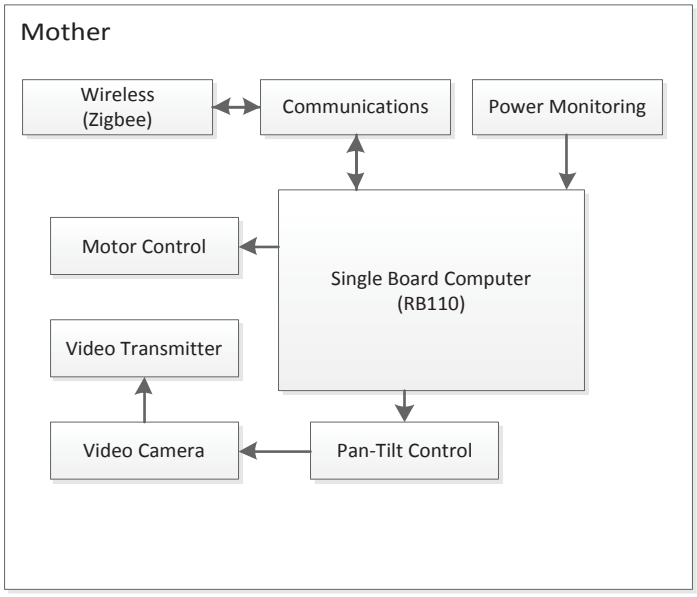


**Figure 4-1. Functional block diagram of the Mother robot**

Figure 4-1 shows the functional block diagram of the developed Mother robot. The system, in its current generation, has only the very critical features for testing the functionality of the network. The most important systems are the Motor control and the Communications, via the ZigBee module. Power monitoring was added as having knowledge of the battery levels during operation will be vital for estimating range of operation and the point of no return. During testing and development both of these factors are negligible, but due to the nature of lithium polymer batteries, it will help us ensure the batteries are not damaged, due to excessive discharging, while testing. The

video feedback system was added to the initial design because it will provide useful information for testing the usability of the system and allow us to undertake basic control tests, for example navigating an obstacle course without line of sight. The video system will be implemented in a simple fashion and the data will be transmitted independent of the control network via an off the shelf transmitter. The overall system design provides us the basic operations for this research, while still allowing a very expandable system that will be able to incorporate a large number of subsystems in future.

## 4.1. Mother Robot Chassis

The chassis design for the Mother robot is a simple structure designed with ease of manufacture and expandability in mind. The robot, shown in figure 4-2 as a CAD model, features a large area for mounting electronics and components. This was done intentionally as the system will be expanded as the research continues and restricting the amount of working area of the robot would cause difficulties at a later stage.

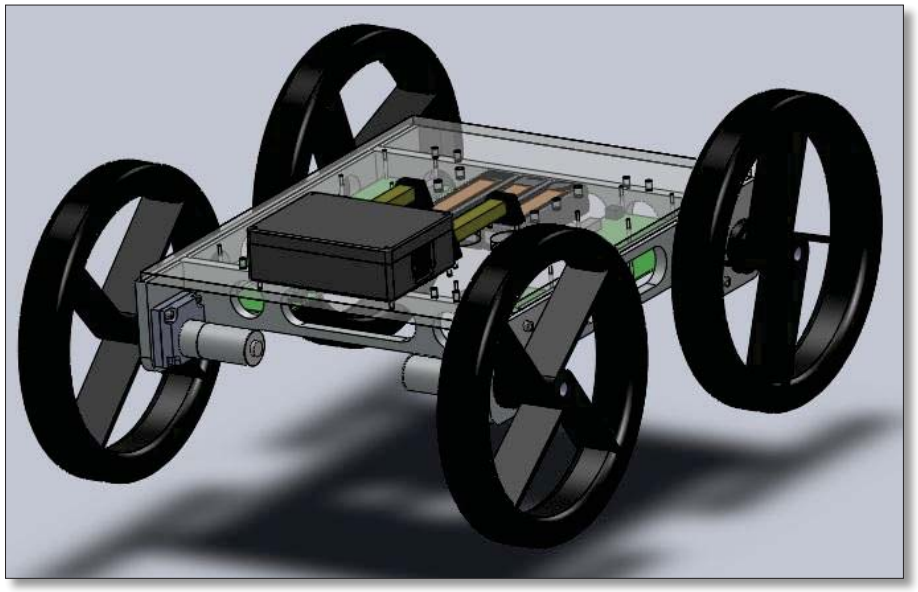

**Figure 4-2. 3D CAD model of the Mother robot**

The chassis was designed to be used in very structured environments. This is in direct contradiction to its intended application environment, but the nature of an urban search and rescue environment is so varied and challenging that it is impossible to build a universally capable design [15]. Because the challenges of locomotion and durability in a post disaster zone are so numerous and difficult, the decision was made to focus on the systems and operation of the network as opposed to the ability of the

robot to navigate these environments. This decision was made to allow the system to be fully developed and deployed into a variety of chassis, allowing the same system to utilise specialised chassis for specific tasks. This influenced the design of subsystems as well as simplifying the mechanical design. The only consideration made for the design of the chassis was the workspace of the system. The minimum door width, in a public building in New Zealand, is 710mm [69]. This value is defined to ensure the access and egress of disabled persons in a public building and may not strictly apply to residential premises. This value is the defining factor in the workspace of this system. Arbitrary values of 1000mm in length and 500mm in height were applied to the workspace to define it completely. Figure 4-3 shows the CAD model of the system within a transparent representation of the workspace. The current design fits easily into the defined workspace. This means the robot will be able to move in and out of public buildings with ease.

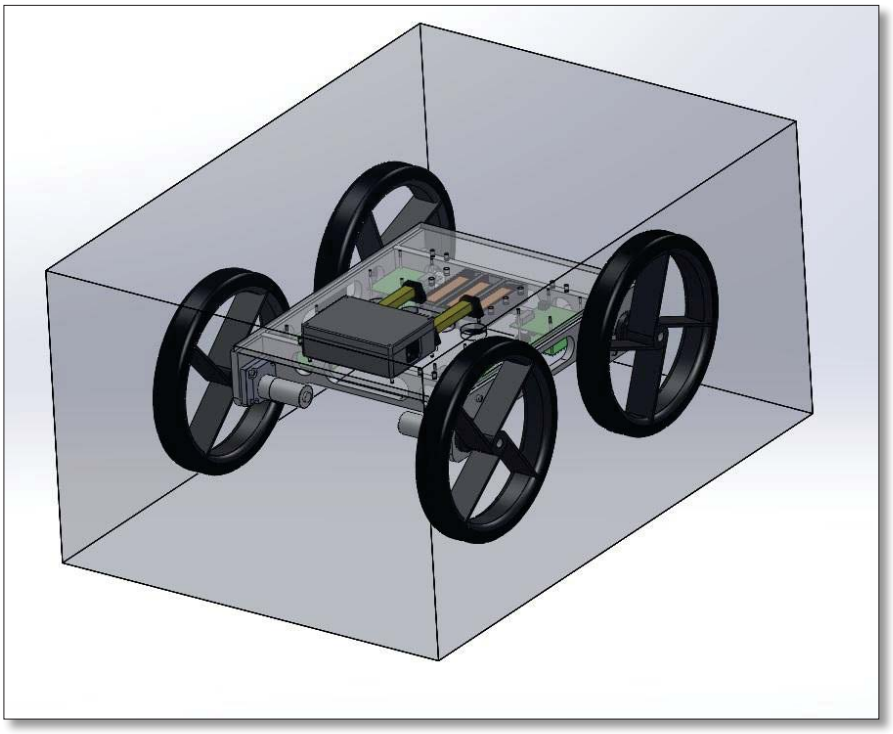

**Figure 4-3. The Mother robot shown within the workspace**

## 4.2. CHASSIS STRUCTURE

The chassis structure, as previously mentioned, was designed to maximise available space for mounting components and hardware. Figure 4-4 shows the prototype of the Mother robot. The acrylic allows us to drill holes and add thread to the, which will enable hardware to be directly screwed onto the plate giving stability

for all the components. Acrylic was used because it will allow us to mount hardware on both sides, but still be able to see indicator lights and other such features of the hardware.

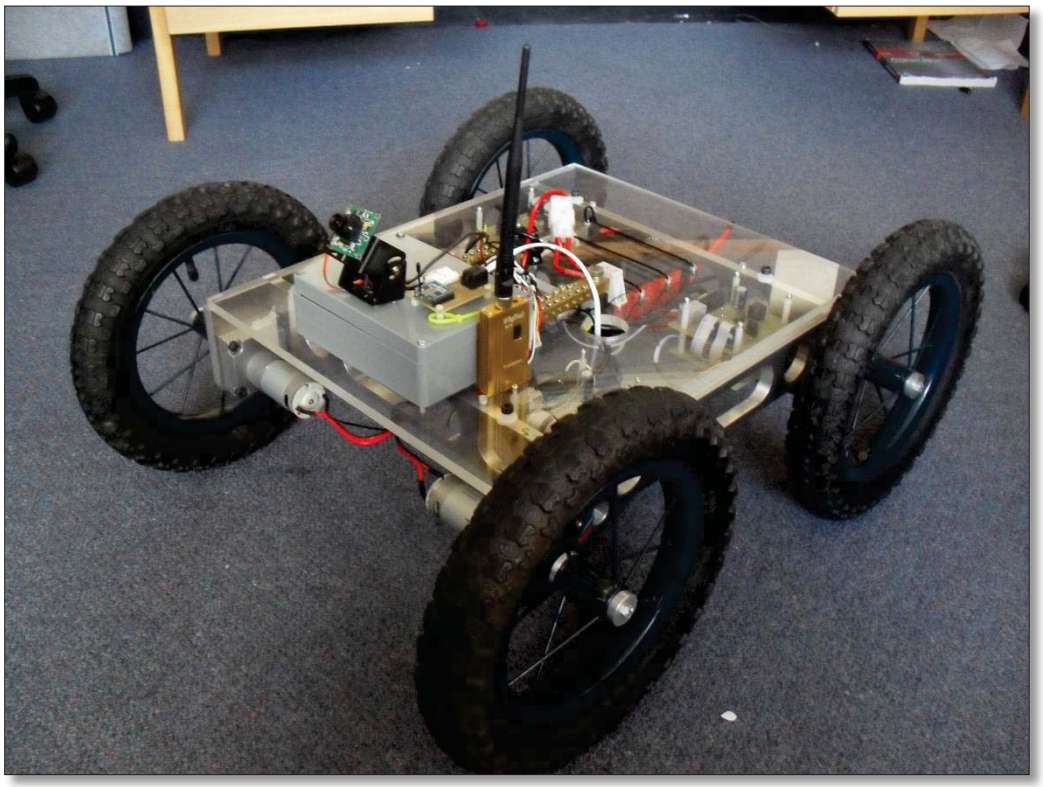

**Figure 4-4. The current version of the Mother Robot**

There are two simple rigid bodies that constitute the chassis of the Mother robot. Both are designed to look similar, with only minor differences in the dimensions and other specific features. Figure 4-5 shows a drawing of the side of the chassis. These parts were cut using the CNC machine at Massey University and are made from 13mm thick aluminium, as it readily available. The design is very simple. Each end has a large section for mounting the motor assembly and is connected by a long section with large holes cut from it. These holes were cut out to minimise the weight of the system and also to provide a very convenient carrying handle. The part show makes the side of the robot; it has slots milled into each end which allow another plate, the chassis rib, to be placed perpendicular to it. This provides very simple and intuitive assembly. The slots also provide extra strength and rigidity to the chassis once assembled.

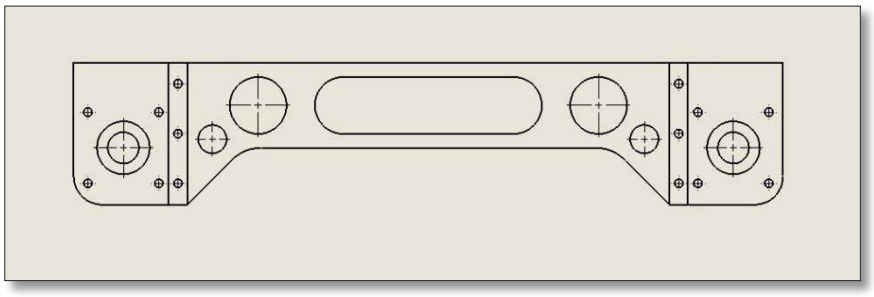

**Figure 4-5. A Solidworks drawing of the chassis side**

The design was also made in such a way that parts were symmetrical; this meant that manufacture was simplified by only having one drawing per part instead of specific drawings for left and right sides.

## 4.3. MOTORS AND ENCODERS

The motors and encoders provide the locomotion and the odometry for the system. By combining the feedback from the encoders and manipulating the control signals to the motors the system has the ability to control the speed of the motors and by extension the direction of travel, etc. The system requires 4 motors, one for each wheel, which are controlled independently. By controlling these motors in pairs they provide the ability to control the robot in the same manner as a tank. This provides the system with skid steering and the smallest possible turning circle. By driving each side in an opposing direction the system is able to turn on the spot. Skid steering is inherently inefficient as the system is forced to overcome the friction between itself and the ground to turn. While the steering is inefficient it is also highly manoeuvrable which is more beneficial to the application then improving efficiencies, at least at this stage of development.

### *A. Motors*

The motors used were chosen based on three main factors. Voltage, torque, and cost were the deciding factors when choosing the motors. The major factor was cost, due to budget constraints. The next most important factor was the torque. The motors needed to be able to drive the robot around a structured environment with relative ease. Since it had previously been determined that the environment would be a simple area, most likely flat ground, with minimal slopes we could calculate the torque required. The torque was calculated with the following assumptions:

- Total weight of the system would be under 20kg
- Wheel diameter is 12.5inchs or 0.3175m
- Centre of Gravity is in the centre of the robot
- Each wheel provides equal torque
- The robot will not accelerate faster than 1 m/s$^2$

Torque Calculation:

$$\begin{aligned}\tau &= F \times (D \div 2)\\ &= m \times a \times (D \div 2)\\ &= 20 \times 1 \times (0.3175 \div 2)\\ &= 3.175\ Nm\end{aligned}$$

$$\begin{aligned}\tau &= 3.175 \div 4\\ &= 0.794\ Nm\\ &= 8.097\ kg/cm\end{aligned}$$

With these factors in mind the best motor available was a 12V geared motor. The price point was low enough that it was suitable, as well as having a 12V rating and an output torque of 50kg/cm, a picture of which is shown in figure 4-6 [49].

**Figure 4-6 Photo of the Geared motors** [49]

## *B. Encoders*

The encoders used on this project were selected by similar criteria. Cost was held as the most important factor. Resolution and size were considered as well. The main consideration for size was that it would be mounted easily on either the drive shaft or the motor shaft. The resolution was considered in terms of where the specific encoder could be mounted. Higher resolutions would be required for drive shaft mounted encoders as there would be less turns, while lower resolutions could be used if the encoder was mounted on the motor shaft. This is due to the difference in the rotational speeds of the shafts caused by the gearbox. The gearbox has a 50:1 Ratio which means for every 50 turns of the motor shaft the drive shaft will turn once. This increases the

output torque while sacrificing speed. Therefore, a low resolution encoder on the motor shaft will turn far more and transmit more pulses than an encoder on the driver shaft. This trade off was the major consideration for the resolution. The encoders used were supplied by Phoenix Encoders. They provided a low cost solution that could easily be mounted on the drive shaft and made mechanical design simpler. Encoders from this company had also previous been used in a different project at Massey University and performed very well [50]. The encoders provide a resolution of 256 counts per revolution. This coupled with the quadrature output of the device gives 1024 distinct points and the direction of the movement. Using the knowledge of the wheel size we can determine that these will allow us to measure 0.9355mm of linear distance per count.

**Figure 4-7. Photo of the Phoenix America encoder used**

Figure 4-7 shows the encoder mounted on the outside of the chassis. This position allows it to measure the drive shaft position and calculate the speed. These calculations are done by the motor controller and allow the system to accurately control its own speed.

## 4.4. Motor Mounts

The motors provided a simple bolt pattern as a mounting system. A simple mounting assembly was design so that the motor could be mounted, with a bearing and the encoder all on the drive shaft. To mount the motors directly to the drive shaft required first a simple mounting plate the would allow the motor to be bolted to it using the holes provided on the motors face. However, the motors output shaft is not centred,. This offset forced the motor to be offset towards the top of the mounting plate. By offseting the motor position we created a design where the drive shaft would be central to the motor mounting system.

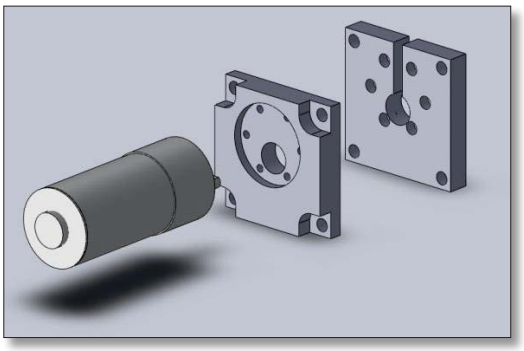

**Figure 4-8 Exploded view of the motor mounting assembly**

To prevent excess wear to the bearing the bolt heads from the motor mounting must be kept from rubbing on the bearing. This and the desire to hold the bearing in place lead to the design of the second plate. The plate features the same bolt pattern as the motor in the same positions and a long slot leading to one side. The matched patterns of the motor bolts provides a forcing function and limits the user to only one orientation for bolting the two plates together. This helps to simplify the assembly. The slot is present to allow the user to insert grub screw into the drive shaft to attached the drive shaft and motor output shaft. The grub screw prevents and slippage of the shafts and provides an easy way to remove the shaft from the chassis if necessary. Both plates and the way in which they come together can be seen in figure 4-8.

## 4.5. Mother Robot Hardware Components

The Mother robot hardware which has been developed is shown in the functional block diagram in figure 4-9. The figure shows that the entire system is focused around a single board computer. This computer, which is later detailed, provides the processing for all of the higher level algorithms and communications of the Mother. One of these communication protocols is RS485. This is connected, in a daisy chain manner, to the Motor Controllers. This communications bus provides the ability to communicate with all of the motor controllers simultaneously. Each controller has a motor driver that it is directly responsible for. The controller and driver work together to maintain the motor at a set speed that is determined by the single board computer. To do this the motor has an encoder attached to its drive shaft. This encoder sends quadrature encoded signals to the controller that provide the feedback used to maintain speed. As a protection feature a current measurement is also taken on each motor and this helps to prevent damage from excessive current draws.

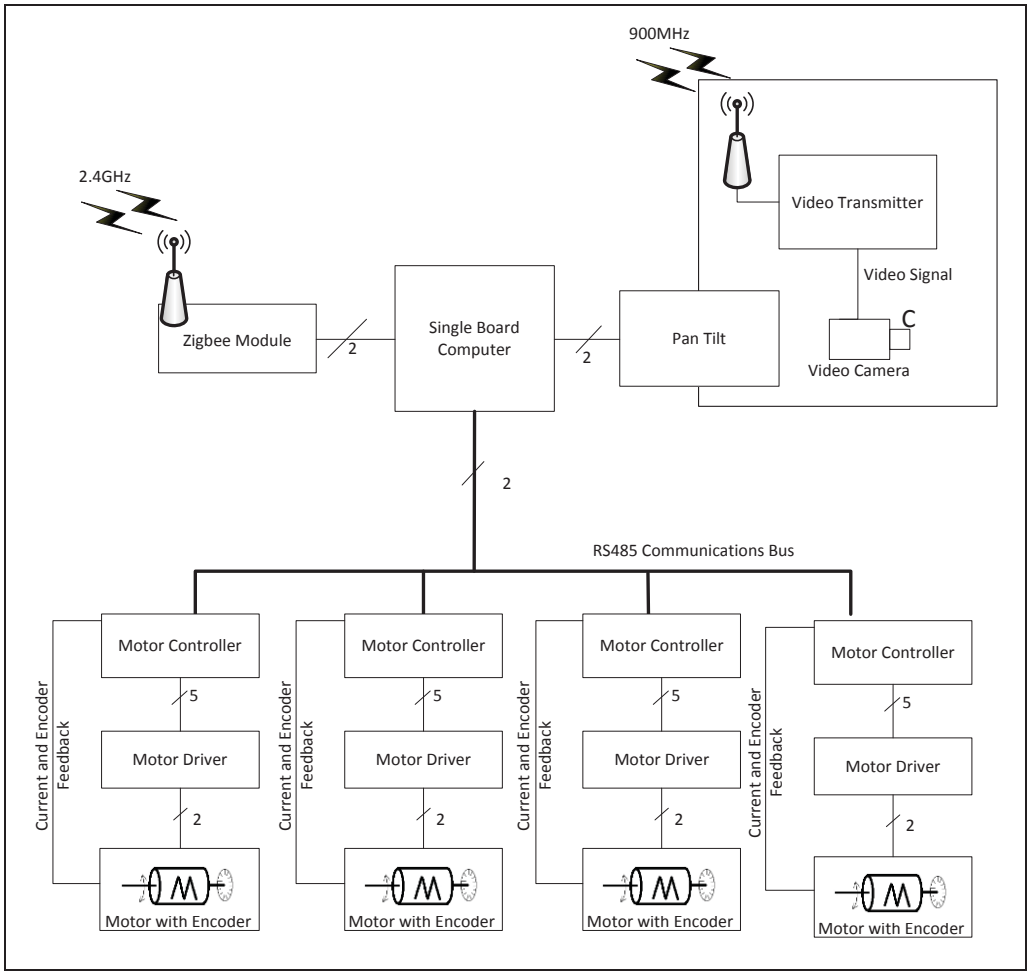


**Figure 4-9. Full functional block diagram of the Mother robot hardware**

Another communications channel used is a serial port that is connected directly to a ZigBee module. This module provides connection from the Mother robot to the network. This provides the link back to the user base station and allows the entire system to be controlled remotely.

The single board computer also controls a pan tilt mechanism that has a camera mounted on it. This camera is connected to a transmitter that sends the video to a receiver on the base station. This video is a debugging and testing feature and that is why the data is no stored, interpreted, or even routed through the single board computer.

Each of these components will be discussed in further detail in the following sections.

### 4.5.1. MOTOR CONTROLLER AND DRIVER

The Mother robot is a wheeled robot. Each wheel has a motor, gear box, and encoder attached to it to drive it and provide feedback. This provides independent control of each wheel speed. To drive and control each wheel effectively a motor driver and controller was developed. The initial consideration was to leave all the control algorithms on the single board computer. But with four wheels and the potential for including future systems, such as visions or mapping algorithms, it was decided that moving simple control algorithms, such as the speed control, away onto specific hardware would be beneficial. By creating specific hardware that was in charge of the speed control it also provides another level of flexibility. This flexibility comes from the ability to move the core systems to any robot and interface it with the motor systems. Alternatively the motor controllers can be retained and the motor drivers swapped out for more robust drivers for larger motors. This gives us the ability to use commercial drivers or specifically designed ones with minimal changes required. By doing this we have designed a system that when implemented into a different mechanical design that the system will easily be able to adapt to the requirements of the design, be that the number of motors being driven or the size of the motors.

#### *A. Motor Controller Hardware*

The motor controller was developed to provide a programmable and interchangeable control system for the motor driver. Figure 4-10 shows the functional block diagram of the motor controller system. It shows each of the subsystems that are implemented. There is provision for RS232 and RS485 communications. It also shows features added for debugging and testing purposes such as the indicator LEDs.

The RS232 communications was implemented purely for testing and learning purposes. The ability to directly monitor one board allows the user to check for faults easily from a laptop, independent of the Mother robot. The conversion from TTL logic levels to RS232 were done by using an ST232CDR IC. This chip is simple to implement requiring only four external capacitors to function. This IC has also been used before by Massey University and its reliability proven. Figure 4-11 shows the circuit diagram of the implemented RS232 convertor.

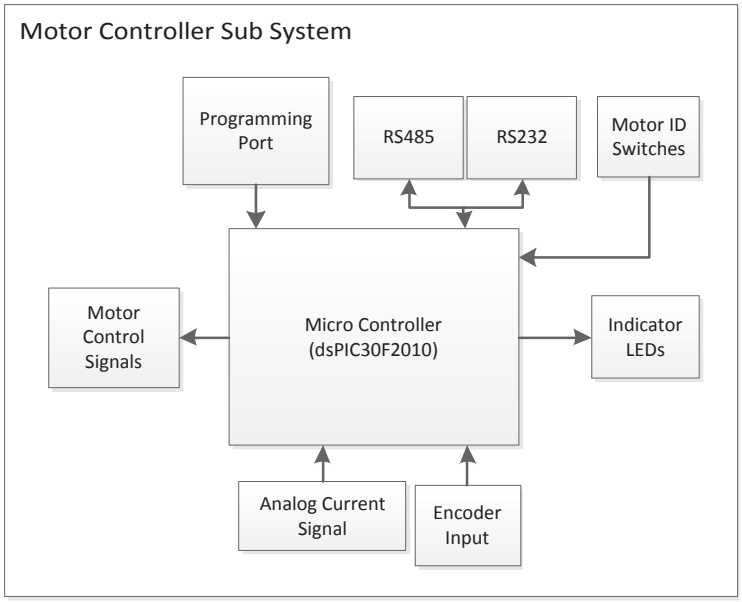


**Figure 4-10. Motor controller functional block diagram**

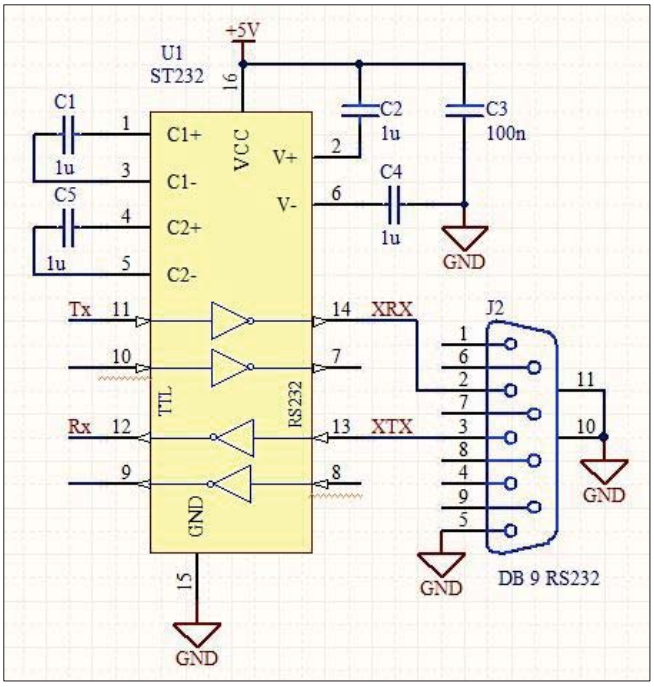


**Figure 4-11. RS232 convertor circuit**

The RS485 was implemented to provide a communications protocol that allowed a large number of motor controllers to be controlled at once. Using RS485 it allows the system to send a single message that contains data for multiple motor controllers. The controller has an ID number assigned to it and will extract the data sent to that ID from the message. This system allows for flexibility in the number of motors attached to the system. The current version has a four motor set up, but different mechanical designs may incorporate six or even eight motors. This communications system allows us to add more motor drivers and controllers as necessary. The only limitation in the current

version is the ID number, which is set via hardware jumpers. There are four jumpers, giving a maximum of sixteen ID numbers. This can quickly be overcome by assigning the ID number in software, but limits the ability to rapidly swap one controller out if it breaks or fails. The SP3485 IC converts the TTL signals into RS485 compatible signals, and unlike the ST232 it requires no external circuitry. The major difficulty with this arrangement is that the RS485 communications bus is a half-duplex communication channel, meaning that there can only be one transmitter on the channel at any time. Since the majority of the data is coming from the main computer on the Mother robot this problem only occurs when the motor drivers need to inform the Mother of an issue. To solve this, the Mother robot periodically queries the state of each controller to ensure no fault has occurred and the controller internally deals with issues such as over current.

The Controller board provides access to analog inputs and digital input output pins via two headers on the board. This provides an interface for the output of the control signals to the motor driver and allows the input of signals such as the encoder and the analog current signal. The analog current signal is fed into the on board analog to digital convertor of the micro controller. The encoder signals, however, are fed into the quadrature encoder interface, or QEI. The QEI is a hardware module built into the dsPIC30F2010 micro controller [51] and is the primary reason for selecting this micro controller over any other.

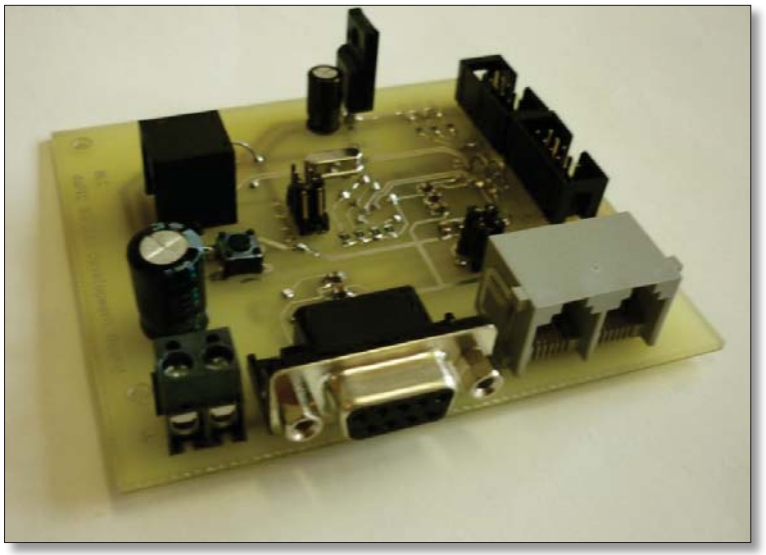

**Figure 4-12. Finished motor controller board**

The QEI has the hardware to measure the quadrature signals from the encoder and track the position of the encoder. It stores the position value as a number in an

internal register that is accessible via software. This makes measuring the encoder position incredibly simple and eliminates the need for external hardware to convert this data. The board also features three indicator LED's. These were added to allow quick feedback about the state of the machine, for example an LED could be programmed to show whether or not the motor has breached the allowed current level. This flexibility gives the user a very quick and effective diagnostic tool that will simplify fault detection.

### *B. Motor Driver Hardware*

The motor driver is a simple H-bridge design, based on the Open Source Motor Controller (OSMC) design [52] and the design of Ryan Thomas [50]. The functional block diagram shown in figure 4-10 shows the basic parts the system contains. Since the motor driver is designed to be replaceable, the motor driver only requires the full bridge for driving the motor, a method of measuring the current through the motor, and an interface for the motor and control signals.

The design was built around an H-bridge motor controller IC, the HIP4081 from Intersil. This IC is a complete high frequency, medium voltage full bridge FET driver[53]. The HIP4081 not only boast a wide range of features that make it ideal for motor driving applications, but it is also low power, making it ideal for mobile applications. The HIP4081 requires very simple input signals to driver a motor. These signals and a basic description are shown in table 4-1. The HIP4081 driver requires an input of into a combination of either AHI and BLI or BHI and ALI. This will turn on the alternate side MOSFETs and drive the motor.

| | |
|---|---|
| DIS | Enables or Disables the HIP4081 driver |
| AHI | Turns on the High side MOSFET on one side of the H-bridge |
| ALI | Turns on the Low side MOSFET on one side of the H-bridge |
| BHI | Turns on the High side MOSFET on the other side of the H-bridge |
| BLI | Turns on the Low side MOSFET on the other side of the H-bridge |

**Table 4-1. HIP 4081 input signals**

The HIP4081 also has built in shoot through protection which prevents both the high side and low side MOSFETs on the same side of the H-bridge being turned on at the same time. A PWM signal can be put into either the high side or low side to control the speed of the motor through traditional methods

One of the key features that make the HIP4081 such a simple H-bridge driver to use is the in-built charge pump circuitry that allows it to switch high side N-channel MOSFETs.

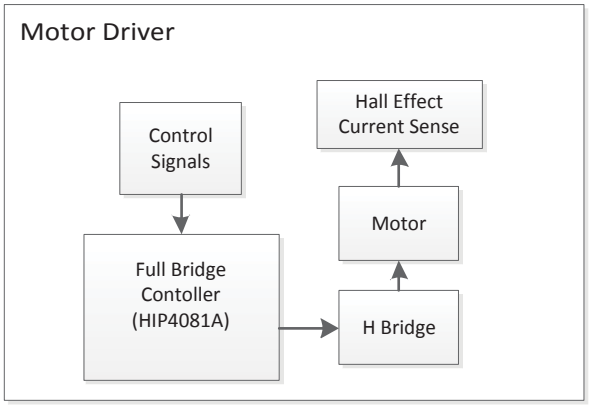


**Figure 4-13. Functional block diagram of the motor driver**

This is achieved by connecting the AHS or BHS pin to the source of the MOSFET and connecting some external circuitry to specific pins to allow the internal charge pump to work. The external circuitry, as shown in figure 4-14, is minimal and consists of a capacitor and diode for each side of the H-bridge, shown by C4, C5, D10, and D11.

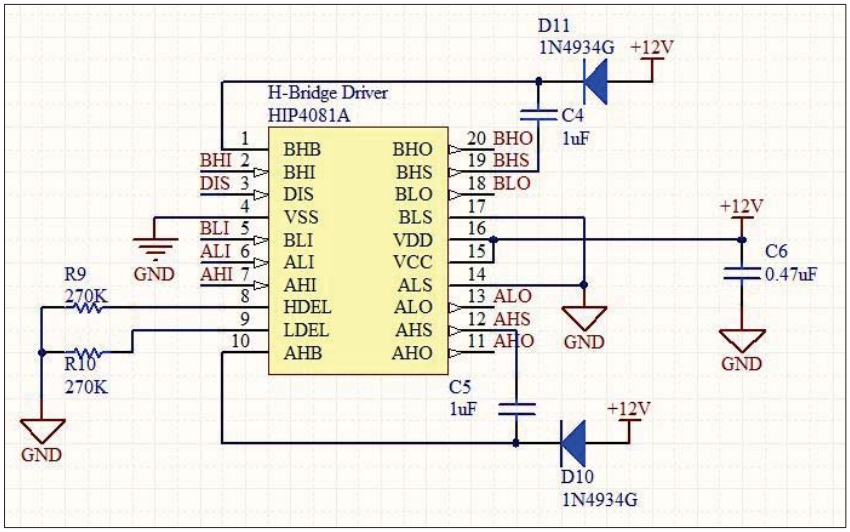


**Figure 4-14 HIP4081 External Circuitry**

The other components connected to the HIP4081 are a 0.47uF capacitor, acting as a decoupling capacitor and two 270kΩ resistors. The resistors are connected to the

HDEL and LDEL pins of the device. These pins allow the switching timing of the MOSFETS to be adjusted, so that shoot through never occurs. This circuit can be attached to an H-bridge consisting of four N-channel MOSFETs. The H-bridge uses IRF1407 N-channel MOSFETs [54]. These MOSFETs are very capable of very high continuous currents, allowing a continuous drain current of 130A. This is far beyond the requirements of the system but gives the motor driver the ability to work on much large motors if needed. The IRF1407 also boasts a power dissipation of 330W, which within this system allows us to use them without further heat sinking. The system is designed to run at 12V and that allows us 27.5A of current before requiring further heat sinks. This means that the H-bridge, shown in figure 4-15, will be able to run the motors easily.

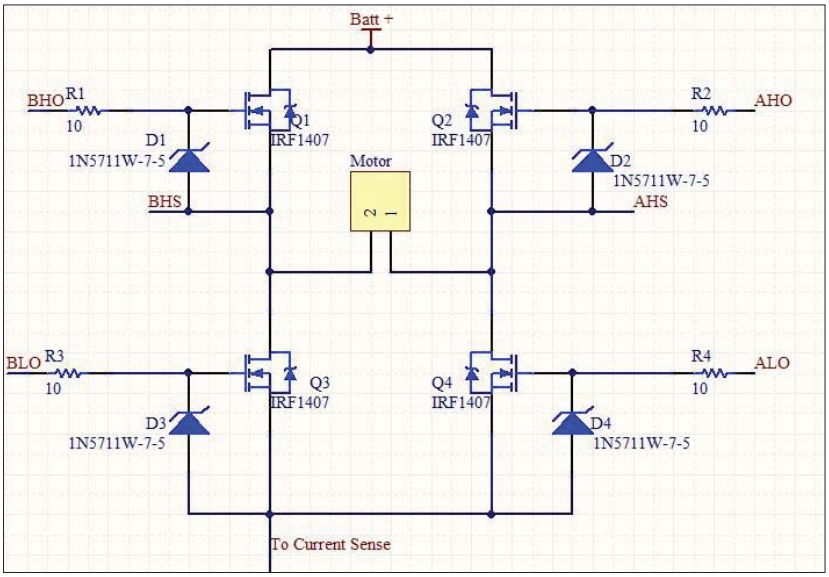


**Figure 4-15 H-bridge circuit used in the Motor Driver**

Protection for the MOSFETs is included in the form of zener diodes and resistors on the gate. The zener diodes prevent the voltage exceeding the allowable gate source voltage, while the resistors ensure the current is within allowable limits. The major difference between this H-bridge and others is the lack of fly wheel diodes. These have intentionally been left out as the IRF1407 MOSFETs have built in diodes which carry out this functionality. Inclusion of fly wheel diodes has been shown, by Thomas [50], to cause issues with this system and impede functionality.

The final important component of the Motor driver system is the current sensing device. This device is crucial as it provides feedback on the amount of current being used and allows the system to decide whether or not to cease action to prevent damage to the motor or driver. The device used is the ACS712-30A from Allegro

Microsystems [55]. The ACS712 device is a hall effect based current sensing IC. It provides a linear output proportional to the current flowing through the device. The linear output of the device makes interfacing with it very simple. It can be used in conjunction with a comparator to check absolute level of the current or, as implemented in this designed, directly interfaced with an ADC. By connecting the output directly to the ADC on a microcontroller it allows the system to see the actual current. This data could be used later on to develop a motor controller that not only prevents over current damage but can make decisions about the state of the wheel using the speed and current values. Figure 4-16 shows the basic circuit used to implement this system and figure 4-17 shows the assembled motor driver board.

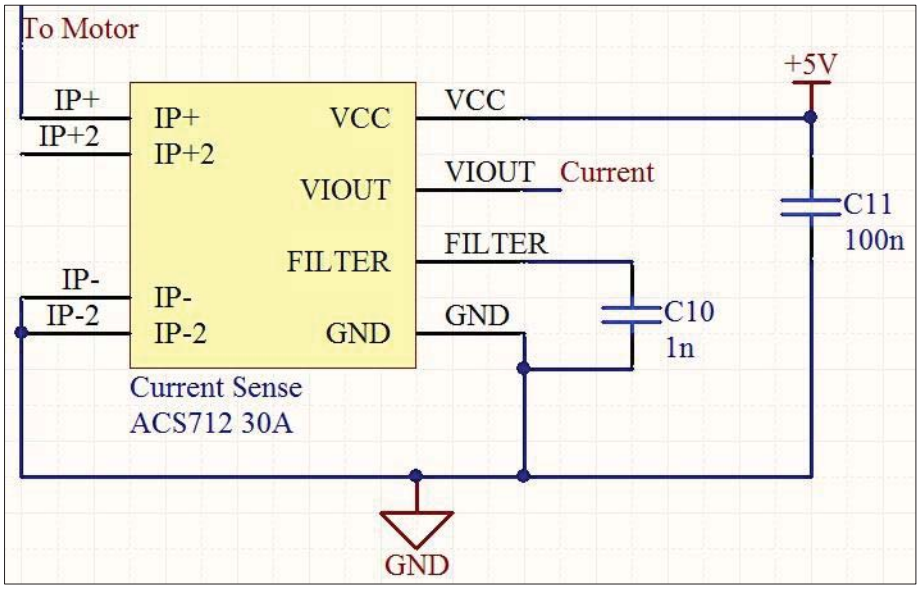


**Figure 4-16 Current Sensing Circuitry**

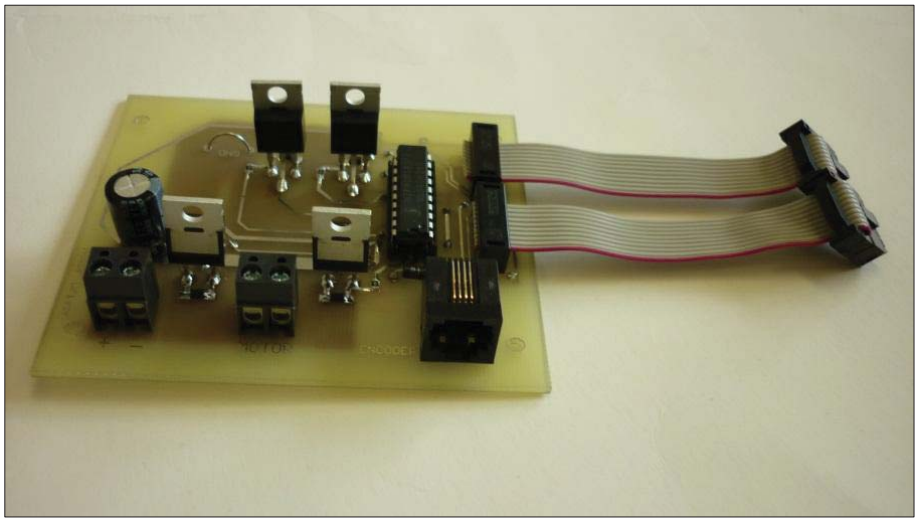

**Figure 4-17. Complete motor driver board**

### 4.5.2. VIDEO CAMERA SETUP

The Mother will act as a tele-operated robot and because of this there is a need for some sort of video feedback so that the operator can see whether there are obstacles in the path of the robot. This feature is not critical at this stage of development and due to this a low cost off the shelf video camera and transmitter were purchased. The video set up was a 1/3" CCD camera with a 1500W 900MHz transmitter [56]. This set up is shown in figure 4-18.

**Figure 4-18. Camera, Receiver, and Transmitter set from Hobby king** [56]

The camera is mounted on an off the shelf pan tilt system [57]. This is connected directly to the servo outputs of the Single Board Computer. The system can be controlled by the user. This allows the user to see what the robot can see and if needed look around and investigate the environment.

### 4.5.3. BATTERIES

The batteries used on the Mother robot are 4S 30C Lithium Polymer batteries [58]. These batteries have a nominal voltage of 14.8V, 3.7Ah capacity, and a continuous discharge rate of 111 amps. This provides a continuous supply current that exceeds the needs of the Mother. LiPo was chosen due to its high power to weight ratio. These batteries provide a long operational time without adding excessive weight to the robot. The other reason for using LiPo was the low price. Compared to other battery types, LiPo provided the best option to ensure that requirements for operation were met while maintaining a strict budget.


**Figure 4-19. 4S LiPo Battery used for powering the Mother** [58]

The Mother is fitted with three of these batteries, in parallel, to increase the capacity available. This increases the capacity of the Mother to 11.1Ah. This means the mother can draw 11.1 amps continuously for 1 hour. The only drawback to this configuration is the possibility of one battery discharge faster than the others. If this situation happens then, due to the voltage differences, then battery will be charged by the other batteries. This can cause the batteries to discharge at a rate higher than they are rated for. This can cause damage to the batteries. However, if the batteries are all fully charged before use then this is unlikely to happen.

### 4.5.4. Single Board Computer

The core of the Mother robot is built around a single board computer. While at this stage in development the Mother could be based upon a simple micro controller to receive commands and send them to the appropriate peripherals, later developments will require more flexible and powerful processing. For this reason a single board computer was chosen, as it will provide the ability to process complex algorithms, image processing, and still allows for basic I/O control natively.

The computer chosen was the RoBoard RB-110 [59]. The RB-110 features a wide range of features most useful of which is a large number of communications ports. The RB-110 features two high speed serial interfaces, two TTL serial interfaces, an RS-232 interface, an RS-485 interface, SPI, and I2C interfaces. This is an incredibly versatile number and selection of communications interfaces allowing the RB-110 to easily be connected to a large number of devices or subsystems. This flexibility is essential when developing a complex system such as the Mother robot. On top of the huge number of communications interfaces there is also an on board analog to digital convertor which enables sensors to be directly connected to the computer without external hardware. The RB-110 features 16 pulse width modulated, or PWM, outputs. These can be used to drive servo motors or feed into other circuitry to provide a reference signal. Figure 4-20

shows the RB-110 in its standard configuration. The RB-110 does not provide a default video output system, but a video card can be purchased and installed in the mini PCI slot.

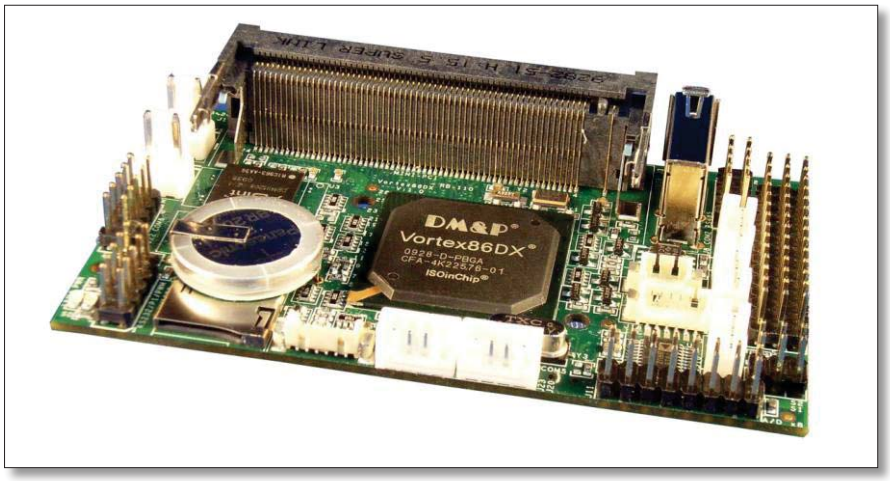


**Figure 4-20. The RoBoard RB-110 single board computer** [59]

By installing the video card the RB-110 provides a fully functional computer that can run Linux or Windows operating systems and will be very suitable for use as the main processor for the Mother robotic system.

## 4.6. Mother Robot Software

The Mother robot software is very simple for this current prototype. There are no higher level task, automation, or vision algorithms being implemented at this stage. The current software is developed to provide the functionality to test individual components. This software only needs to be able to take in data from the network and interpret it, the data is then transferred to the subsystem that uses it. The Mother robot software has no user interface as there is no attached screen on the robot. This lack of graphical interface makes the code much harder to debug and therefore must be made as robust as possible.

### 4.6.1. Mother Single Board Computer Software

The software for the Mother robot is designed to allow the robot to be controlled remotely. The major functions of the software at this stage are communications with the network and communications with the motor controller subsystems. The single board computer, previously discussed, is relatively low powered compared to current generation computers. With this in mind the software was developed in visual basic 6. Visual Basic 6 has existed over 10 years and can run on low powered computer very

easily. Roboard, the manufacturers of the single board computer, have developed a library of functions for the RB-110 that work within Visual Basic 6. This makes developing software that can take advantage of all the features of the hardware fast and simple.

### *A. ZigBee Communications*

The ZigBee communications software is designed to provide an interface for the Mother robot with the network. The communications must be able to send and receive API frames within the network and have the ability to work with AT commands both locally and remotely. Since ZigBee communications will be needed on both the Mother and Base Station computers, a specialised ZigBee communication library was developed to speed up development and ensure that all the functionality was available to both systems. This module would handle all functions from building API frames to checking the checksum of received frames.

### *B. Sending Frames*

The first major function of the software is to send packets formatted following the API framework. To build an API frame that is easily manipulated and displayed the software builds it using ASCII characters, and later turns the values into bytes for transmission. Figures 4-21 shows the how the delimiter value is set to “7E”, which is the hex value of the tilde character which acts as the start delimiter for an API frame. The software then builds a header variable, made of the API type, which is either transmit, remote command, etc. The other part of the header is the frame ID, it is hardcoded to “00” which will transmit the packet and not request an acknowledgment. The options field and 16 bit addresses are also hardcoded to values that will work in this case and simplify usability. The 64bit address is set from an input to the method. This must be in the form of a string, for example “0013A20040773345” is the address of one of the ZigBee modules used.

```
'Header and option data
delimiter = "7E"  'Standard Start Delimiter
header = apiType + "00" 'Frame type, Frame ID no ack
address64 = addr64 '64-bit Address of destination
address16 = "FFFE" 'Ignore the 16-bit address
options = "0000" 'broadcast radius, options
```

**Figure 4-21. Start delimiter, frame type and address data for an API frame**

The next step is to build the body of the API frame. This is done, again, by taking the message input and storing it in a variable. Since the frame is being built as a string of hex values, the message must be turned into hex values before being stored in the body variable. Once the message is converted into a usable form, the body is built by concatenating the header, 64-bit address, 16-bit address, options, and body strings. Figure 4-22 shows this process.

```
'Build body as string of Hex values
body = StringToHex(message)
body = header + address64 + address16 + options + body
checksum = calcChecksum(body) 'calculate checksum as string
length = calcLength(body)
```

**Figure 4-22. Building the body of the API frame and calculating required values**

The body is now a long string of all the values between the length and the checksum. By reading, and summing these values we can calculate the checksum value for the packet. This also gives us the length of the packet which can be calculated.

```
'Build frame from delimiter, length, body, and checksum
frame = delimiter + length + body + checksum
APIframe = HexStrToByte(frame)

sendAPI = APIframe
```

**Figure 4-23. Build the API frame in the proper format and then convert into bytes for transmission**

Figure 4-23 shows the final step for sending an API frame. The frame is built in the proper format by concatenating the delimiter, length, body, and checksum. This frame is then converted into bytes and returned from the method. The value that is returned is sent directly to the serial port for transmission. This process works very well and can be used for any of the API commands shown in the ZigBee manual [45].

To send an AT frame we use the same basic format, with a different header. To make the software easier to use a new function was created to do this. The frame type is hardcoded to “08”. Figure 4-24 shows the code that completes this. There are parameters which can be used to set different AT parameters; this exists as an input to the function.

```
'Header and option data
delimiter = "7E"
header = "0801" 'Frame type AT command, Frame ID
options = Para
```

**Figure 4-24. Start delimiter for an AT command frame**

The software then executes the same code to build the rest of the frame, however, instead of a message the AT frame uses one of the predefined AT commands. This determines what value is being set or retrieved.

### *C. Receiving Frames*

The software to receive an API frame is more complex. The serial port can often receive more than one API frame before the software executes. This means the software needs to go through all the data in the serial port and extract the API frames.

```
' Check for data
For i = 0 To m - 5 Step 1
    temp = Hex$(msg(i))
    If (temp = "7E" And i < m - 2) Then
        l = 0
        For k = i To k + msg(i + 2) + 3 Step 1 'grab the packet
            MSGtemp(l) = msg(k)
            l = l + 1

        Next k
        i = k - 1 'set next i to the start of the next packet
        packets(numPackets) = MSGtemp 'store into the packet ar
        numPackets = numPackets + 1
    End If
Next i
```

**Figure 4-25. Software for extracting received API frames**

To ensure that the packets are being received completely, and all packets received the software loops over the received data searching for the start delimiter '~' or 0x7E. Once a start delimited has been found the software then build up a temporary message from the start delimiter, to the end of the frame. The end of the frame is determined by looking at the length value within the frame and adding the 3. This will ensure the software gets all the relevant data from the start delimiter to the checksum. Once this is done the software stores the temporary message into an array and increments the packets count. The software then resumes its search for start delimiters from the next byte. This loop continues until there is no more received data to check. This is shown in the code in figure 4-25.

Each received frame is then read and then tested to ensure the frame was received without errors. The error check is done by checking whether or not the sum of the bytes from the length to the checksum, and including the checksum, sum to 255. If this is true then the packet has been received correctly. If the packet has been received without errors then the frame ID is read to see what type of data has been received. The software then executes a function that will act appropriately on the data. The read data method purely takes the data and sets the received string to a message box, or label. Figure 4-27 shows the atResponse function for reading frame that contain AT messages, such as the RSSI value or network discovery addresses.

```
For i = 0 To numPackets - 1 'loop over all received pack
    If (testChecksum(packets(i))) Then 'checksum returns
            l = packets(i)(3) 'read the frame ID value
            Select Case l
                        Case 139 'Acknowledge frame

                        Case 136 'AT response frame
                            atResponse packets(i)
                        Case 144 'Data recevied frame
                           readData
            End Select
    End If
Next i
```

**Figure 4-26. Check all API frames and respond based on the type of frame**

```
Public Function atResponse(ByVal msg As Variant)
Dim temp As String, a As String, b As String, address As String
Dim i As Long
            a = "&H" + Hex$(msg(5))
            a = Chr(val(a))
            b = "&H" + Hex$(msg(6))
            b = Chr(val(b))
            temp = a + b

        Select Case temp
            Case "ND"
                 For i = 10 To 17 'loop over address data
                    temp = Hex$(msg(i)) ' Add 64bit address of dev
                    If Len(temp) = 1 Then temp = "0" + temp 'pad o
                    address = address + temp 'build the address
                 Next i
                 Form1.foundAddress.AddItem (address) 'add the add
            Case "MY"
                MsgBox a + b, vbOKOnly
            Case "DB"
               temp = "&H" + Hex$(msg(8)) 'read the RSSI value
                Form1.rssiVal.Caption = CStr(val(temp)) 'turn the
                Form1.rssiValue = val(temp) 'use the integer value
        End Select

End Function
```

**Figure 4-27. Code snippet of the atRepsonse function**

## *D. Motor Control*

The motor control software is done in a two-step process. First receive and manipulate the control data and second, transmit the data to the motors. The first step requires checking whether the received data is a control packet or not. Figure 4-28 shows the code for checking this. Currently the check is simply checking if the first character is an “L”. Once the software and transmission become more complex this may need to change to a more unique character or value. If an “L” is the first character the software finds the index of the “R” and stores this in a variable.

```
If Mid$(recMsg.Text, 1, 1) = "L" Then 'Data is a Control F
For i = 1 To Len(recMsg.Text)
    If (Mid$(recMsg.Text, i, 1) = "R") Then
        r = i
    End If
Next i
    rightval = val(Mid$(recMsg.Text, r + 1, Len(recMsg.Tex
    leftval = val(Mid$(recMsg.Text, 2, r))
End If
```

**Figure 4-28. Checking for control data**

Figure 4-28 shows the software for retrieving the values for the left and right stick from the received string. The code simply takes substrings from the received string

based on the index of the “L” and “R”. These are turned into integer values and stored in appropriate variables. These values that are received are in twos complement format and the next step is to transform these into a signed integer format so that the data can be sent.

```
Private Sub motor_output()
Dim i As Integer
        control(2) = "&H" + Hex$(recRight)  'right back
        control(6) = "&H" + Hex$(recRight) 'right front
        control(10) = "&H" + Hex$(recLeft) 'left back
        control(14) = "&H" + Hex$(recLeft) 'left front
           For i = 0 To 15
              MotorControl.Output = Chr(control(i))
           Next i
End Sub
```

**Figure 4-29. Transmitting the control data to the motors**

Once the data has been manipulated to the correct format for transmitting it is stored in the recRight and recLeft variables. Top transmit the data to the motors an array is initialised that is 16 bytes long, 4 for each motors. The array is initialised with the appropriate addresses, start bytes, and end bytes. When the motor output sub routine is called is overwrites the speed values within the array with the new speed values. These values are formatted in a way that ensures the data will be sent in the correct way for the motors to interpret. The final step is to loop over the length of the array and transmit all the bytes, thus transmitting all the data to the motors. The motors will remove the data destined for them and act appropriately.

### 4.6.2. MOTOR CONTROLLER SOFTWARE

The software for the Motor controller was developed for the aforementioned dsPIC30F2010. The software was designed to be an independent, self-contained system. The aim of this was to take simple, but repetitive, tasks such as the calculations for a control loop and move them away from the central processing system. To do this the controller needed to be able to receive commands from the central computer and react to these. Primarily, when a set speed was transmitted to the controller it must change its output to ensure that the wheel spun at the correct speed. Other internal functions that the controller required were current monitoring and the ability to respond to the central computer if required.

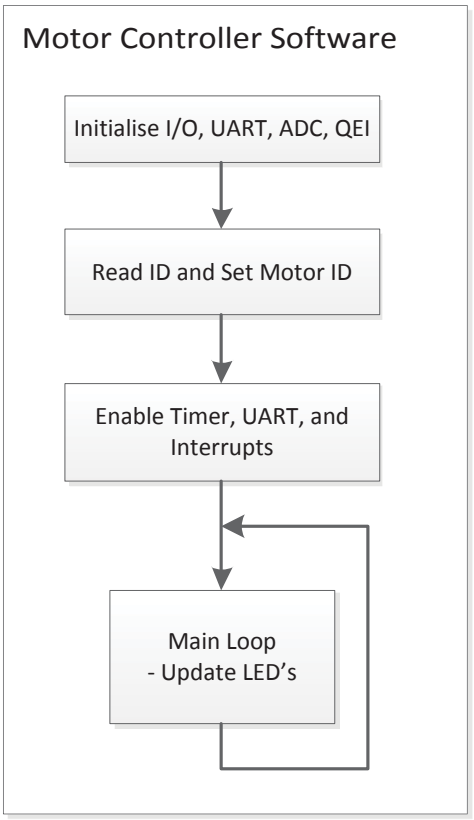


**Figure 4-30. Flow diagram of the Motor Controller software**

The flow diagram, shown in figure 4-30, outlines the basic operation of the software works. The first part of this is to initialise all of the hardware peripherals. This includes clock sources, the UART, the Timer, the ADC, and the QEI. The code for initialising these is included in the supplementary material and specific details about how they are configured will not be discussed.

After this phase the software reads the state of the DIP switches and from that calculates the ID of the motor driver. Figure 4-31 shows the code used to calculate this ID.

```
int getID(void) //Get the ID of the Motor from the jumpers
{
        int newID = 0x00; //blank the ID

        //Sum the ID pins to create the Motor ID value
        if(DIP1) { newID += 0x0008; }
        if(DIP2) { newID += 0x0004; }
        if(DIP3) { newID += 0x0002; }
        if(DIP4) { newID += 0x0001; }
        return newID;
}
```

**Figure 4-31. Motor ID calculation code**

The code moves over each pin and checks its state. If the pin is high the corresponding binary value is added to the newID variable. This value is returned at the end of the method and stored in the Motor ID variable for use in determining whether data is intended for this driver or another.

The main loop only updates the on board LED's in the final build of the software. This is not necessary and was only added for debugging and testing purposes. The LEDs can be controlled by any signal, or by a combination of signals to show complex functions. The current generation has one LED for the state of the start variable, one for the direction, and the last to show whether an over current event has occurred. These are often changed depending on the function being tested. The main loop executes no important actions as the important functions, communications and control, are executed by interrupts. This is to ensure that the control loop executes at the same rate at all times and that no communications are missed.

### *A. Interrupts*

The interrupts are the backbone of this software. The ability to maintain constant communication, monitors the current draw of the motors, and execute control loop at a constant rate makes the system work as well as it does. The control loop is the highest priority interrupt. This is because the control is the main function this software must have. The software that executes during these interrupts will be explained in great detail in the next two sections.

### *B. Communications*

The communications interrupt is designed to read the incoming data and interpret it accordingly. The normal data that is sent is broken into four bytes. The first byte is hex 55 and is simply a start byte; this is the true with the fourth byte as well which is hex AA. The two middle bytes constitute the data. The first is broken into its component nibbles. The first nibble is the ID data. This directly corresponds to the ID set during start up. The second nibble gives the command data. This gives us the option to execute up to 16 different commands. This is not used heavily in the current software but allows for future expansion and for testing/debugging to be done. The third byte is the data related to the command, if any is needed. For example, a speed command uses the most significant bit of this byte to convey direction, 1 for reverse and 0 for forward, and the other seven bits for the speed. This gives a speed range of

127, which is far higher than we will ever need. To read this data at the correct points a switch statement is used, shown in figure 4-32.

```
switch(i){
        case(1):
            if(temp == 0x55){
                header = 1;
            }
            else i = 0;
            break;

        case(2):
            if(ID == (temp>>4) && header == 1){ //check header
                CMD = temp & 0x0F; //Remove CMD data
            }
            else header = 0; //Else dont update next data
            break;

        case(3):
            if(header ==1){ //check if update allowed
                runCMD(temp);
            }
            break;

        case(4):

            header = 0; //stop bit, clear variables
            i = 0;
            break;
}
```

**Figure 4-32. Software used to store received communications byte in the correct placeholder.**

This switch statement first checks that the header byte is correct, this is purely to ensure that we start from the beginning of the data and that data is not erroneous. The next step is to extract the ID from the upper nibble of the second byte. If this and the header are correct the command data is stored into a variable for later use. The next byte is read and the value of the byte is passed as a parameter to the runCMD method. This method is shown in figure 4-33. The method works by using the command value to switch between different operations. Case 1, set speed, is the most commonly used. It reads the direction, from the data, and sets the rec_dir variable to this value. If previous rec_dir is not the same then the system stops the motor. This is to prevent rapid changes in direction. The important line of code is where the quad encoder control register is altered. The bit set, SWPAB, controls whether the encoder inputs have been swapped. By changing these when we change direction we ensure that the

encoder continues to increment the internal register, simplifying the maths for the control loop.

```
// Switches between commands sent from the Master
// Allows for action to be taken on one of 16 tasks.
void runCMD(int temp)
{
        switch(CMD){ //command can range from 0 - 15 eac
            case(1):    //update desired speed
                if((temp & 0x80) == 0x80){ //check the d
                    QEICONbits.SWPAB = 0; // left side =
                        if(rec_dir == 0) { stop(); }
                    rec_dir = 1; //positive direction
                }
                else {
                    if(rec_dir == 1) { stop(); }
                    rec_dir = 0; //negative direction
                    QEICONbits.SWPAB = 1; //ensures the
                }
                 // set the received speed value
                rec_speed = temp&0x7F;
                start = 1;
                break;
            case(2):    //Return ID
                U1TXREG = ID;
                break;
            case(3): //Stop!
                start = 0;
                I = 0; //Clear I
                prev_error = 0; //clear previous error
                prev_pos = 0;
                U1TXREG = 'S';
                break;
        }
}
```

**Figure 4-33. Software used to execute commands. Commands are selected by an index, CMD, and each command has a different purpose.**

The UART interrupt also has global commands that are executable. These are executed if the ID in the transmitted packet is 0. This means that a packet has been sent as a global command and all motor drivers should act upon its contents. This ensures that we can issue a system wide stop command or change in speed if necessary.

### *C. Control*

The control loop is the most important part of this software. It is tasked with ensuring that the output of the motor is at the correct speed. This control software is called within the timer interrupt. The timer has been set up to run for 10ms. This means the control loop will be run every 10ms and must complete its functions within this time frame. Figure 4-34 shows the timer interrupt. The control loop is executed

only if the start variable is set. This allows the software to omit control if necessary, such as after an overcurrent event or before a control command has been sent.

```
//_T1Interrupt() is the Timer1 Interrupt Service Routine
//The routine must have global scope in order to be an ISR.
//The ISR does the PID loop mathematics
void __attribute__((__interrupt__)) _T1Interrupt(void)
{
        if(start == 1) //if the start flag is set run the control loop
        {
            control();
        }
        else //else stop all the motors!
        {
            stop();
        }
        IFS0bits.T1IF = 0; //Clear Timer1 Interrupt Flag
}
```

**Figure 4-34. Timer Interrupt service routine**

The control loop itself follows a very simple structure and utilises proportional, integral, and derivative control, or PID control. PID control is widely known amongst engineers and is very commonly used. The software structure has been modified since the publication of the ICARA paper [9]. The software used in those tests determined the direction the motor should spin by checking the 8$^{th}$ bit of the received data. This led to a situation, which was not obvious during testing, where the control failed if an external force was applied in the opposite direction. This was due to the large error being produced causing a large reaction from the control loop in the direction set. This meant that during operation if one wheel was to stop faster than the other it would measure a rotation on the wheel that was caused by the other wheels. This has been remedied by having the control effort determine the direction of rotation from within the control loop. Figure 4-39 shows the updated control software flow diagram. The other major change since the ICARA paper is that speed profiling has been included within the control loop and the proportional gain is now a constant value.

The first step of the control loop is to check for an over current situation. An over current situation will require that the motor be stopped, or disabled, to prevent damage to any of the subsystems. The ADC interrupt, which will be explained in the next section, sets a variable to 1 if an over current situation has happened. The control software checks to see if this value is set. 4-34 shows the IF statement used to check the variable. If the variable is set then the software will clear the start flag, and set the duty cycle to 0.

```
//Check for over current
if(over_current == 1)
{
    start = 0; //clear the start flag (stopping the control loop)
    duty = 0; //set duty to 0 stopping the motors
}
else
{
```

**Figure 4-35. If statement for checking for over current situations**

If an over current situation has not been detected then the software continues as normal. The next step of the control loop is to measure the speed of the motor. The motor speed is calculated as the number of encoder counts since the last interrupt. Figure 4-36 shows the code that calculates the speed. It is measured as the difference between the current position and previous position.

```
//read current speed
current_pos = POSCNT;// store poscnt to ensure calcs are done with the same value
int actual_speed = current_pos - prev_pos; //Calculate the speed
```

**Figure 4-36. Calculate the current speed of the motor**

Once the speed has been calculated then the reference speed is set using the received speed. The received speed is set from the communications. A speed profiler could easily be placed here to ensure the speed never jumps excessively. However, it was determined that the testing speeds are low enough to not need a specific profiling function. Therefore, the reference speed is set to the exact received speed. The code in figure 4-38 shows that the received value being set into the reference value. The received value is transmitted as a two's complement number, which allows values from -127 to 127. If the number is negative the code uses some simple maths to transform the binary value into a signed integer value.

Once the new reference speed has been set then the control loop checks whether the motor needs to stop or change direction. If either of these conditions occurs then the control software needs to clear the integral term. The integral term builds up to remove steady state error, thus if there is a change in direction the integral term must be reset to ensure the control loop works correctly. To check for this situation the software uses an IF statement looking for a reference speed of 0 or if the reference speed multiplied by the previous speed is less than zero. This maths works because if the speed was negative and is now positive, or vice versa, then the result will be negative. The same maths will return positive result if the direction is still negative or

still positive. This maths is very simple but works to detect the change in direction. This is shown in fig 4-37.

```
//set the reference speed
if(rec_speed > 127)//twos complement into signed int
{

    ref_speed = -1*(256 - rec_speed); //if its a negative number
}
else
{

    ref_speed = rec_speed; //else its a positive number anyway
}
```

**Figure 4-37. Convert from twos complement and set the reference speed**

```
// check for stops or changes in direction?
if(ref_speed == 0 || ref_speed * prev_speed < 0)
{
    I = 0;
}
```

**Figure 4-38. Code to detect a change in direction**

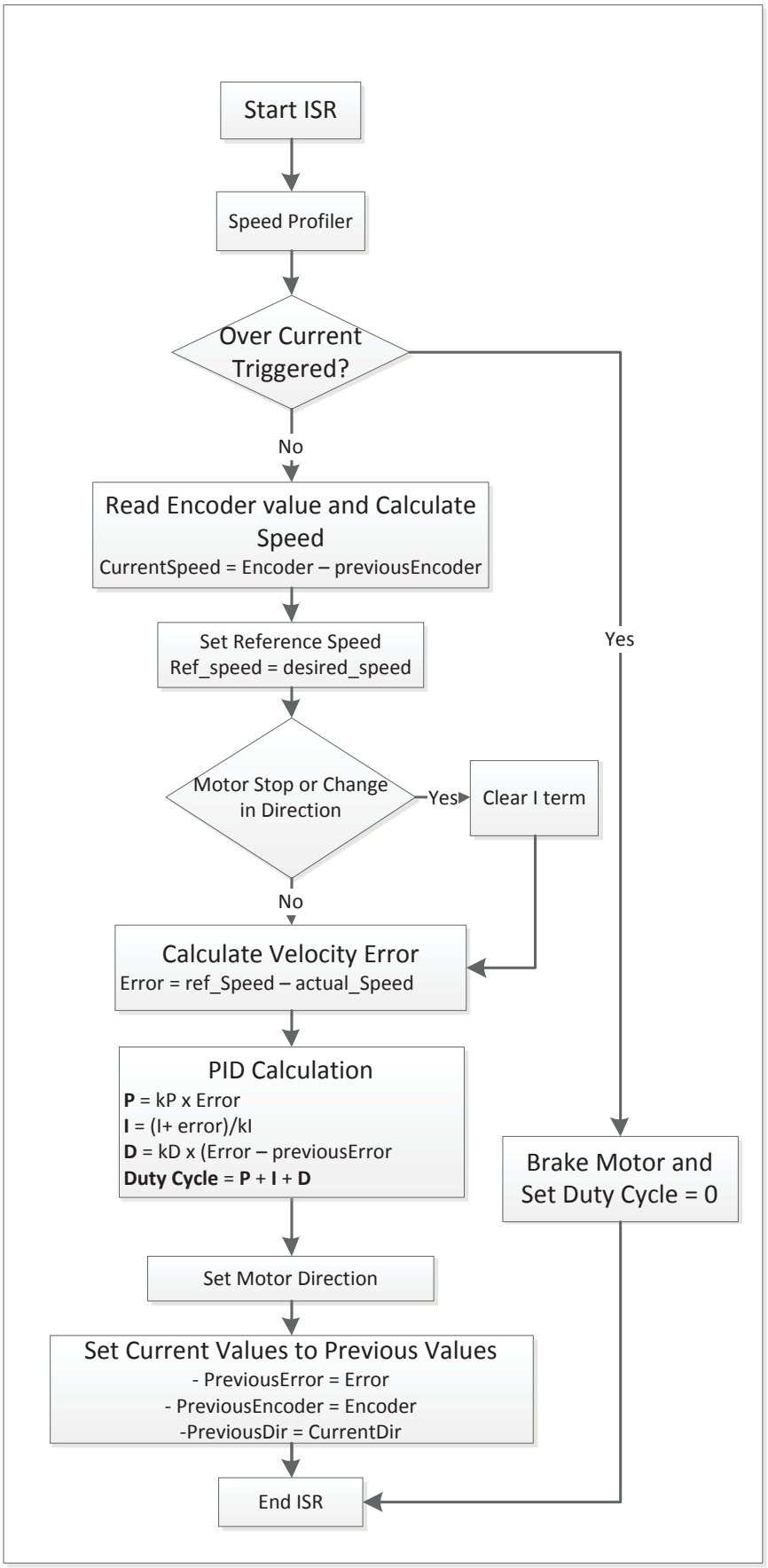


**Figure 4-39. Flow diagram of the control loop**

The next stage of the control loop is to calculate the error between the actual motor speed and the reference speed. The error then allows us to use the PID equation to calculate the control effort required to drive the motor. Equation 1 shows the standard equation used for this control loop.

$$Duty = kP \times e(t) + \frac{1}{kI} \times \int_0^{\infty} e(t) + kD \times \frac{de(t)}{dt} \quad (1)$$

Equation 1 shows the code for the calculation of the error and the calculation of the control effort, or duty cycle. The constants kP, kI, and kD can be changed to create difference accelerations or mechanical reactions. The tuning of the PID loop was done with a simple method of setting all gains to zero initially, and then increasing each control constant until an acceptable response was achieved. The acceptability of the response was determined by comparing the results to predefined metrics. The metrics are defined as less than a certain value; this helps to eliminate poor control constants quickly. The criteria for response time, i.e. total time taken to reach the set speed, was defined to ensure that the system would respond quickly, but still allows time for the user to react if the new speed were to cause an accident. The dead time criteria, i.e. time before a physical response occurred, was simply defined as less than half the response time. Overshoot was determined as less than 10% of the maximum speed, which is 30 counts per control loop period, Ts, to minimize the chance of over accelerating. Finally, the steady state error was defined as 1 to ensure the system was as close as possible to the set speed.

```
//calculate current error
signed int error = ref_speed - actual_speed; //Expected speed

//Calc PID values
P = kP * error; //proportional term
I = (I + error)/kI;//Integral term
D = kD *(error - prev_error); //Derivative term

//Calculate control effort
duty = P + I + D; //sum for control
```

**Figure 4-40. Calculate the error and the control effort**

Once the control effort has been calculated, and stored into the duty variable, the direction can be determined. The sign of the error and the PID maths will create either a positive or negative control effort depending on the direction the motor need to be driven. By checking the sign of the duty variable we can determine which direction the motor should be driven. The code in figure 4-41 shows how the direction is derived from the sign of the duty variable.

```
//set direction
if(duty == 0 )
{
    stop();
}
else
{
    if(duty > 0) //motor must go forward
    {
        rec_dir =0;
        forward();
    }
    else //else it needs to reverse
    {
        rec_dir = 1;
        reverse();
        duty = abs(duty);
    }
}
```

**Figure 4-41. Set the direction of the motors based on the sign of the control effort**

The control loop implements a simple speed limiter as well. This code is in place only for testing purposes to prevent an error causing the robot to crash or damage a subsystem. The software checks the current speed versus a threshold and if the current speed has exceeded this value it stops the motors, sets the duty cycle to zero, and the clears the start variable; which stops the control loop.

The penultimate step of the control loop is to transfer the value of the duty variable to the PWM generator. This will change the duty cycle that is being output on the I/O pins. Once the PWM out has been changed the values of error, position, and reference speed are stored into previous value variables to be used in the next control loop.

### *D. ADC interrupt*

The ADC interrupt is designed to take the analog signal, that represents the current flow of the motors, and convert it to a digital form. The data sheet of the current sensor shows that a level of 3.5V corresponds to 10A flowing through the sensor [55]. A power supply was used to simulate this and the ADC reading monitored. Using this it was determined that a reading of hex C0 corresponded to an overcurrent situation. Figure 4-42 shows the code used to detect this situation.

```
void __attribute__((__interrupt__)) _ADCInterrupt(void)
{
        int k;
        k = ADCBUF0; //read adc bufer
        amps = k; //store current draw in global variable
        if(k+0x80 > 0xC0+0x80){ // Over 3.5V?
            over_current = 1;   //set overcurrent to 1
        }
        else over_current = 0;
        IFS0bits.ADIF = 0; //clear interrupt flag
}
```

**Figure 4-42. ADC interrupt service routine**

The code simply compares the current reading with this predefined threshold and sets the flag if necessary. The software can be changed so that the flag is either set permanently or will be reset once the overcurrent situation has passed. Currently if the current draw drops to acceptable levels the flag will be reset and control resume.

## 4.7. MOTHER ROBOT TESTING

The Mother robot testing was designed to test the basic operations of all the software systems as well as testing the efficacy of the motor control algorithm and hardware. The motor controller tests were designed to tune and then test the motor controller's ability to maintain a specified speed. These tests were undertaken in a controlled environment and helped tune the control constants of the PID loop. The software tests were aimed primarily at testing the developed systems and ensuring that they worked as expected.

### 4.7.1. MOTOR DRIVER AND CONTROLLER TESTING

The Motor driver and controller testing was undertaken to check the control ability of the control loop, and the reliability of the motor driver and controller subsystem. The results from this test were presented at the 5th International

conference on Automation, Robotics, and Applications 2011 [9]. The test was run, initially, using a development board and off the shelf motor driver system. This initial test was run to check the effectiveness of the control loop. Figure 4-43 shows the development board, motor driver, and the Mother robot.

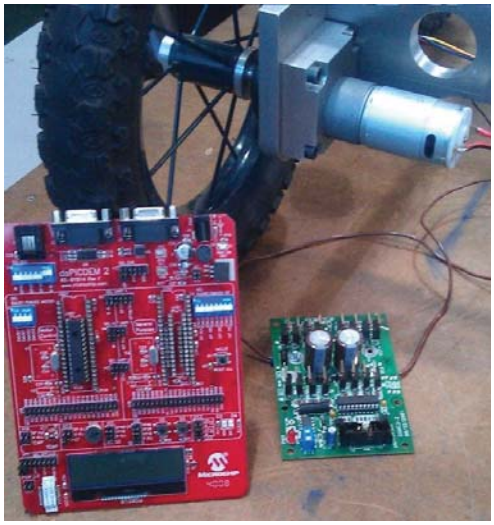

**Figure 4-43. Equipment used in the initial motor subsystem testing**

The test was run with the wheel spinning freely with no load. Specially developed software was run to send the controller a set speed, and record the current speed of the motor. The controller software was edited to return the current speed of the motor, via the RS232 port, once a control cycle. This would provide the results from the test. Figure 4-44 shows the software developed for testing the motor.

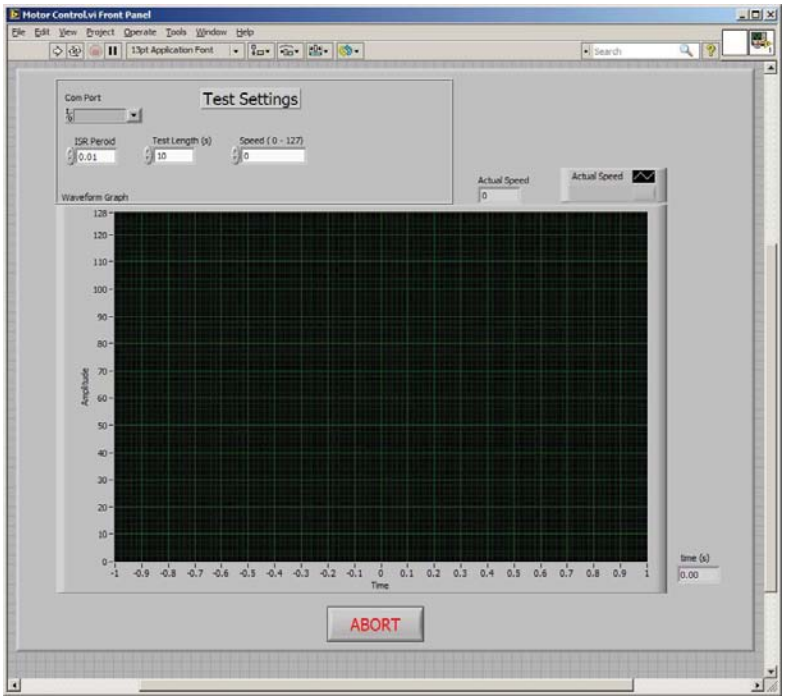


**Figure 4-44. Screen shot of the LabView test software developed**

The test was designed to check the motor response with changing PID control constants. With this in mind the motor was tested at a range of speeds, with a range of control constants. The results were recorded within the testing program could then be analysed.

The initial results were promising. The motor was responding well and reached the set speed with little steady state error. Figure 4-45 shows the response of the motor with the control constants of kP = 6, kI = 2, and kD = 1. The blue line shows the set speed, starting at 0 with a step change of 10 at 2s. While the red line shows the measured speed of the motor. There is a clear dead time between the change in the set speed and the first response of the motor. The length of this time can be attributed to the time between control loops.

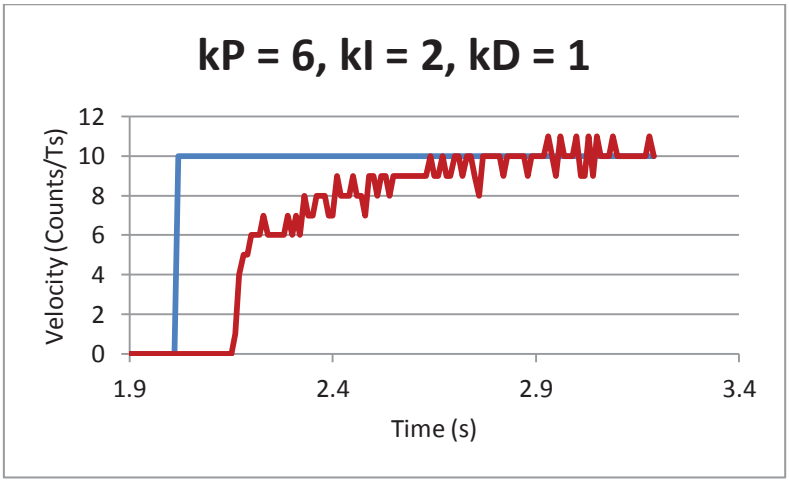


**Figure 4-45. Graph of motor response for gain constants kP = 6, kI = 2, kD = 1**

While the control shows promise of working the tuning of the current constants can be improved. The test was run over a range of values and the results collected. Table 4-2 shows a selection of the results and the results broken down into important categories. The response time refers to the amount of time before the measured speed first reaches the set speed. While the deviation is measured as the amount the measured speed deviates from the set speed after reaching the set speed.

| kP, kI, kD | Dead time (s) | Response (s) | Deviation |
|---|---|---|---|
| 6,2,1 | 0.14 | 0.62 | 2 |
| 6,,31 | 0.15 | 0.34 | 2 |
| 6,3,2 | 0.15 | 0.39 | 2 |
| 8,2,1 | 0.14 | 0.55 | 1 |
| 8,3,1 | 0.14 | 0.31 | 1 |

**Table 4-2. Specific results from the motor tuning tests**

These values allowed us to show that the best constants for kP,kI, and kD were 8,3, and 1 respectively. The response curve from these values can be seen in figure 4-43. Table 4-3 shows the results of these control constants at various speeds.

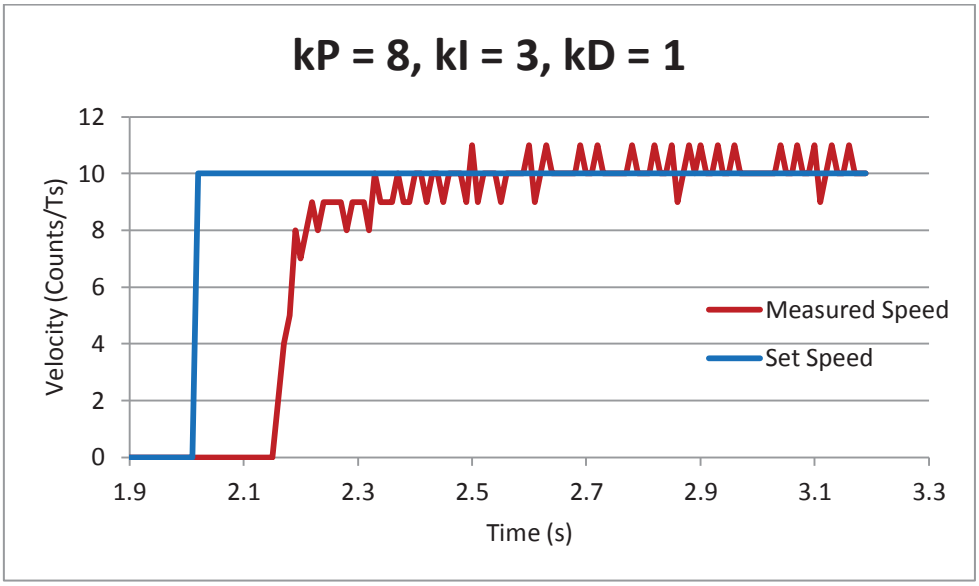


**Figure 4-41. Graph of motor response for gain constants kP = 8, kI = 3, kD = 1**

| Reference Speed | Dead Time(s) | Response(s) | Deviation |
|---|---|---|---|
| 2 | 0.36 | 0.38 | 1 |
| 5 | 0.16 | 0.4 | 1 |
| 10 | 0.14 | 0.31 | 1 |
| 15 | 0.09 | 0.56 | 2 |
| 20 | 0.15 | 0.65 | 2 |
| 25 | 0.14 | 0.36 | 2 |

**Table 4-3. Test results for control constants at a range of speeds**

These tests proved that the control loop was effective at controlling the motor to a set speed. However, during later of both the Mother robot and the final controller and driver systems issues were noted with the original software. The control loop was altered to the form shown in 4.3.2 Motor Control Software.

The final versions of the motor control subsystem have been tested for reliability of the hardware and for the ability to control the motor to the correct speed. Further testing is needed to tune the amended control loop for optimal control.

### 4.7.2. MOTHER SOFTWARE BASIC TESTS

The single board computer constitutes the major on board intelligence of the Mother robot. At this stage of the development it needs only to receive communications, interpret them, and pass on information as necessary. The major functions that the Mother needs at this point are communications between the base station and itself, via the ZigBee network; the ability to transmit control data to all the motors or a single specific motor; and the ability to receive control data from the base station and translate this into a usable form for driving the Mother. Simple tests were performed to show that these functions had been implemented and were working as needed.

#### *A. Communications*

Testing the communications is very straight forward. The base station and Mother simply needed to show that the communications between them worked. This test was undertaken during the early stages of the software to ensure that the ZigBee module was working successfully. The systems were both set up on a table and an arbitrary data packet sent from the base station to the Mother and vice versa.

Figure 4-47 shows the received data, highlighted in red, and figure 4-48 shows the transmitted data, highlighted in green. The received data matches the data transmitted and thus the communications software and hardware are functioning. This test was repeated in reverse to ensure that two way communications was possible and was also successful.

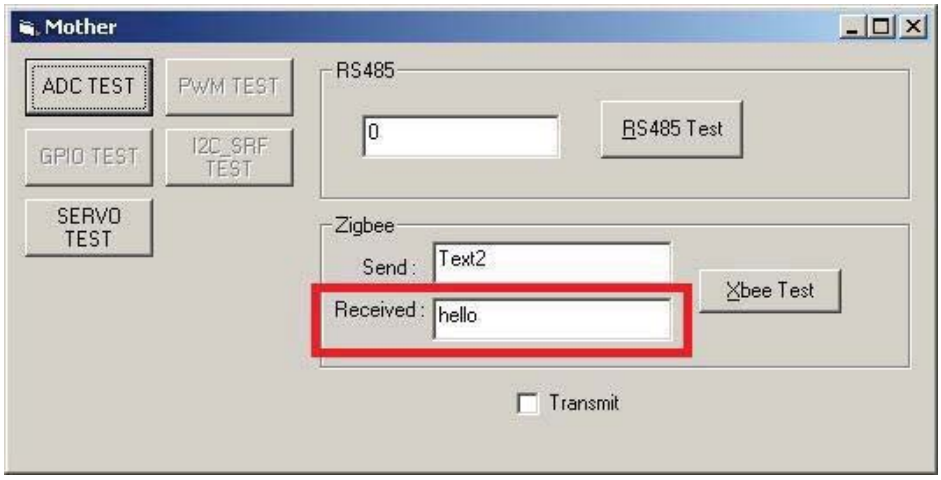


**Figure 4-47. Received Data highlighted in red**

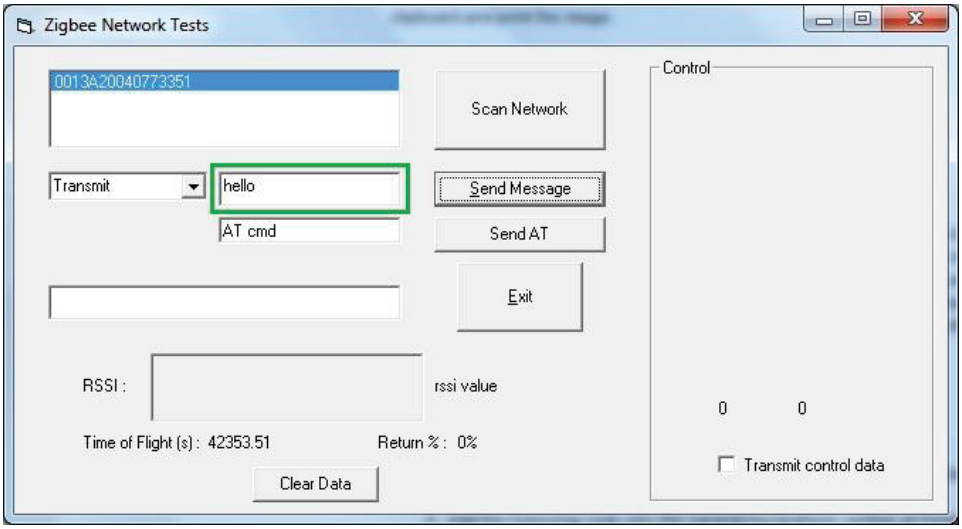


**Figure 4-48. Transmitted Data highlighted in green**

This was a simple test to check that the communications worked and that the system could receive a non-erroneous message. The test showed that this communication was possible between base station and the Mother system

### *B. Motor Driving*

Another basic function the Mother needs is to be able to drive one or all four motors. The Mother needs to be able to send data to each motor and ensure that the correct control data has been sent to the correct motor. To test this, the Mother was sat on top of two chairs so that the wheels could spin freely, but so that the Mother would not move.

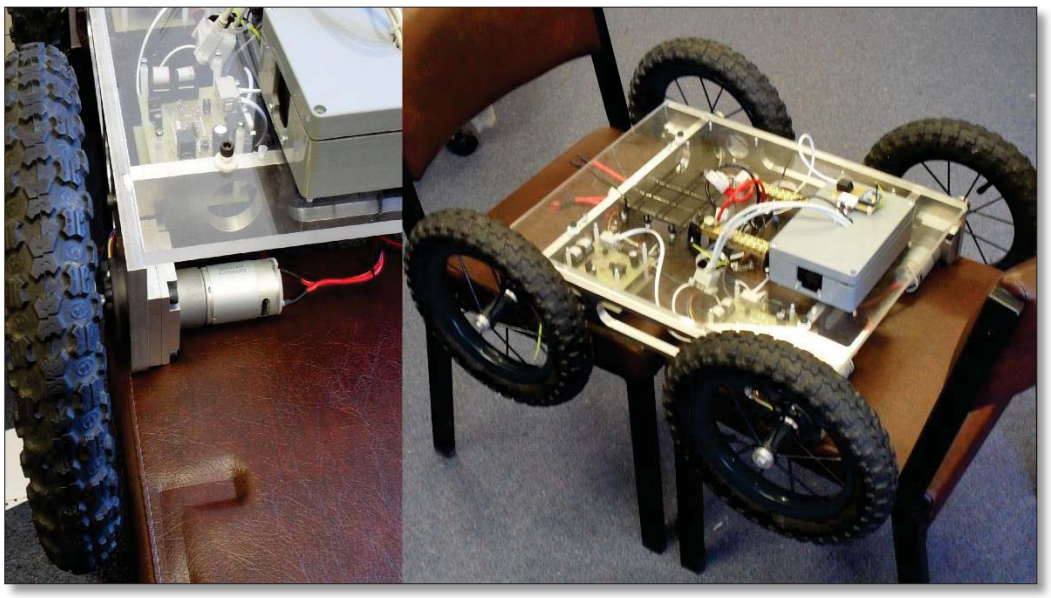

**Figure 4-429. Close up of the suspended wheel (left) and a view of the Mother sitting on the chairs for testing (right)**

Figure 4-49 shows the way the Mother was set up for this test. The computer was given specific control speeds and target motors to send the data to. The test simply showed that the software could send data out on the RS485 channel that could be interpreted by the motor controllers. Figure 4-50 shows the basic software that allowed us to choose which motor would receive that the data.

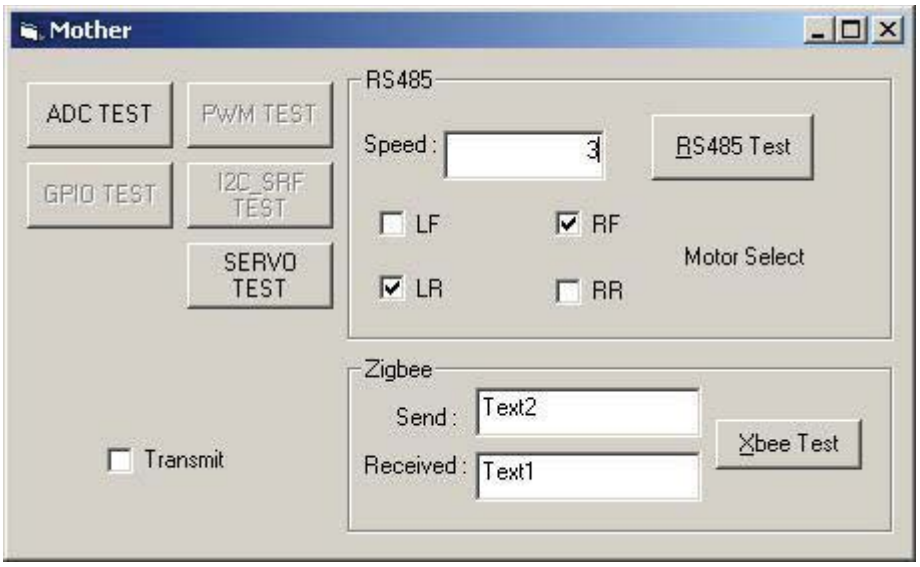


**Figure 4-50. Motor Testing software with individual motor select functionality**

The software utilises check boxes to decide which motors would receive information and which would be ignored. This provided a simple method for testing if individual communications were possible. The tests proved that each motor could be individually communicated with or different data sent to all four. This showed that the communications with the motors was working as intended.

### *C. Control Data*

The test to confirm that control data was being sent and interpreted successfully was conducted in the same manner as the communications test. The Mother software was edited slightly to ensure that it displayed the received data and the interpreted data. While the base station software showed the data in the form of the speed and direction. To check that this functionality was working the control was manipulated to give a wide range of values and progress bars were used to display the numbers. The progress bars show 50% full when the data is 0 and grow or shrink depending on the value of the control data. The data was transmitted and the Mother interpreted the value and showed the interpreted values in the same manner. Figure 4-51 shows the software on the Mother with data being received and figure 4-52 shows the base station software. Both figures show the same values. The data shown in figure 4-52

also shows the received string of “L3R253”, which shows the transmitted values in twos complement, as well as the converted data on the right with the progress bars.

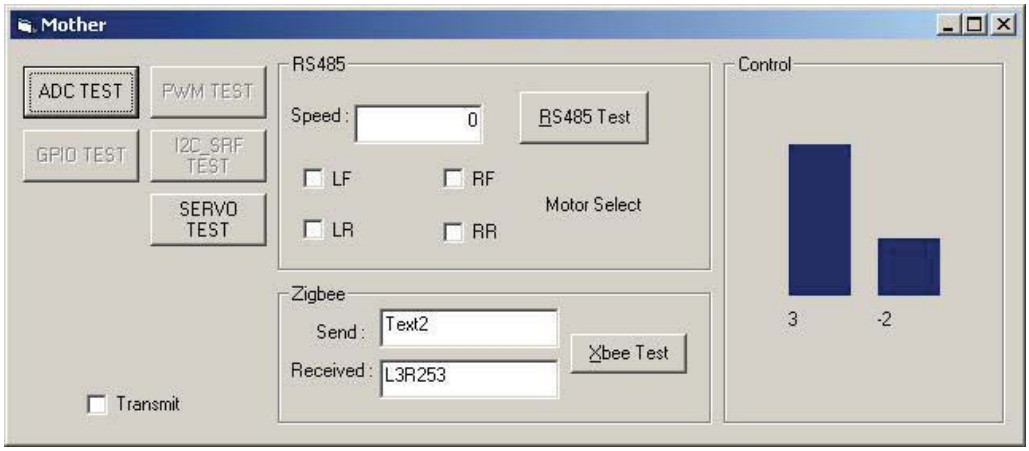


**Figure 4-51. Mother software receiving and interpreting the control data**

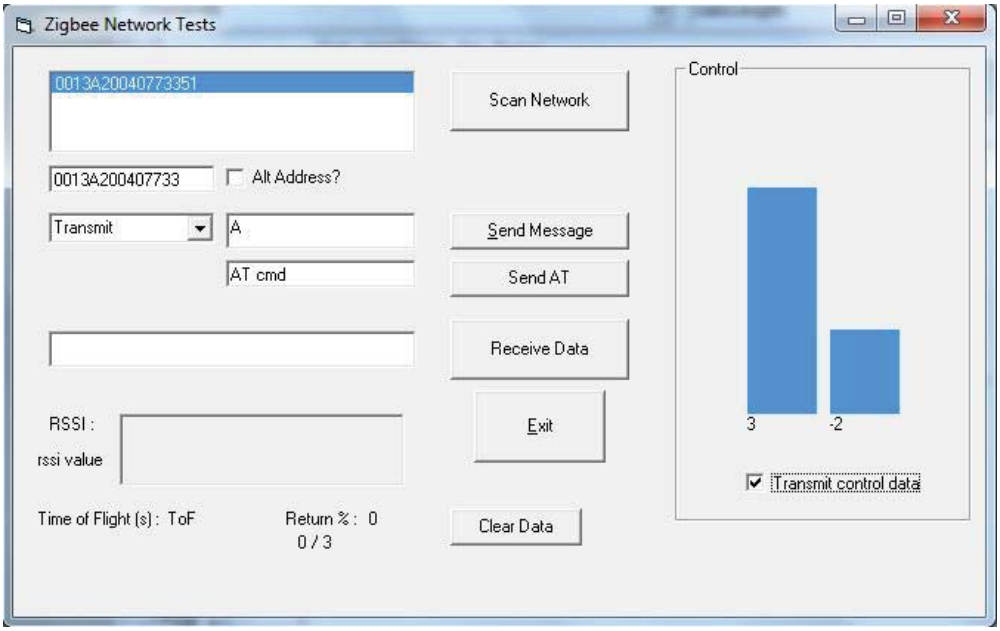


**Figure 4-52. USAR Base Station display of the control data being transmitted**

The test was conducted and the values updated on both screens in approximately real time. This showed that the data was being transmitted and that any change in the data would be relayed to the Mother by the communications.

# 5. Daughter Node System Design

The Daughter robot is designed as a small, disposable robot. Its small size will allow it to penetrate much deeper into the disaster zone and give a much more in-depth view of the situation. The disposability will make it ideal for search and rescue operations, making the primary focus the recue or survivors without the worries of retrieval of expensive robots. The Daughter will be able to undertake different operations dependant on how they are configured. Different sensors will provide operators with more, diverse information which can only help improve decision making in such a stressful environment. These robots, once fully developed, will hopefully be able to autonomously search and will be able to much more than just detect changes in the environment. Figure 4-1 shows the functional block diagram of the current version of the Daughter robot. At this stage calling it a robot is misleading. It acts as a wireless sensor node and a wireless communications relay. These features are the core of the system and show how the system can be implemented in a wide range of situations. For example, if these nodes were built with soil moisture sensors and pH sensors they could be deployed within a vineyard and keep the farmer appraised of the state of the soil. This is but one application of the system and the other potential applications are limitless.

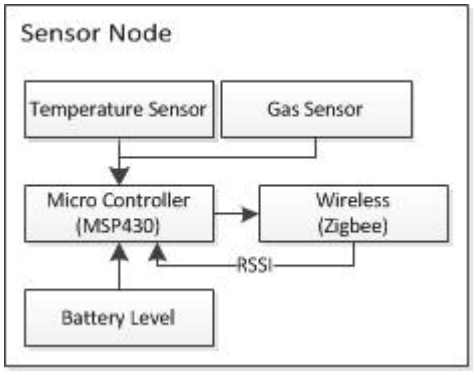


**Figure 5-1 Functional block diagram of the Daughter nodes**

The current design of the Daughter nodes, or Daughters for short, will only carry a limited number of sensors and a wireless communication module. Figure 5-1 shows these sensors as a temperature sensor and a gas sensor. These were chosen as another common implementation of this system could be factory fires. The system can measure the temperature and monitor combustible gas levels and warn fire fighters if these increase. These sensors will also be very useful in an urban search and rescue environment for the same reasons. The Daughter is not only monitoring these values,

but internally considering them. The node uses software to check the value of the sensor reading against a pre-set threshold. If the reading exceeds this threshold the user is alerted. This is helpful as the user is not required to poll each node asking for information about the environment. Any dangerous situations are flagged and sent automatically. However, a high reading indicates danger has occurred, not that it may be about to occur. To help warn the users of impending danger the Daughters also monitor the changes in the sensor readings. A large enough change in the readings also forces the system to set a flag and warn the user. This helps prevent rescuers being trapped in a building that starts to burn, or farmers losing crops because the pH has dropped past a point of no return.

## 5.1. DAUGHTER NODE HARDWARE

The Daughter node is a basic system that consists of a microcontroller board, a sensor board, and a battery. These three simple components create a very flexible wireless sensor node platform. By changing the sensors, or adding more, the types of environmental factors that can be measured are changed. These changes affect the operation of not only the Daughter node but the system as a whole. By measuring different factors the Daughters can be used in different ways to complete different tasks. The hardware boards connect in a simple manner making swapping one sensor board for another very simple. This gives great flexibility to the design and makes changing the operation of the Daughter a simple task.

### 5.1.1. TEXAS INSTRUMENTS LAUNCHPAD

The Daughter Node was designed with low cost and ease of development in mind. To facilitate rapid prototyping, flexibility in the hardware, and low production cost it was designed as a attachable board to the Texas Instruments Launchpad.[60]. The Launchpad provides a very small form factor development board complete with USB connector for programming and debugging. Figure 5-2 shows the Launchpad. The board provides an SPI to USB convertor, for implementing communication; Power regulation, so the board can run off power from the USB port; Access to the port pins, for developing external circuitry; a push button, for testing and development; an external crystal oscillator, for improved clock speed and accuracy; and more.


**Figure 5-2. Texas Instruments Launchpad Development Kit**

The Launchpad is broken into two main parts: Emulation and Development. These two segments are connected by five jumpers. These jumpers include power, communications, and debugging connections. By removing the power jumper the user must supply the board with its own external power source. This is ideal as it allows the user to provide a power source that is not restricted to the USB connector. The reset and test jumpers are used for debugging and programming. Reset is important as it allows the programming software to reset the controller when needed. The Tx and Rx jumpers are only needed for communication with the SPI module on the board and for our microcontroller are not necessary. This is because the micro controller used has specific UART hardware.

The Launchpad can be implemented with a wide range of microcontrollers from Texas Instruments value line range. The small size of the microcontrollers means code size and specific hardware modules are limited, but for the use in a sensor node they are more than apt. The specific microcontroller chosen for the Daughter nodes was the MSP430G2553 [61]. This microcontroller provides all the hardware peripherals required for the Daughter.

### 5.1.2. GAS SENSOR

The gas sensor used within this system is a liquefied petroleum gas sensor [62] or LPG sensor. However the name is misleading. The sensor can measure the levels of a wide range of combustible gases. The sensor, as shown in the datasheet, has a higher sensitivity to LPG, thus making it most effective at measuring this gas. The fact that different gases can cause the output value to change is an issue if trying to measure the

exact level of gas present. Because we have no intention of measuring the specific quantity of combustibles in the air it is not an issue. Our premise is that the sensor can detect a gas that may cause fire or an explosion. By detecting the presence of any number of combustible gases we can ensure that this is met with a broader range, at the expense of accuracy. Figure 5-3 shows an image of the LPG sensor installed on a prototype Daughter module. The sensor is the large metallic dome in the centre of the image.

**Figure 5-3 LPG sensor installed on a Daughter node prototype**

The major issues discovered during the initial testing phase were that of heat dissipation and power usage. The sensor utilises an internal element to heat the air inside to a specific temperature so that the sensor can make accurate readings. This element outputs a large amount of heat and greatly affects the temperature of the air nearby. This is a problem as the temperate sensor is within close proximity. The added heat from the LPG sensor could offset the temperature sensor reading and may cause fluctuations that are seen as dangerous or irregular. This excess heat output comes at the cost of large power consumption. The gas sensor draws approximately 150mA while operating. This large current draw will drain the battery very quickly and reduce the operating life of the node. As these nodes will be deployed in situations that require long life this is unacceptable. To overcome this issue a MOSFET was installed to switch the gas sensor off and on as needed. Figure 5-4 shows the schematic design used for the gas sensor and the MOSFET that controls it.

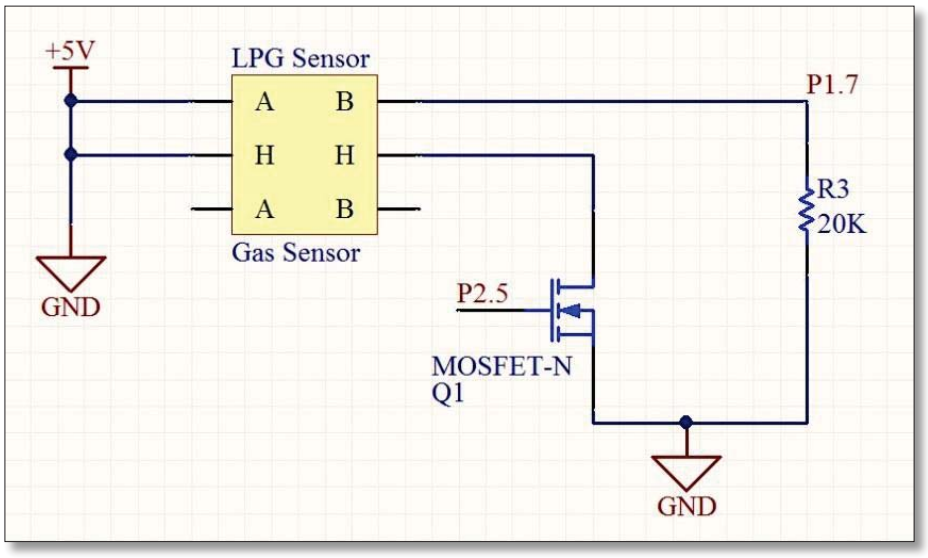


**Figure 5-4 Gas sensor and power switching MOSFET schematic**

The final major electronic system for the Daughter is the voltage regulators. There are two low drop out voltage regulators working on the Daughter node. The first is a simple 3.3V regulator implemented to provide a stable voltage for both the ZigBee and the MSP430 microcontroller. The second provides a regulated 5V supply for the gas sensor. As the gas sensor, and more importantly the thresholds for the software, has been calibrated using 5V across both the element and the sensor it is critical that it be run from this voltage. The datasheet values for sensitivity and output values are all taken using the 5V input. By supplying a 5V reference we can ensure that the sensor will run correctly and provide accurate readings.

### 5.1.3. TEMPERATURE

The temperature sensor used in the Daughter nodes is an LM94022 multi gain analog temperature senor from National Semiconductor [63]. The LM94022 provides a simple analog signal output that is easily interfaced with the current hardware of the Daughter node. This is ideal because it is identical to the gas sensor, making a consistent interface across all sensors. This standardisation makes programing the system simpler. This sensor also provides a multi gain output. Which means the output gradient can be changed to adjust the range of output values. Figure 5-5 shows the temperature sensor installed on the Daughter board.

**Figure 5-5. Close up of the temperature sensors (circled in red)**

The temperature sensor is very small in size, with a footprint of less than 2mm squared. This makes it very easy to incorporate into the design and at under $3 per component it is very good value for money. The small size comes with the added bonus of being light. While this is not an issue in the current level of research when this research reaches commercial viability or operational viability this will be incredibly helpful. This is because when all the Daughters are being transported on the Mother any weight reductions will be beneficial to improve battery life. This will allow further exploration or the ability to carry a larger number of nodes.

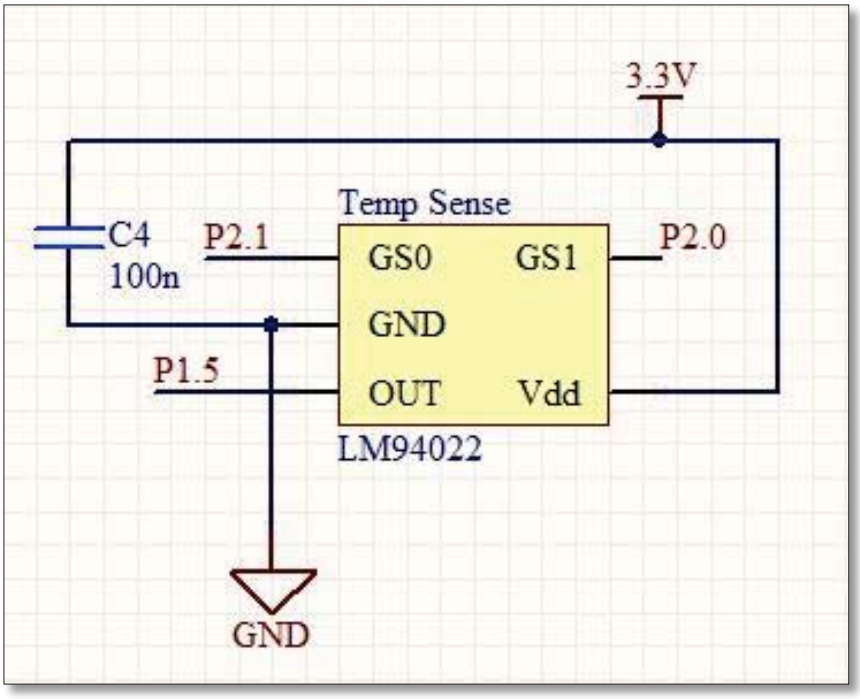


**Figure 5-6. Schematic of the temperature sensor**

### 5.1.4. BATTERIES

The batteries selected for the Daughter nodes were Lithium Polymer batteries or LiPo batteries. The decision to use LiPo batteries was based primarily on their size and capacity.


**Figure 5-7. Lithium Polymer battery used with the Daughter node**

The batteries selected were two cell batteries; this means the battery itself consists of two lithium polymer cell rated at 3.7V nominal. This means the nominal voltage of the pack will be 7.4V. 7.4V was chosen because the gas sensor requires a 5V source to operate and regulating from 7.4V is very simple. However, 7.4V is not a constant voltage, these batteries range from 8.2V, at full charge, to 6V at the maximum allowable discharge. As is commonly known with LiPo batteries each cell should never be discharged below 3V to avoid damaging the battery. Figure 5-7 shows one of the batteries purchased and used.

This battery was low cost at under US$5 and weighs a mere 59g [64]. They have a 1000mAh capacity. This also provides information about the discharge The ‘C’ rating on the battery refers to how much current can be supplied by this battery. To figure out the maximum discharge current one multiples the capacity by the discharge value, for example this battery can safely provide 15A of continuous current, as it is rated at 15C. This exceeds the required 250mA for the Daughter nodes and means we will only damage this battery if a major short occurs.

### 5.1.5. ZIGBEE MODULE

The ZigBee module used on the Daughter nodes is an Xbee Series 2 module [65]. The module boasts better power consumption and increased mesh network capabilities over the series 1. The price of these modules is low enough to make them affordable.

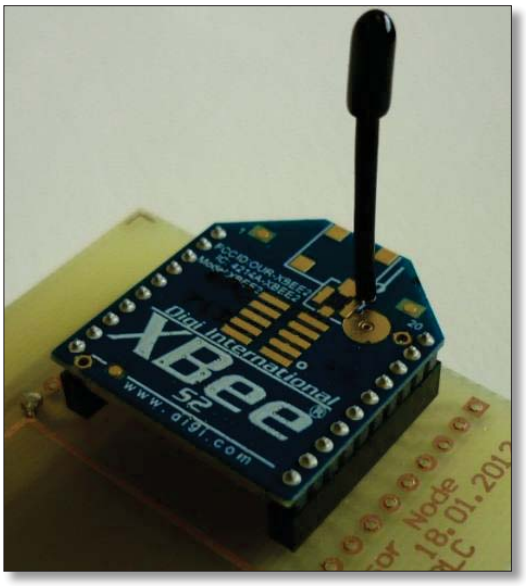


**Figure 5-8. The ZigBee module installed on a Daughter node**

This module can handle both AT and API mode communications and has the full range of feature of the ZigBee protocol. These features have been covered in chapter 3.

## 5.2. DAUGHTER NODE SOFTWARE

The software is required to complete two main functions. These functions are measuring the sensor values and providing communications with the user. The first function requires the implementation of an analog to digital converter module to read the sensor outputs and convert them to an integer value. Once the value is accessible from the software it must be monitored and compare with predefined thresholds and rate of change metrics. The second function, providing communications, is required to allow the daughter to send and receive information with the user. This function requires conforming to the packet requirements of the ZigBee protocol.

Figure 5-9 shows the basic flow of the daughter software. This flow diagram shows the initialisation and the main functions which are looped over indefinitely. These are not the only sections of the software, which relies heavily on the use of interrupts to trigger major events and functions. To successfully complete these tasks in the correct

order without making errors there is a flag system in place so that certain tasks must be completed before others begin. This helps keep the system structured and operating in a way that is both predictable and meets the requirements. Each of the functions will be explained in following sections.

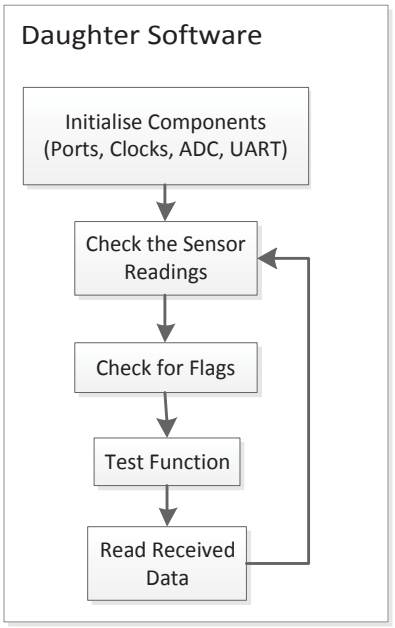


**Figure 5-9. Flow diagram showing the basic operation of the Daughter software**

### 5.2.1. Initialisation and the Main Loop

The First initialisation is the input output pins and the clock source. The code utilises the internal digital clock source set to 1MHz. This clock will be used for all other modules within the system. Once the clock source is set up the port pins are the next step. The digital port pins are initialised using three registers. Firstly the port direction register, PxDIR, defines whether the pin is an input or an output. Secondly, the output value of the register, PxOUT, if it is configured for output. Finally, the port resister register, PxREN, defines if the pin is pulled high or low by internal resistors. Figure 5-10 shows the code required to set one port up with the correct values.

```
//Digital IO
P1DIR = 0x41; //Set P1.6 and P1.0 to output, rest to input
P1OUT = 0x00; //Set Ouput to 0 on all pins , set P1.3 to 0 for switch
P1REN = 0x80; //Set P1.3 pull up to enabled, so switch pulls low
```

**Figure 5-10. Digital Input/output setup code**

Port 1 is set to provide pins P1.0 and P1.6 as outputs and the rest as input pins. This is done by the first line of code in figure 5-10. The output value of the pins is set to low for all pins on the port. The only pins affected by this are P1.0 and P1.6 as they are

digital outputs. The only pin that is set to be pulled high by an internal resister is P1.3. This pin is connected to the on board push button. By using the pull up resistor, the input pin will register high when the button is inactive and will register a low when the tube ton is pressed. This allows us to easily read the state of the button within the software and use it for testing purposes.

First operation is to implement all the modules required for operation. The first of which is the universal asynchronous receiver transmitter, or UART. This module will provide an interface between the ZigBee module and the micro controller. This interface can be set up in a wide range of configurations, but for simplicity it has been configured to run at a baud rate of 9600bps, 1 stop bit, and no parity. This is a very standard configuration. Figure 5-11 shows the code used to implement the UART module.

```
//Initialise the UART module for communications
void init_UART(void){

    UCA0CTL1 = UCSWRST; //disable the UART
    UCA0CTL0 = 0;       //set control register 1
    UCA0CTL1 = UCSSEL_2; //set control register 2
    UCA0BR0 = 104;       // Set up Baud Rate to 9600bps
    UCA0MCTL = UCBRS_1 + UCBRF_0;
    P1SEL = BIT1+BIT2;
    P1SEL2 = BIT1+BIT2; //Set up output pins
    UCA0CTL1 &= ~UCSWRST; //Reset UART
    IFG2 &= 0xF7; //clear TX interrupt flag
    IE2=UCA0RXIE;  // enable RX interrupt

}
```

**Figure 5-11. Initialisation code for the UART module**

The next major module to initialise is the analog to digital converter, or ADC. The ADC is needed to measure the analog signal and convert it to a digital value to be interpreted. The ADC module needs to read three separate input channels in order. The code shown in figure 5-12 shows the set up required for the ADC. The ADC is initialised to use the positive voltage rail as the reference for conversions. The other options set up during initialisation are to have the ADC only complete a single conversion when the ADC10SC bit is set high. This allows the ADC to be activated only when needed. The final step is to select the input channel. Input channel four is selected as this corresponds to P1.4. This pin is connected to the battery voltage measurement. The input channel will be changed after each conversion is completed to cycle through all

necessary channels. This is done in the interrupt which will be discussed in a later section.

```
//Initialise the ADC for sensor measurements
void init_ADC(void){

            ADC10CTL0 = SREF_0 + ADC10SHT_2 + ADC10ON + ADC10IE + REFON;
            // uses V+ as reference, ADC10ON, interrupt enabled
            // Sample Timer = /16
            ADC10CTL1 = ADC10SSEL_0 +  CONSEQ_0 + SHS_0;
            //Single Conversion, ADC10SC bit starts conversion, use ADC1
            ADC10CTL1 = INCH_4;
            //Select A4 as input initially
}
```

**Figure 5-12. The initialisation code for the ADC module**

The final module that is initialised is a timer. This is not essential but has been set up to limit the power drawn by the gas sensor. The timer is used to switch the sensor on and off periodically and only turn it on when a measurement is to be taken. The timer is initialised to use the auxiliary clock as its source. This clock is has been set up using the on board low speed clock source. This source runs at approximately 12kHz and can be used for peripheral hardware as needed. The timer is set up in continuous mode. This means that it will continue to increment its internal register till the register is filled and overflows. The register is 16bits wide giving a maximum count of 65,535. When this internal register overflows the timer interrupt occurs. Because the timer is incrementing at 12kHz and counting to 65,535 we can easily calculate that it will take 5.46s to reach this value. This knowledge will be used within the interrupt to create a timer that toggles the MOSFET for the gas sensor at predefined timer intervals.

```
void init_TimerA_PWM(void)
{
    TACTL = TASSEL_1 + ID_0 + MC_2 + TAIE;
}
```

**Figure 5-12. Timer A initialisation code**

Figure 5-12 shows the required initialisation code for the timer. The clock source, TASSEL, is set to use the auxiliary clock. The clock division, IDx, is set to divide the clock source by 1. MCx sets the mode of the counter, in this case to continuous mode. Finally, TAIE enables the timer interrupt.

### 5.2.2. FLAGS

The flag system used in the Daughter software is implemented to alert the system to specific events and control the flow of the software. The flags are all contained within one byte and can easily be set or cleared by simple Boolean operations. Figure 5-14 shows a simple situation in which a flag is set and another is cleared.

```
if(dataLen > 5 && dataLen == recFrame[2]+3)
{
    flags &= 0xEF; //clear recstart
    flags |= 0x20; //set data ready
}
```

**Figure 5-14. Clearing the recStart flag and then setting the dataReady flag**

The Boolean operations are simple and effective making use of basic Boolean principles. This allows the software to change the state of a flag without affecting the others within the byte. There are eight flags within the byte 'flags'. These flags consist of two flags for each sensor, one to show a rate of change event and one to show that a reading has exceeding a threshold. There is a flag that shows whether an API packet is currently being received and one to show that a packet has been completely received. The final two flags show the status of the battery voltage, one for two separate levels of voltage. The first voltage level shows that the battery has reduced below the drop out voltage of the regulator that supplies the gas sensor. Once this flag is set, the gas sensor cannot be relied upon to give accurate readings and is turned off. This not only prevents false positive readings from the sensor but helps prolong the operation of the Daughter. The other flag is set when the battery voltage drops below an acceptable level for the batteries. The batteries, as previously mentioned, are LiPo batteries. These types of batteries are not meant to be discharge below a certain voltage per cell. This second flag warns the system that the voltage is nearing this voltage. The flag is primarily for testing to avoid deep discharging the batteries and damaging them, but can also be used in the intended application. This flag would be useful to warn the user if certain nodes are about to become inactive or unreliable for communications. This would allow the user to replace them with new nodes or inform them that parts of the operational area will soon become unreachable by wireless communications.

### 5.2.3. INTERRUPTS

There are two very important interrupts that run within the Daughter software. They are the ADC interrupt and the Received Data Interrupt. They are required to measure the voltage and store it in a variable and build an API compliant packet from received data, respectively. These two interrupts are vital as they process incoming data from the world and format in such a way that the software can process it quickly and effectively.

#### *A. ADC*

The ADC interrupt runs every time the ADC finishes converting an analog signal into a digital value. Once this is done the data is stored within the ADC10MEM register. The primary job of this interrupt is to take this data and shift it into a variable that can be read by the rest of the software and clear the ADC busy flag so another measurement can take place. The advantage of the MSP430 development environment is that the flag for the ADC is cleared as soon as the interrupt is vectored to. This means there is no requirement to explicitly clear the flag, although it is good practice to do this anyway. The flag is explicitly cleared in the first few lines of the software. After the flag has been cleared the major functions can begin. This is broken down into five major functions, shown in the functional block diagram in figure 5-13

The first step is to read what the current measurement channel of the ADC is. This is because the Daughter can measure multiple input channels depending on the number of sensors, it needs to check what sensor is being read currently. This is done by checking the ADC control register value against predefined channel values. The second step is to use a series of IF statements check the channel and then set the mode variable based on the outcome. Figure 5-14 shows the code used to make this decision. The major advantage of this is that it can be expanded to accommodate any number of sensors. The other advantage of this system is that it checks the channel at run time and doesn't rely on a stored variable to keep track of the current channel.

The third step is to read the ADC register, which is done by simply taking the ADC10MEM data and storing it into a temporary variable. The fourth step is to take the temporary variable data and store it into a global variable associated with that input channel. This is to ensure the data is store in memory in such a way that when reading the sensor variable the most up to date data is always accessed.

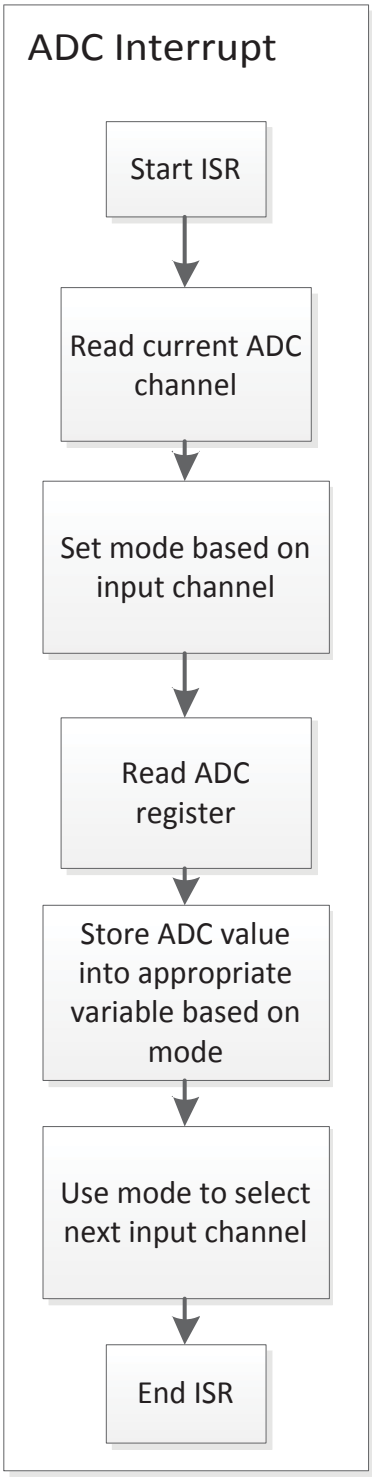


**Figure 5-13. Flow diagram of the ADC interrupt of the Daughter software**

```
//Set the mode ( which channel variable) for data discrimination
if((ADC10CTL1 & 0xF000) == INCH_4){ mode =0; }
if((ADC10CTL1 & 0xF000) == INCH_5){ mode =1; }
if((ADC10CTL1 & 0xF000) == INCH_7){ mode =2; }
```

**Figure 5-14. Channel discovery code**

```
//store the raw sensor data
if(mode == 0) {BattLvl = raw;} //A4 , P1.4, Batt Level
if(mode == 1) {SensorB = raw;} //A4 , P1.4, Temp  f
if(mode == 2) {SensorA = raw;} //A5, P1.5, Gas
```

**Figure 5-15. Code for storing the raw data into the appropriate variable**

Figure 5-15 shows the code to store the raw data into the appropriate variable. This is done, again, by IF statements. The if statements use the mode variable previously set when checking the input channel. This means the data that has just been measured will always be stored into the variable associated with the channel. This system has a drawback that the variable names are specific to the current iteration of the system and have to be changed within the software, but this can easily be changed to have plug and play sensors if necessary.

The final step of this interrupt is to change the input channel. This is done using the current mode value and setting the ADC control register channel bits to the next channel. By doing this at the end of the interrupt the software will only change input channels once it has completed all ADC functions with the current data. This provides reliable data and means data will always be up to date and stored in the appropriate variable. This action is completed by using similar code as before. Figure 5-16 shows how the mode is used to change the channel to the next within the loop. This also allows the user to quickly change the order of the sensor readings within the software.

```
//change the measurement channel, based on previous channel
if(mode == 2) {ADC10CTL1 = INCH_4;} // Check batt
if(mode == 1 && P2OUT & 0x20) {ADC10CTL1 = INCH_7;} // Check
if(mode == 1 && !(P2OUT & 0x20)) { ADC10CTL1 = INCH_4; } //Sk
if(mode == 0) {ADC10CTL1 = INCH_5;} // Check Temp
```

**Figure 5-16. Code for cycling through ADC input channels**

The statements which set the Gas sensor as the next input have the additional condition that the Gas sensor MOSFET, port 2 pin 5, must be high or it will be skipped. By including this into the cycling it will ensure the gas sensor is only read while the element is on. This avoids taking measurements when the element is not on. Discarding of data that is taken before the element has been on for a long enough period of time is controlled within the sensor reading methods. Overall the ADC interrupt works well, it takes the raw data and stores it in the proper variables for later analysis and ensure that the input channels are cycled through to measure all sensors. It also takes care to

not cycle to a channel that is not valid, in the case of the gas sensor, and prevents taking excessive measurements that would be of no use.

### *B. Received Data Interrupt*

The Received Data Interrupt, or Rx interrupt, occurs when a byte is received by the UART of the microcontroller. This interrupt is vital to both the Daughter and the Network. The Daughter requires this interrupt to be able to receive packets and react to requests or commands. The Network relies on the ability of the Daughter to receive data and react to keep information up to date. This feature is not required for overall network communications as routing is done internally on the ZigBee module but would limit the functions of the network if removed. This interrupt acts as a method of building the individual bytes received into a proper formed API packet. API packets will be explained in detail in chapter 5, but for now it is only important that there is a very specific format to these packets. Figure 5-17 shows the basic operation of this interrupt. API packets have three important bytes that help with formatting. The first is the start delimiter. The start delimiter is the tilde character, '~' , and indicates the start of a packet. The interrupt looks at the received byte and if it is the start delimiter stores it in the recFrame array at index zero. This starts the API frame. Once an API frame has begun, the recStart flag is set to show that all bytes from now form the body of the current packet.

The interrupt occurs as each byte is received. So a global counter, dataLen, is incremented each time the interrupt runs to keep track of which index of recFrame should hold the data. The data is inserted into the recFrame array as it is received. Each time a byte is received the interrupt checks the dataLen variable against the expected packet length. The expected packet length makes up the other two important bytes of the API packet. They are the second and third bytes, respectively, and convey the length of the packet in bytes, excluding the start delimiter and length bytes. Using this data we can tell how long the packet should be and store data into the recFrame array until the expected length is reached. Once this is completed then the API frame is complete and can be read. To show that a packet is ready to read the dataReady flag is set. The recStart flag is cleared and thus ensures that the interrupt knows that it must find a start delimiter before storing data into the recFrame array. This prevents extra errant bytes from corrupting a complete message.

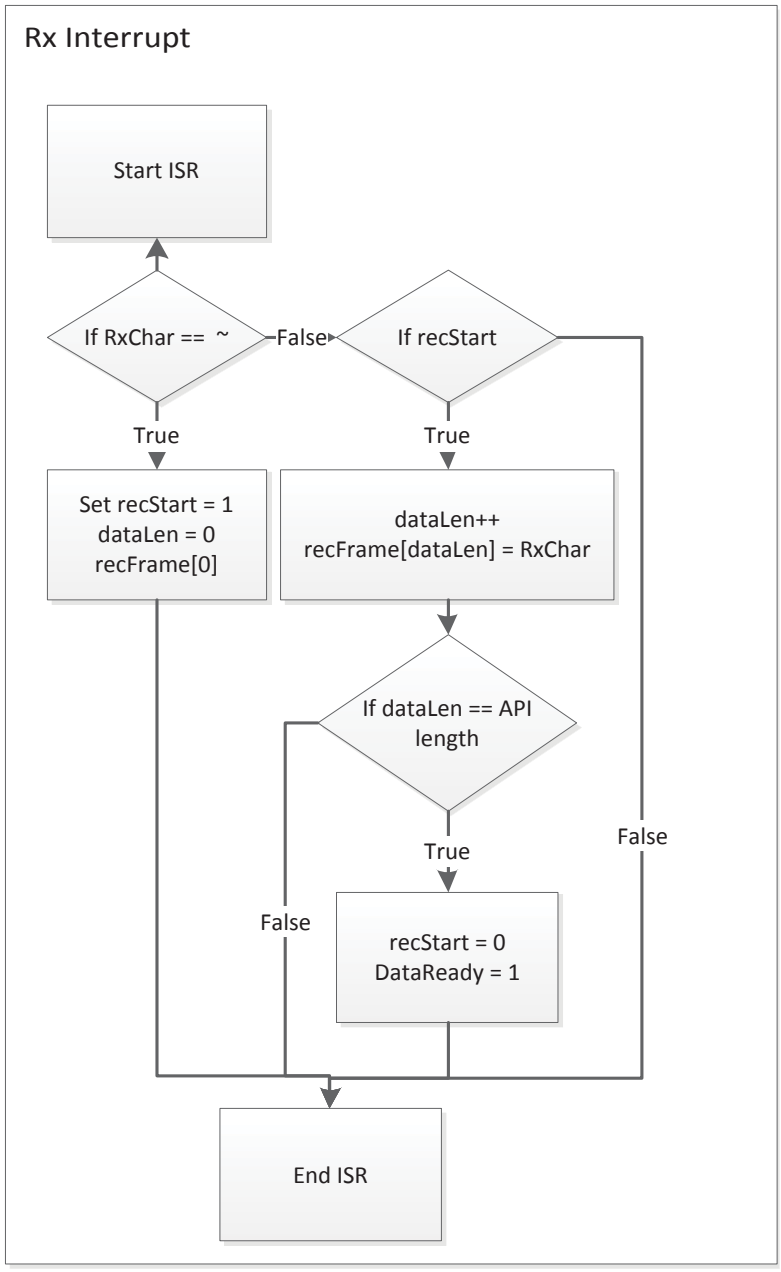


**Figure 5-17. Flow diagram of the Received Data Interrupt**

### 5.2.4. SENSOR READING

The raw data measure by the ADC is useful, but doesn't provide us with a comprehensible value. ADC values range from 0 to 1024; this is not easily understandable as a temperature, gas level, or battery voltage. However, a simple mathematical translation can turn these values into useful data. This takes very little effort and can be undertaken on the microcontroller, however, we have determined that within the software it is easier to work with the raw data and even transmit the raw data. Temperature, for example, can be measured to sub degree accuracy. To store floating point numbers takes significantly more memory then an integer and avoiding

this is beneficial on a small micro controller. The transmission of a floating point temperature is difficult as well; the numbers and decimal point are sent as characters and would have to be interpreted on the receiving end. To avoid this complication the translation of the data from raw data to representative data is handled by the Base Station software. This decision affects the discrimination software within the Daughter. By only storing raw ADC values on the Daughter all discrimination and testing will need to be done with respect to this scale. The sensor reading method is run after each ADC measurement. It is tasked with reading the current data and checking it against predefined threshold and checking the rate of change. The method is also tasked with setting the warning flags if these thresholds are exceeded. The sensor read method, shown in figure5-18, works using a switch function to choose which sensor data to work on. The data that will be checked is the most recently changed value. This is selected by using the mode variable set by the ADC interrupt.

```
//Read the Sensors, cycle through to read all
void sensor_Read(void){
    int diff;
    //int temp=0; // for LED only
    switch (mode){
            case 0: //Measure the Battery Level, A4
                    if (BattLvl < 460)//check if Battery lvl
                    {
```

**Figure 5-18. The sensor read method and the switch statement that controls it**

The number of cases can easily be increased to accommodate more sensors if they are added. This makes adding sensors with specific interpretations easy and fast. The first case, which can be seen in figure 5-19, is the battery voltage case. This case checks the current reading against predetermined values. It uses simple if statements to check whether the voltage is lower than expected values. If the battery voltage is lower than the point a flag is set that corresponds with that voltage. For example, when the ADC reading is lower than 460 it means the battery has discharged below 7V. After this point the Gas sensors regulator will not be providing a reliable 5V output. After the flag is set the Gas sensor should cease operation and warn the user. These actions are undertaken in the flag check method.

```
case 0: //Measure the Battery Level, A4
        if (BattLvl < 460)//check if Battery lvl
        {
            if (BattLvl < 410)//check if Battery
            {
                flags |= 0x80;
            }
            else
            {
                flags |= 0x40;
            }
        }
        break;
```

**Figure 5-19. Code that sets the low battery flags when the ADC value below a set value**

The other two cases currently implemented in this method are for the two sensors; Gas and Temperature. These two have identical format and this can be seen in figure 5-20. The only difference in the code is that of the threshold and rate of change values used in the IF statements.

```
case 1: //Check Temp Sensor, A5
        //temperature = SensorB; //this is for turning raw
        diff = abs(SensorB - prevB); //Calculate the rate
        if(diff > 100) { flags |= 0x08;} //Set flag to rat
        if(diff <= 99) { flags &= 0xF7; }
        if(SensorB <= 499 ) { flags &= 0xFB; } //Check for
        if(SensorB > 500) { flags |= 0x04; } //Check for t
        prevB = SensorB; //Store the current value in the
        break;
```

**Figure 5-20. Code for checking the temperature reading for important events**

The code shows that the first step is to calculate the rate of change. This is done as the absolute different between the current and previous readings. The reason for an absolute value being used is so that the test is done on the magnitude of the change not the direction. A rapid drop in temperature could be just as important to note as a rapid increase. The code shown in figure 5-20 is using arbitrary thresholds for the discrimination. Rate of change and absolute thresholds would need to be determined by trained professionals and would need to relate to the type of work being attempted. For testing purposes the arbitrary values prove that the system works. The code simply compares these thresholds against the reading and sets or clears the flags when appropriate. The final step is to save the current reading into the previous reading variable so that the rate of change can be accurately monitored.

### 5.2.5. FLAG CHECK

The flag checking method is very important to the Daughters operation. This method will check the flags array and react to the current state of the flags. These reactions can vary from simple tasks like turning off a timer to something more complex like alerting the user to an event. These reactions can be modified and expanded based on need or the specific operation. The flag check method simply provides a frame work to implement reactions to events in a straight forward way. This method is called during the main loop after the ADC measurement has been triggered and the sensor values read and analysed. If the action required was carried out only because the flag was set then the system would send continuous error messages once a flag was set. This would flood the network with messages and severely limited the network. To avoid this, a variable is initialised in parallel to the flags character. This new variable is also a character and each of its bits represent whether a warning has been set, 0 for no and 1 for yes. Using both the flags and warning characters we can ensure that the flag checking method only sends one warning. Figure 5-21 shows the flag check for a below 7V battery flag.

```
//when Vbatt < gas level - stop timer, turn off mosfet
if(flags & 0x80 && !(warningSent & 0x80)){ //check flag
    warningSent |= 0x80; //Warning has been set flag
    P2OUT &= 0xDF; //turn off mosfet
    TACTL  = MC_0; //turn off the timer
    buildDataString(BattLvl , '8', mystr);
    length = buildAPIframe(APIframe, mystr);
    txAPIframe(APIframe, length);
}
```

**Figure 5-21. Code for reacting to a low battery voltage flag**

The code checks whether the flag and corresponding warnings are the correct state and if so executes the action. The action for this case is that of setting the warning flag to high, turning off the MOSFET, stopping the timer, and transmitting an API frame with the measured battery level. The reason the MOSFET and timer are turned off is because this stops the gas sensor from functioning. The gas sensor measurements are dependent on the state of the MOSFET control pin. By setting this low it ensures that no more gas sensor measurements are taken. To ensure this never changes the timer, whose interrupt is the only other place the MOSFET control pin state is changed, is

turned off. The flag check system contains an IF statement and reaction code, like the code from figure 5-21, to handle all the flags.

The rate of change flags are exception to the single warning system. Because a rate of change may only occur once, or may occur on simultaneous readings it is important to take note of all of them. A single spike may be nothing of value, but two or more in a row shows a rapid and continuous change in the readings. This is very important and should be reported. This means that once a rate of change warning has been completed the flag must be reset. If this causes excessive transmission of packets over the network then either the rate of change threshold can be increased to make the flag harder to trigger, or code similar to the battery readings can be implemented.

### 5.2.6. BUILDING API FRAMES

Building a data frame that complies with the API framework set forth by ZigBee requires specific code. While the details of the API framework will be covered in chapter 5, a brief description of the software that builds and transmits these data frames will be covered here.

The first step to building a useful API frame is to decide on the message data to be sent. Primarily, the Daughter will be sending an ADC reading and a code character. The reading can be translated into a real world value for analysis and the code character allows the receiver to know what data has been sent. The method buildDataString is designed specifically for this. It takes the inputs of an integer value, a single character, and a pointer to a character array. Within the method, as shown in figure 5-22, the code first checks the value of the integer and using modulo division turns it from a large number into a series of ASCII character that represent each individual digit. This is done by taking the result of the modulo division and adding the ASCII character 0 as an offset. This gives the correct value for the ASCII representation of a digit.

After the digits have been added to the temporary array the flag character, which is the identifying code, is added. Finally the method loops over the length of the array and takes each individual character and transfers it to the pointer array.

Once the message, in this case the ADC reading, has been converted into a character array then next step is to use that as the transmitted data in an API frame. The advantage of the setup of the Daughter nodes is that the majority of the API frame is constant. For example the 64 bit destination address is always 0. This is the ZigBee defined address for the co-ordinator module, which is our user base station. The packet

type, frame ID, and options all remain the same and can be hard coded to their specific values.

```
void buildDataString(unsigned int val, char flag, char *str)
{
  int temploc = 0;  int i = 0;
  int digit = 0; int strloc = 0;
  char tempstr[5] ; //16-bit number can be at most 6 ASCII di

  //Turn the measured value into a string of ascii characters
  do
  {
    digit = val % 10;
    tempstr[temploc++] = digit + '0'; //add '0' to ensure the
    val /= 10;
  } while (val > 0);

  //Add the flag status to the string
  tempstr[temploc++] = flag;

  // reverse the digits back into the output string
  while(temploc>0) str[strloc++] = tempstr[--temploc];
  str[strloc]=0;
}
```

**Figure 5-22. BuildDataString method**

While this limits flexibility of the code, it also makes the software much simpler. The only things that are required to be set are the length, message data, and the checksum. To calculate the length of the packet the software uses the fact that the packet must contain at 17 bytes of data for header and checksum. To calculate the total length the software initialises a variable to 17 and loops over the message adding it to the API frame and increments the count variable. Once the message has been completely stored within the API frame the software now has the length, which can be stored within the 3rd byte. Figure 5-23 shows this looping. The loop increments along both the API frame and the input message array simultaneously transferring the relevant data. The loop finishes execution once it finds an empty character.

```
    i=17;
    while(*msg != '') //Add msg to the frame
    {
        APIframe[i] = *msg;
        i++;
        msg++;
    }
```

**Figure 5-23. While loop for transferring message array to the API frame**

The last part of the API frame is the checksum. This is a one byte value at the end of the frame used for error checking. The checksum is the 255 minus the sum of all the byte after the length up until the checksum. Since the checksum is only a single byte as it is added up it will overflow and start again. This means the checksum value will always be a value between 0 – 255. Figure 5-24 shows the code that loops over the message and sums the values and the value being stored in the final location of the API frame. This location is determined by the length variable, in this case 'i', that has been incrementing as the packet was built.

```
for(j=3;j<i; j++){ //calc check sum data between length and
    temp += APIframe[j];
}

checksum = 0xFF - temp;
APIframe[i] = checksum;
```

**Figure 5-24. Code for calculating the checksum value**

The final step is to transmit the frame. This is done by iterating the length of the API frame and transmitting each character. The loop waits for the transmitter to be clear and then proceeds to send the next character.

### 5.2.7. READING API FRAMES

The reading of API frames is just as important as sending them. To read an API frame some of the previously mentioned important bytes must be used. For example, bytes two and three tell us the length of the API packet. If the number of bytes received does not equal this then the packet has errors or missing data. If the packet is the correct length then the checksum value will tell us whether there are any errors present. Figure 5-25 shows the code that loops over the received packet and calculates the checksum. This is very similar code to the code used to build an API frame. Using the dataReady flag, and having it reset to low, means the same packet will not be read twice.

```
flags &= 0xDF; //set dataReady low till more data is received.
for(j=3; j<dataLen; j++){ //calc check sum data between length
    temp += recFrame[j]; //sum values
}
check = 0xFF - temp; //calc checksum
```

**Figure 5-25. Code for calculating checksum**

The calculated checksum is stored in a variable and this is compared to the checksum within the received frame. The checksum is located at the end of the

received frame and can easily be indexed using the dataLen variable. Figure 5-26 shows the IF statement that compares the checksums. If the checksums match then the software will read the 15th byte. This byte is always the message sent by the user. If it is a '1' or 'A' then the software returns 1, and returns 2 for '2' or 'B'. If the checksum fails then the method will return a 0.

```
if(check == recFrame[dataLen]) //check sum matches
{
    if(recFrame[15] == '1' | recFrame[15] == 'A'){ return 1;
    if(recFrame[15] == '2' | recFrame[15] == 'B'){ return 2;

}
return 0;//check sum doesnt match, return 0
```

**Figure 5-26. Code for reading the received message and acting upon the data**

These values are used in another method that selects an action based on the returned value. 0 is the default case and executes no action, while a 1 would cause the software to build an API packet with the current Gas sensor reading and transmit it. This provides the user a system to request specific data whenever necessary. This is used later to periodically poll all the nodes and updates the map of the environment.

## 5.3. Sensor Tests

To be successful and operate as intended the Daughter must be able to accurately measure the environmental factors, and its own battery voltage, as well as have working discrimination. With this in mind experiments were carried out to test each feature and sensor as necessary. The tests were designed to test the functionality of each subsystem and check the accuracy of each. These tests were published at the 2012 IEEE International Instrumentation and Measurement Technology Conference [11].

### 5.3.1. Thresholds and Rate of Change discrimination

The threshold and rate of change tests were completed during the prototyping of the Daughter nodes. The board used to complete these tests was a simplified version where the sensors were replaced by potentiometers. The board is shown in figure 5-27 and shows the two potentiometers that are substituted for the sensors.

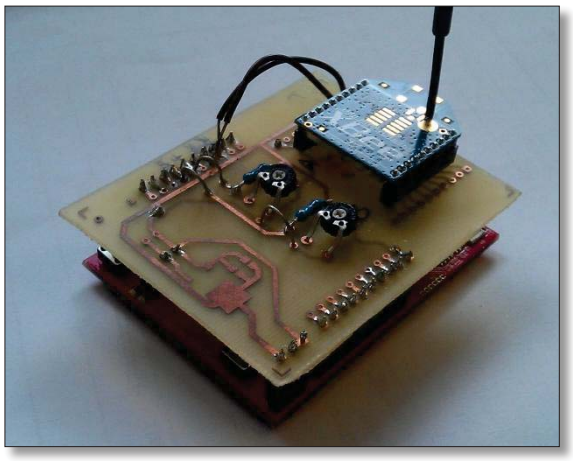

**Figure 5-27. Prototype of the Daughter node without sensors**

The tests to check the threshold and rate of change consisted of simulated sensor reading changes. The potentiometer provides a variable voltage as it is adjusted and this was fed directly into the input pins of the microcontroller. The first test was that of a basic threshold. To do this the software was set with an arbitrary threshold of 500. When this threshold is exceeded the software would turn on an LED on the microcontroller board. This test was performed by incrementing the voltage going into the input pin by adjusting the potentiometer. At increments of 0.1V the ADC value was checked and the state of the LED observed.

| Voltage (V) | ADC value | warning |
|---|---|---|
| 1.5006 | 428 | FALSE |
| 1.6006 | 456 | FALSE |
| 1.7005 | 485 | FALSE |
| 1.8006 | 514 | TRUE |
| 1.9003 | 542 | TRUE |
| 2.0005 | 570 | TRUE |

**Table 5-1. Threshold test results**

Table 5-1 shows results from the threshold test. The table shows selected results around the transition point. The table clearly shows that as the voltage and ADC reading increase past the threshold of 500 the warning changes from true to false.

The second test is to check that the rate of change discrimination works. To do this the input was provided by a power supply. The supply was controlled by the laboratory computer and allowed changes in voltage to be made by command. This allowed us to step from one voltage to the next without having to manually ramp the voltage. Again the software was edited to have an arbitrary rate of change threshold. In this case a value of 100, or approximately 0.4V was chosen. The LED warning was coded to turn

the LED on when the threshold was exceeded. Table 5-2 shows the results from this test.

| Voltage (V) | ADC value | warning |
|---|---|---|
| 1.5006 | 428 | FALSE |
| 1.6006 | 456 | FALSE |
| 1.7005 | 485 | FALSE |
| 1.8006 | 514 | TRUE |
| 1.9003 | 542 | TRUE |
| 2.0005 | 570 | TRUE |

**Table 5-2. Rate of Change test results**

As the table shows the LED warning was not present during low changes in voltage. The other notable result is that the warning worked on magnitude of change and ignored the sign. AS previously mentioned this is very important as drops in sensor readings can be just as important as rises.

From these tests we can conclude that the threshold and rate of change software works. This will help the Daughter monitor the environment and internally decide what sensor readings are of note and only inform the user when vital. This test must be considered successful only in the fact that the software works, exact thresholds and values must be determined once more is known about the final operating environment is known. The thresholds that are set will directly affect the operation of this system and determining these must be done carefully.

### 5.3.2. Temperature Sensor

The test undertaken to check the temperature sensor was designed to check the ability of the sensor to accurately measure an environmental factor. The test consisted of measuring the air temperature in the laboratory with both the Daughter node and a reference temperature sensor. In this case the reference was a fluke thermometer that was available. To adjust the air temperature an electric heater was used and both sensors placed directly in front of the heater. The heater was turned on until the air temperature had reached a maximum and was not increasing any further. The heater was turned off and measurements taken as the air temperature returned the temperature sensor, as describe earlier in this chapter, has a variable gain function. The test was carried out with the gain set to its highest value.

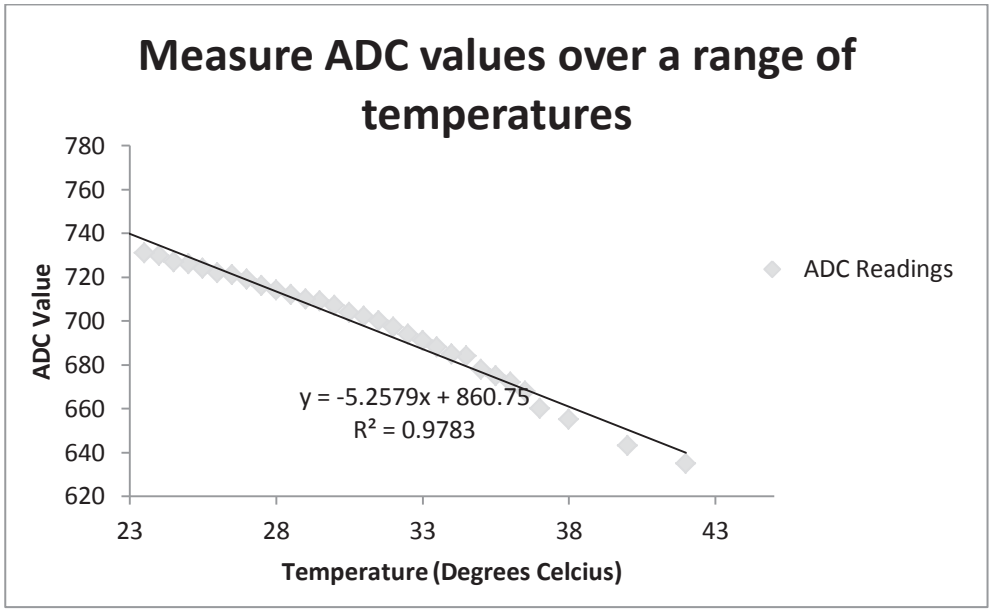


**Figure 5-28. Graph of measured ADC values over a temperature range**

Figure 5-28 shows the measured ADC values as the temperature changed. These results show a fairly linear output from the ADC as the temperature changes from around 23 degrees right up to 43 degrees. There is some clear variation and this can be attributed to the sensor output not stabilising fast enough during the tests. To confirm that these results are what should be seen from the sensor we can compare them to the datasheets expected values. The data sheet provides a table of expected voltages, in mV, for each gain setting at a range of temperatures. Figure 5-29 shows these values plotted on a graph.

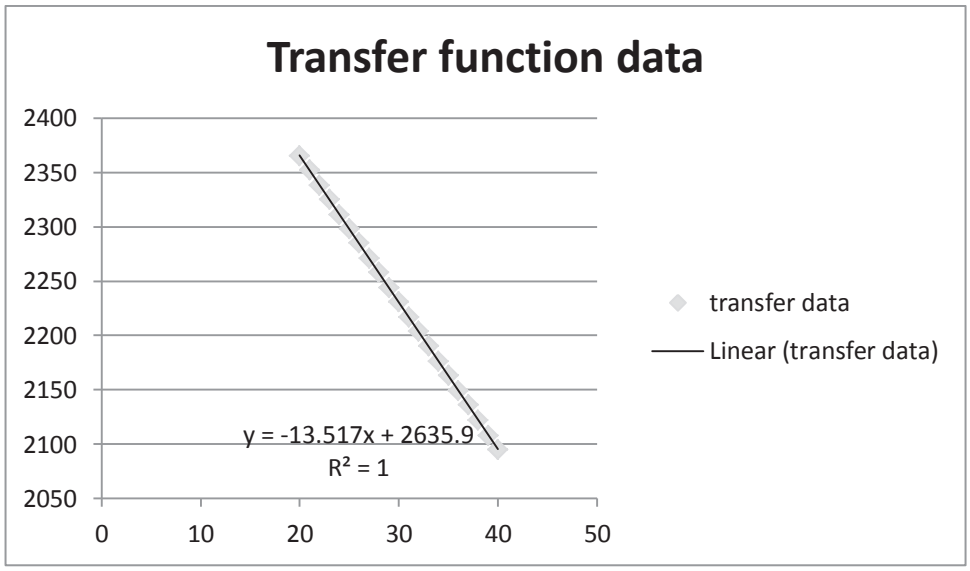


**Figure 5-29. Expected output voltages from temperature sensor datasheet**

However, as the data stands we cannot compare the two sets of data. The easiest method to make the data equal is to normalise it. To do this we can divide the equation by the intercept value. This will give us a normalised equation that has an intercept of 1. Doing this to both the expected and measured results gives us a gradient of -0.005 and -0.006 respectively. The slopes of the curves are very close and indicate that our sensor is working and that our readings are very accurate.

### 5.3.3. GAS SENSOR

The experiment to test the gas sensor was designed to see if the gas sensor could detect the presence of gas in the air. This test was designed to prove that the gas sensor works within our requirements. The test rig consisted of a 3-ring gas burner and an LPG gas bottle. The sensor was placed on top of the gas burner, inside a plastic box for insulation, and the gas turned on. The software had been modified to turn the LED on when gas was present. Figure 5-30 shows the set up in place. The threshold was set to be an ADC value of 11. This was determined through measuring the base line ADC reading when no gas was present and then exposing the sensor to LPG to see what change could be observed.

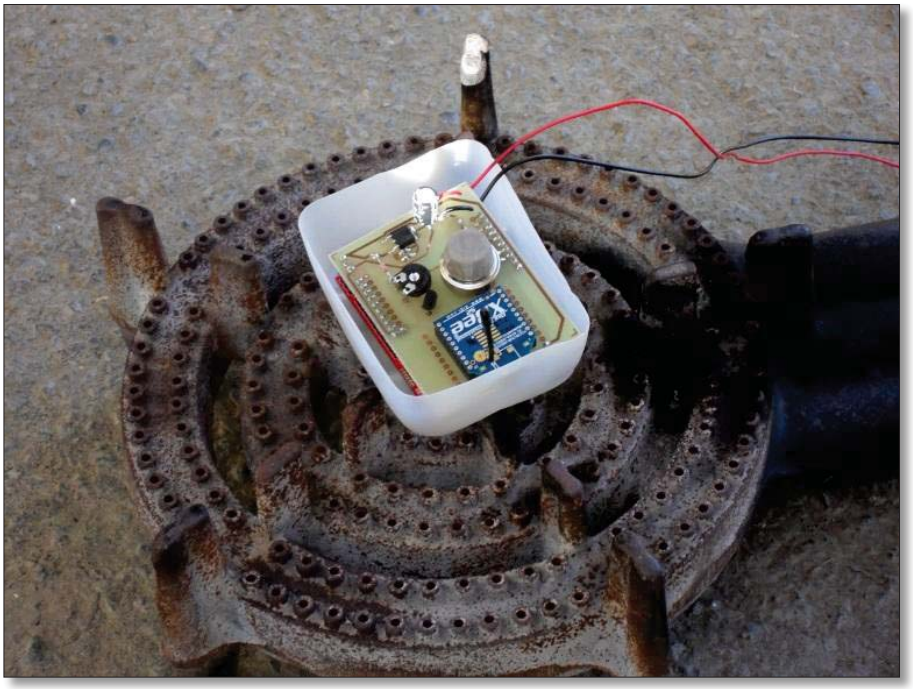

**Figure 5-30. Gas sensor test rig**

Table 5-3 shows the results obtained from the test. The test shows that there is a clear change in the warning output after the ADC reading exceeds 11. This confirms the initial tests and proves that the ADC value increases when gas is present. Repeat tests have confirmed that the presence of gas is detectable and reliably measurable.

| ADC value | Warning |
|---|---|
| 8 | FALSE |
| 8 | FALSE |
| 8 | FALSE |
| 8 | FALSE |
| 9 | FALSE |
| 9 | FALSE |
| 10 | FALSE |
| 10 | FALSE |
| 12 | TRUE |
| 13 | TRUE |
| 13 | TRUE |
| 14 | TRUE |
| 14 | TRUE |
| 14 | TRUE |

**Table 5-3. Gas sensor test results**

## 5.4. Battery Level Measurement

The test to check the battery voltage measurements was designed to test both the reading of the voltage and the voltage divider that makes it possible. The Daughter node was connected directly to a power supply with a variable voltage output. The voltage was varied from 7.4V down to 6V. 7.4V is the nominal voltage of the batteries used, and 6V constitutes the lowest possible voltage before the batteries may be damaged. As previously mentioned LiPo batteries should not be discharge below 3V per cell. The Batteries when fully charge will reach up to 8V but since the issue is discharging this was not measured. Table 5-4 shows the results from this test. It shows the input voltage from the power supply, the voltage has been scaled down to via the voltage divider, and the ADC reading from this voltage. This checked that the ADC could easily measure changes in battery voltage and also provides data about specific ADC values that correspond to battery levels. These have been used in the software to help warn the user of battery related events.

| Supply Voltage (V) | Sensor Input Voltage (V) | Battery Level (ADC) |
|---|---|---|
| 7.39 | 1.57 | 484 |
| 7.29 | 1.55 | 481 |
| 7.19 | 1.53 | 470 |
| 7.09 | 1.51 | 467 |
| 6.99 | 1.48 | 457 |
| 6.89 | 1.46 | 451 |
| 6.79 | 1.44 | 445 |
| 6.69 | 1.42 | 440 |
| 6.59 | 1.4 | 433 |
| 6.49 | 1.39 | 430 |
| 6.39 | 1.36 | 424 |
| 6.29 | 1.34 | 416 |
| 6.19 | 1.32 | 410 |
| 6.09 | 1.29 | 403 |
| 5.99 | 1.27 | 397 |

**Table 5-4. Battery Level measurement results**

### 5.4.1. Power Consumption

The power consumption of the Daughter nodes is very important. The longer the operational life time of these devices the more information that can be gather and the longer we can monitor the environment. For USAR the longer we can gather information the better. Details about an environment can help make rescue operations safer for longer periods of time and hopefully increase the chance of finding someone who is trapped. The power consumption of the Daughter has two distinct phases. The first is the initial stage when the battery voltage is above 7V. Within this phase the gas sensor will be turned on and off at 1minute intervals. To calculate the consumption of the node a few assumptions need to be made. First is that the ZigBee module is drawing a constant current. In the real world the current differs as it receives, transmits, or sits idle. But for this calculation we will take the worst case scenario of an always transmitting ZigBee. The second is that 0A are drawn by the regulator and the gas sensor when it is not operational. The final assumption made is that as power is consumed the voltage will drop in a linear fashion. For example, 100% capacity remaining will have a voltage of 4.2V per cell, while 0% capacity remaining will have a voltage of 3V per cell. Figure 5-31 shows the theoretical curve of battery voltage as the battery discharges. This curve is idealised and should be viewed as such. There are discharge curves available and battery models proposed, but because this is a simple

estimation it does not matter that the discharge curve has been simplified to a basic linear function.

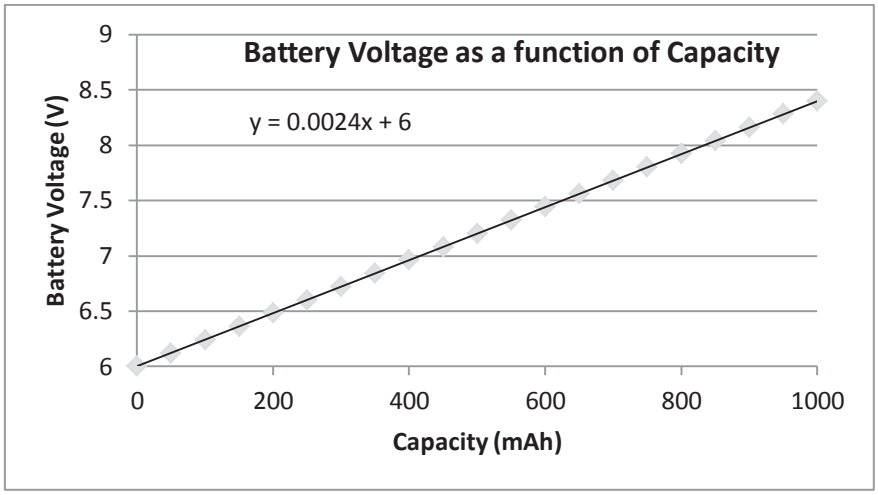


**Figure 5-31. Battery voltage as a function of capacity**

The next simplification is averaging the current draw by the Daughter per hour. The Daughter draws 200mA when the gas sensor is on and only 50mA when transmitting. Since the gas sensor is only running half of the time the average current draw over an hour is 125mA. Using the equation from figure 5-31 we can see that 7V corresponds to 416.7mAh remaining, or 41.7%, capacity remaining. This means that theoretically the Daughter can run, with the gas sensor, for up to 4.5hours. After this point the current draw is only 50mA per hour which translates to another 8.5hours of operation before the batteries reach the maximum discharge voltage. This gives the Daughter nodes an operating time of 13 hours.

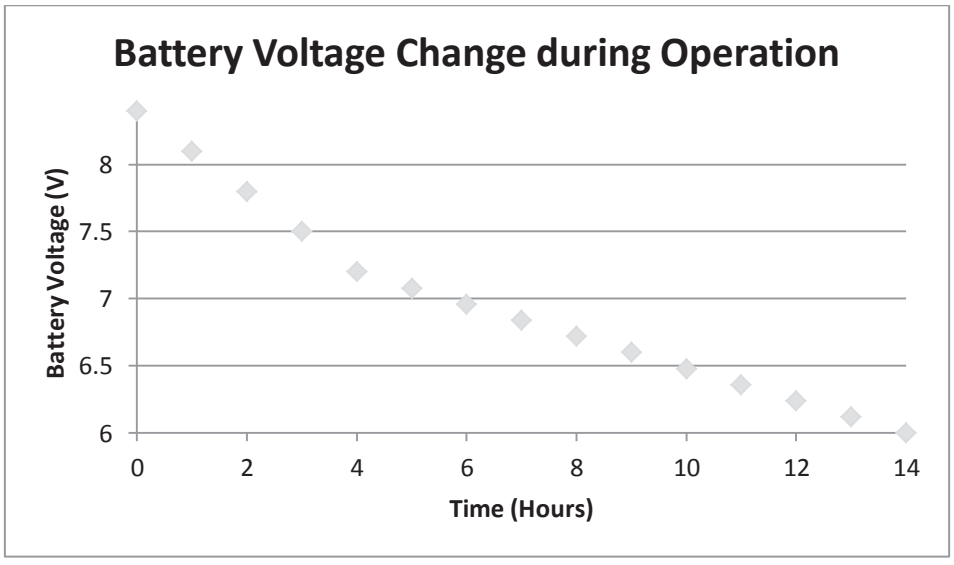


**Figure 5-32. Battery voltage shown as a function of time**

Figure 5-32 clearly shows a much faster drop in battery voltage over the first 4.5 hours. After the 4.5 hour point, which also represents the 7V mark, the rate of battery discharge decreases and continues consistently until 14hours. It must be restated that this mathematics shows a much idealised view of the system. Factors such as variations in the current draw by the ZigBee and the gas sensor are not accounted for. Also, and most importantly, the battery discharge has been assumed to be linear and the capacity to be 0% at 6V. To check the validity a real world operating test must be undertaken.

### 5.4.2. Actual Power Consumption Test

The actual operating time test was run to check the accuracy of the mathematically calculated operating time and to check that all the systems and operations will work over an extended period of time. The Battery was charged fully and the full charge voltage measured at 8.354V. The Daughter was set up on a desk and left to run. The test was run without continuous transmission, which lowers the overall current draw of the node. The results of the test can be considered a best case scenario for the Daughter.

To measure the voltage of the battery during the test an existing Massey LabView program was edited to request data from the multi meter at specified intervals. The current software allowed users to choose one of the measurement devices currently connected to the computer and perform predefined actions. These actions include requesting measurements, changing modes, setting outputs, etc. The software was modified so that a request was sent to the multi meter for the current measurement. This request was delayed by an adjustable amount and the delay could be controlled from the test computer. The software writes the current data to a file and stores it on the computer. The end result is a file that contains all the measured values and a time stamp for when they were taken.

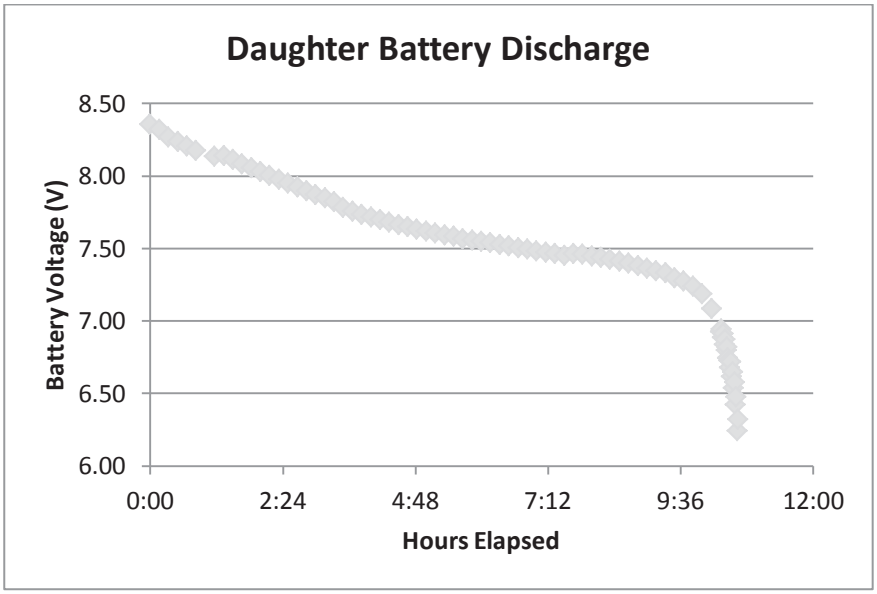


**Figure 5-33. Graph of battery voltage during operation**

Figure 5-33 show the voltage of the battery during the test. There is one missing measurement at the 1 hour elapsed time; this is due to an error in the software that was corrected. The graph shows that the node was operational for over 12 hours. During the first 10 hours of operation shows a reasonably linear decrease in battery voltage. The major difference between this and the idealised curve from the theoretical calculations is that after the 7V point the battery voltage decreases dramatically. This can be considered an advantage or disadvantage. From an operational stand point it means that during the entire life of the Daughter the Gas sensor will function. This provides results for all sensors during the operation time. Once the battery has discharge below 7V there is approximately 30 minutes before the battery reaches the 6V mark. This provides very little time from the failure of the gas sensor to the potential failure of the battery. If this node, and by extension nearby nodes deployed at similar times, fail simultaneously then sections of the network may become unreachable. The system can operated for 10 hours in a best case scenario. This may be enough for rescue workers to secure the area and ensure that gas, power, and other hazards have been turned off or mitigated.

# 6. User Base Station System Design

The User Base Station acts as a coordinator for the entire system. This is a simple computer interface that allows an operator to issue commands to the smaller systems. The User Base Station does the high level work in setting tasks, or objectives as appropriate for the situation. These tasks are then translated down into simple terms that can be interpreted by the lower tier robots. The User Base Station will also act as an interface allowing the rescue teams to monitor the system. This will provide the search team with information such as the location of survivors, potential hazards, through to complex data such as maps of the environment.

## 6.1. HARDWARE SETUP

The User Base station has been built around off the shelf components and utilises common interfaces to make it easy to set up and easy to use. The system is designed to be used with any PC with the peripheral hardware all using the USB interface. Figure 6-1 shows a basic layout of the base station components.

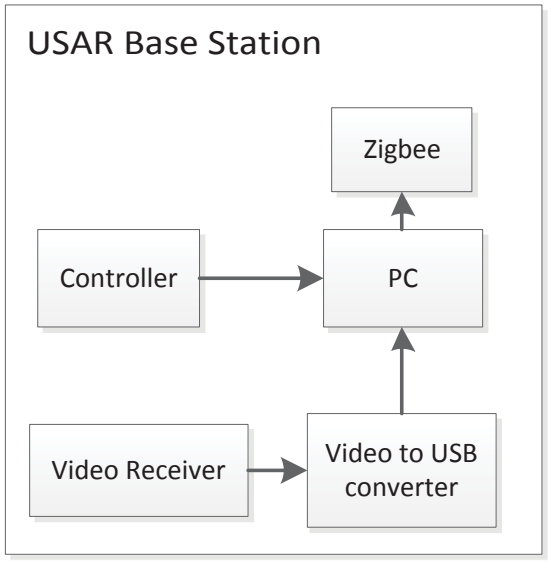


**Figure 6-1. Functional block diagram of the base station**

The base station, in its current form, has 3 major components connected to the PC: The controller, the communications, and the video receiver. Each of these systems provides an important feature to the system. The PC acts as a hub for these devices and the software allows them to work in unison to allow the user to control the system.

### 6.1.1. THE CONTROLLER

The controller was developed to provide a very simple interface for driving the Mother robot. One of the most common input methods in the world today is joysticks. All the major gaming consoles utilise joysticks to get the user to interact with the game. Joysticks have a natural mapping [66] that allows any user to quickly grasp how they are used. This makes them ideal for controlling a system such as this. Thumb joysticks, shown in figure 6-2, were cheaply and readily available [67]. The joysticks provide two analog signals; one for the horizontal axis and one for the vertical axis. This makes interfacing the signals very simple. The joysticks also provide a push button, which is activated by pressing down on the stick, which provides further flexibility.

**Figure 6-2. Thumb joysticks used for the controller** [67]

To simply design and development the controller was designed to run on a Texas Instruments Launchpad, the same used for the Daughters. This provided a USB interface for the controller and the ability to quickly and easily develop software that would read the joystick positions and transmit them to the PC. It also provides the ability to develop the controller further and directly interface with a ZigBee module to control the Mother without a PC. The software for the controller was derived from the Daughter software, utilising the UART and ADC functions to read the input channels and transmit the data. This software is included in the companion CD and will not be discussed in detail due to its simplicity.

The controller itself was only developed as a prototype and, therefore, no considerations have been made for the ergonomics or aesthetics of the device. Figure 6-3 shows the prototype developed for the controller. The joysticks were placed on a

custom PCB that acts as a shield and is placed on top of the Launchpad, in the same manner as the Daughters.


**Figure 6-3. Prototype of the handheld controller**

### 6.1.2. COMMUNICATIONS

The communications module for the base station is simply an Xbee module that allows the PC to communicate with the network. To keep the connection interface consistent the Xbee module needed a way to become a USB module. Sparkfun produces the Xbee explorer[68] which provides a USB interface. This board provides on board regulation so that the Xbee module is USB powered. Figure 5-3 shows the Xbee explorer.

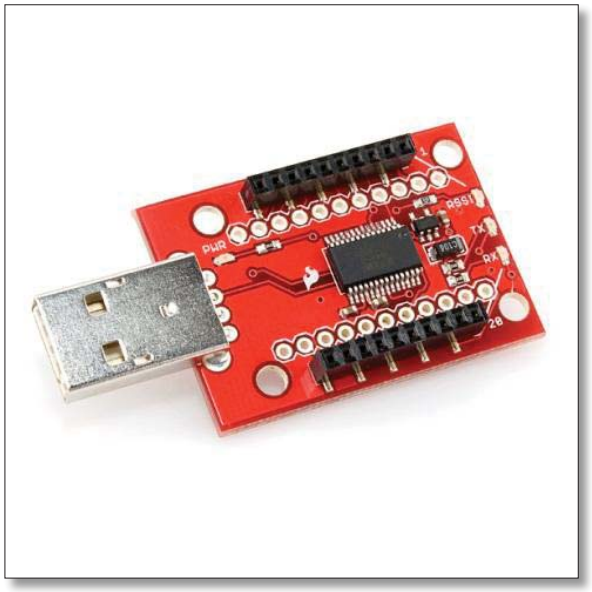


**Figure 6-4. Xbee explorer from Sparkfun** [68]

## 6.2. User Base Station Software

The User Base Station software was developed to provide a simple user interface for debugging and testing system functionality. The software links together all of the elements of the search and rescue system and will provide an operator with a simple UI to control the system. The software is event driven and only acts in response to user generated events. However, there are functions that are set up upon start-up of the software. The initialisation phase sets up the serial communication between both the ZigBee network and the hand held controller. Once the communications has been set up the system moves onto the main screen, this allows the user to interact with the system.

### 6.2.1. User Interface

The user interface is currently designed to be used by the development team. The interface is not intended to be used by the lay man. The system has labels which give a general idea of what each control does. However, the functionality requires specific actions to be performed in the correct order. The system will disable buttons until actions have completed or before the user has executed actions in the correct order. This is not a perfect system and relies on the user being knowledgeable and experienced with the system.

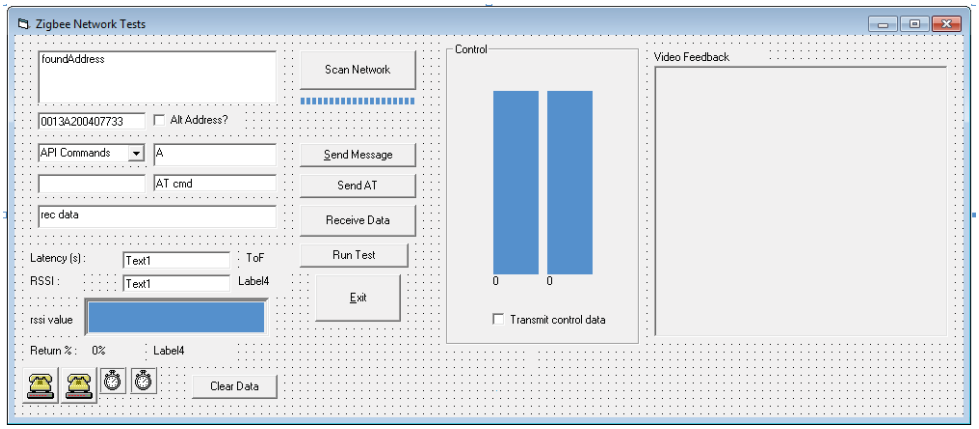


**Figure 6-5. User Interface layout**

This interface will evolve as new system features are added, the final version of it will be very user friendly and intuitive. This will make the system usable by any person, trained or novice.

### 6.2.2. COMMUNICATIONS

The communications is vital to the operation of the system. The communications provides the link between the Mother, Daughters, and User. If this link is severed then the system will become inert. With this in mind the software on the User Base Station must be capable of detecting the network, communicating with one, or all, networked systems. This communication must include both data communication and the ability to transmit remote commands.

#### *A. Controller Data*

The communications with networked systems creates the core of the overall system but there are still communications to components outside of this which are vital to the use of the system. The handheld controller is one of these components. Without the controller the robot would be difficult to control and manoeuvre. The controller communications is a request and reply system. When the controller itself receives a request, it then replies with the joysticks positions in a predetermined format. This data is sent via the serial port to the user base station computer. The serial port is set up during the initialisation phase, where the user must select the serial port that relates to the controller.

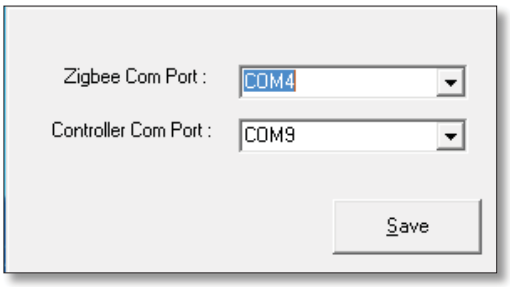


**Figure 6-6. Selecting comm ports for user base station**

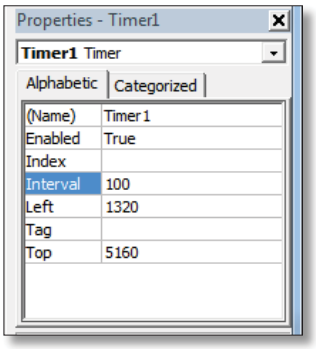


**Figure 6-7. Timer 1 properties menu**

Once this has been completed, along with selecting the network serial port, the program enters its operational state. Within this state a timer runs with a period of 0.1s.

Each time the timer completes counting an interrupt is executed. Within this interrupt a request is sent to the controller. This request has no specific form, it only needs to trigger the receive interrupt on the controller. The message sent in reply contains the ADC measurement of each joystick on the controller. When this message is received it will trigger the received interrupt within the User Base Station software. The interrupt reads the input buffer and stores the data into an array. Once the data has been stored in the array a FOR loop executes to read each byte of data, this is shown below in figure 6-8.

```
For i = 0 To i = 3
    If val(i) = 76 Then
        If i + 1 > 3 Then
        i = i - (3 - i)
        Else: i = i + 1
        End If
        leftS = Round(val(i) / 255 * 10, 0) - 5
        If i + 2 > 3 Then
        i = i - (3 - i)
        Else: i = i + 2
        End If
        rightS = Round(val(i) / 255 * 10, 0) - 5

    End If
Next i
```

**Figure 6-8. For loop that extracts and scales the joystick position data**

The software then converts the data from its current form into a value from -5 to 5. This scale was determined to avoid the Mother being driven to fast during testing. This data is then displayed on the interface, primarily to indicate to the operator that the values are still being updated.

### *B. Network Discovery*

The network can consist of any number of components and can change at any time. For the user base station to be able to communicate with these nodes it needs to be able to identify what devices are available on the network. To do this the user base station transmits a network discovery command, as defined by the ZigBee protocol. This command is sent when the scan network button is pressed.

```
Private Sub findXbee_Click()
     Form1.foundAddress.clear
     If (MSComm1.PortOpen) Then
        MSComm1.Output = sendAT("ND", atParam.Text)
    End If
    Timer2.Enabled = True
    findXbee.Enabled = False
    findXbee.Caption = "Scanning..."
End Sub
```

**Figure 6-9. The software that occurs when the scan network button is pressed**

The code executed starts by clearing the known addresses from the text box on the main form. The software then checks that the serial port is open. If the port is open the software uses the send AT command method. This method works by building the API frame with the correct data to execute the desired AT command. The frame structure for an AT commands, as explained previously, requires a header, body, and checksum. The AT frame is generate in software with hardcoded values for the majority of the frame. Figure 6-10 shows how the delimiter and header are set to specific values.

```
'Header and option data
delimiter = "7E"
header = "0801" 'Frame type AT command, Frame I
options = Para
```

**Figure 6-10. AT start delimiter and header options software**

These values correspond to an AT frame and ensure that the ZigBee modules understand the intention of the frame. The options field is set by the user in the interface, but for most commands will remain blank. The body of the AT frame is simply the command itself. This is set using a string and converted to a hex value within the method. Figure 6-11 shows the body being created from the command parameter, and the message being assembled from its component pieces,

```
'Build body as string of Hex values
body = StringToHex(CMD)
body = header + body + options
checksum = calcChecksum(body) 'calculate checksum as string of
length = calcLength(body)
```

**Figure 6-11. Building the body of the AT Packet**

The length and checksums are then completed, also shown in figure 6-11, and then all of these parts are combined to create the complete frame, shown in figure 6-12. This completed frame is converted into bytes and then returned to the transmit buffer to be sent.

```
    'Build frame from delimiter, length, body, and checksum
    frame = delimiter + length + body + checksum
    ATframe = HexStrToByte(frame)

    sendAT = ATframe
```

**Figure 6-12. Combining the parts of the AT packet and returning them**

Once the network discovery command has been sent the system waits for 6 seconds before reading the received data. This is because the ZigBee modules reply to network discovery commands randomly, within a time frame. The modules are, by default, set to 6 seconds. By having the user base station wait till this time has elapsed we can ensure that all network discovery replies have been received and thus gain the best view of the network possible. The reception and decoding of data relies on the same receive method outlined in the Mother Robot software. The received packets are checked for errors, and then using the frame ID the appropriate reaction is undertaken for the frame type. Figure 6-13 shows how `select` statement and the call to the method `atResponse()`.

```
Select Case l
            Case 139 'Acknowledge frame
                'Form1.received = Form1.received + 1
            Case 136 'AT response frame
                atResponse packets(i)
            Case 144 'Data recevied frame
               temp = ""
               For l = 15 To packets(0)(2) + 2 Step 1 'build string
                    temp = temp + Chr(val("&H" + Hex$(packets(i)(l))))
               Next l
            Form1.recData.Text = temp
End Select
```

**Figure 6-13. Select function used to within received frame code**

This method simply decodes the type of AT response that has been received and uses the received data as appropriate. The first step, decoding the type of response, is done by extracting specific bytes from the message and concatenating them as strings, this is shown in figure 6-14.

```
        Next i
        Form1.foundAddress.AddItem (address + " - " + identifier)

Case "MY"
```

**Figure 6-14. Concatenation of the address and identifier**

Then by using this newly formed string and using it within a `select` statement the software determines the response type. For a network discovery response, “ND”, the system loops over the message and builds an address out of the data. Once the address has been complete the system will build up an identifier string. This identifier is simply a name that was given to the ZigBee module to help make testing and debugging simpler. These values are finally added to the `foundAddress` text box. This method is called for every network discovery reply. Once all the messages have been processed the `foundAddress` will have a complete map of the network, including identifiers.

### *C. Sensor Data Recovery*

The system needs to be able to retrieve up to date information on the environment from sensor network. To do this the system needs to be able to request the data from the Daughter Nodes. This request and reply system allows the User Base Station to keep update its information in whatever fashion is necessary. This system provides a method in which certain areas of the environment are updated more regularly because they are of more risk or more value to the research operation. To implement this method, the software simply sends a specific message to the Daughter Node that elicits a reply. The current version of the Daughter software accepts two separate requests for data. The first is the letter ‘T’, or ‘t’, which will result in the temperature data being returned. The second is the letter ‘G’ or ‘g’, which will return the gas sensor reading. Both of these replies send the raw sensor reading in reply and the User Base Station must interpret them before displaying the result to the user.

# 7. The Network

The network that is created by the deployment of the daughter nodes is the vital component of this system. It provides the infrastructure for this system that will make it useful as a search and rescue tool, research platform, or any application.

The network is tasked with providing two major functions. Firstly, a reliable communications link with the Mother robot. This is the key as the entire system hinges on the mother robot being within communications range, no matter its position in the environment and without the difficulties of dragging a cable with it. This frees up the mother to explore previously unreachable areas and provide the same reliability of a wired systems. Secondly, the network must provide a stable sensor network to keep the user updated on the environmental factors that are being monitored. These sensors supply key information to the user and the loss of these will create blind spots that represent uncertainties in the environment. These two functions are the cornerstone of the proposed system and make it more functional and applicable then systems that have previously been developed.

The network consists, primarily, of the daughter nodes. These act both as the wireless sensors for the system but also provide the wireless relay points that will provide the link for the mother to the user. They are tasked with measuring environmental factors and making internal decisions about the urgency of the measured levels. If the levels are too high or rapidly changing they will alert the user, but until these cases happen they will not transmit data unless requested. This is to prevent unnecessary information taking up bandwidth in the network or preventing vital information reaching the correct destination in a timely manner.

The other major components of this network are the mother and the user base station. At this stage of the research the transmission of data is one way from the user to the mother. The user will send control commands to the mother robot and tele operate the robot. The mother, much like the daughters, will not send data to the user unless absolutely necessary. The few cases where this would happen include low battery, or over current event, or motor failure. These messages are only needed at this stage to prevent damaging the system during testing. Once further research into the autonomy of the mother is completed it should be able to handle the majority of these issues itself.

So by utilising only three distinct systems we have created a network that not only monitors the environment but can control and operate a mobile robot remotely. The key features required to have this network operate as expected are that it must be a true ad hoc network and it must have redundancy built into it from the start. To say the network is truly ad hoc is to say that the network adapts to changes.

If a node is removed the network will adapt finding new paths from source to destination. If more nodes are added these are recognised and added to the model of the network. An example is that of a node being added to the network, where it is not vital for communication. The system must recognise the network has changed and when requesting data from the sensors add this to the list and thus grow the knowledge of the environment for the user. Another example is that of a new node added that provides a shorter route for the communications. The network must adapt, see this path has a better transmission route and change the routing to suit. If the networks not truly ad hoc then it will not be able to adapt to the changes that will occur during operations. The second feature, inbuilt redundancy, is important. The redundancy of the network will decide how much change it can cope with. The loss of any node constitutes a loss of knowledge of the environment which could be dangerous enough. But on top of this the loss of the node causing a loss of communication to an entire area of operation, or worst case the mother robot itself is unacceptable. The network needs to be able to respond to changes in nodes and reroute packets in such a way that the network maintains a reliable connection from user to mother. This is an incredibly difficult goal and harder to achieve by thought alone. To achieve this Lofty goal the method for distributing the nodes needs to be developed with real world data and extensive testing. We have made ground on the requirements for a distribution methodology but with different situations there will be different needs and we have only established basic parameters so far.

## 7.1. Network Specifications

The network developed needed to provide reliable communications coupled with true adaptability in the face of an every changing environment. The major specification that was imposed onto the network was that of transfer success rate. Since the system is controlling an autonomous system it must be able to reliably transfer control packets. Failure in this area means that the system will not be able to complete its goal. The target for success was set at 90% success rate with packets. This presents a very

high target for a prototype system and if met would provide a very strong basis for claims about the efficacy of the system and an impressive starting point for further development.

While the latency of communications would seem to be a major issue with control as well, in the sense that slower communications would lead to dulled responses and potential hazards, it was deemed unnecessary at this level of development. The early phase of development was to show the potential for relayed communications, while latency will be an issue in the final system it is also directly proportional to the number of nodes the message must traverse. Due to budget constraints the initial development was limited to 6 daughter nodes and limits our ability to test the latency/number of nodes problem.

As previously mentioned the system was limited to 6 daughter nodes by the research budget. This proved to be a very serious limitation as it restricted the number of testing possibilities that were available with the system. The limitation of node forced the testing to focus on proof of concept rather than in depth research into the validation of this method.

## 7.2. Testing the Network

The Network tests were set up to establish information about how well the network will work in specific configurations. The network makes use of some of the attributes of the ZigBee protocol and these tests are aimed at checking the functionality of these attributes, such as the ad-hoc routing protocol, and establish the real world connotations of network values, such as the relationship between distance and signal strength. These tests will help determine the scalability and reliability of the network and whether or not this concept will be functional at the level required for an urban search and rescue operation.

### 7.2.1. Point to Point Communications

The first stage of the network testing was to check the point to point communications abilities of the ZigBee modules. This test was designed to check the relationship between signal strength and distance between two nodes. The relationship between these two factors will be vital to the overall network concept. The network relies on the Mother robot deploying more nodes before communications are lost with

the last node. To determine when nodes should be deployed the system must have an idea of the distance between the Mother and the previous node.

The experiment was set up using the car park at Massey University. The car park provides a large, flat, open area with a length of approximately 200m with no obstructions. This provides a perfect area to test the communications. The base station was set up at the 0m mark. A single Daughter node was turned on and a Network discovery completed to check that the node had joined the network. The node was then placed 10m away from the base station. A data frame was sent to the Daughter node, which was told to reply with a specific message. Each time the data was sent, and successfully received the base station would request the signal strength from the attached ZigBee node. This test was repeated with the Daughter's ZigBee module programmed as an end device and a router.

| Distance (m) | RSSI average | Return % |
|---|---|---|
| 10 | 60.00 | 100 |
| 20 | 70.05 | 100 |
| 30 | 74.00 | 100 |
| 40 | 76.59 | 100 |
| 50 | 78.79 | 100 |
| 55 | 78.53 | 100 |
| 60 | 0.00 | 0 |

**Table 7-1. Point to point communication test results for an End Device**

Table 7-1 shows the results from the end device test. The communication failed to reply at 60m. This data is very important, as the current structure of the network has the Mother robot programmed as the end device. The above results show that the Mother robot can move up to 50m away from the closest node without losing communications. Table 7-2 shows the results from the test when the Daughter is programmed as a Router.

| distance | % correct | RSSI |
|---|---|---|
| 0 | 100 | 49.11 |
| 5 | 100 | 50.90 |
| 10 | 100 | 63.30 |
| 15 | 100 | 62.40 |
| 20 | 100 | 66.10 |
| 25 | 100 | 67.25 |
| 30 | 100 | 79.25 |
| 35 | 100 | 76.65 |
| 40 | 100 | 77.15 |
| 45 | 100 | 77.05 |
| 50 | 100 | 74.05 |
| 55 | 100 | 83.90 |
| 60 | 100 | 84.35 |
| 65 | 100 | 84.70 |
| 70 | 100 | 82.65 |

**Table 7-2. Point to point communication results for a Router**

These results show a much longer range with reliable communications. The range is successful up to 70m. This test was not carried to the point where communications would fail. This is because, as the data in table 6-1 shows, the Mother cannot travel further than 50m from anode, which means the nodes will need to be placed less than 50m apart. The results from the second test show that we will certainly have reliable communications if the nodes are less than 50m apart.

In terms of the overall distribution of Daughter nodes within the network these results have shown that while the Daughters, when programmed as routers, can communicate at ranges up to, and including, 70m. However, if there are any end devices present within the network they change the minimum distance to 50m. Because the only advantages of end devices are that they can enter a sleep mode to conserve power we can change the network design and make the Mother robot a router as well. This will draw slightly more power and shorten the operating life of the robot, but when compared to the current draw of the motors, computer, and other systems this is negligible. Doing this means that any two network elements can be up to 70m apart and if the minimum distance between two nodes was set to 35m that there would be an inherent redundancy due to potential communication range

### 7.2.2. AD-HOC NATURE OF THE NETWORK

The Ad-hoc nature of the network is very important to the overall functionality of the system. The system needs to be able to have new nodes added as the Mother robot proceeds further from the base station. The ZigBee user manual [45] outlines the Node Join Time parameter. This parameter determines how long, in seconds, a ZigBee module will allow new ZigBee modules to join the network. This value can be set to 0xFF and this will force the ZigBee module to allow new modules to join indefinitely. By doing this any new ZigBee modules added to the network will be able to join the network no matter where in the network they are added. Figure 7-1 shows an example network. The test undertaken has R3 at a distance from C1 which makes direct communications impossible, approximately 100m. R1 and R2 are situated next to each other at approximately 50m.

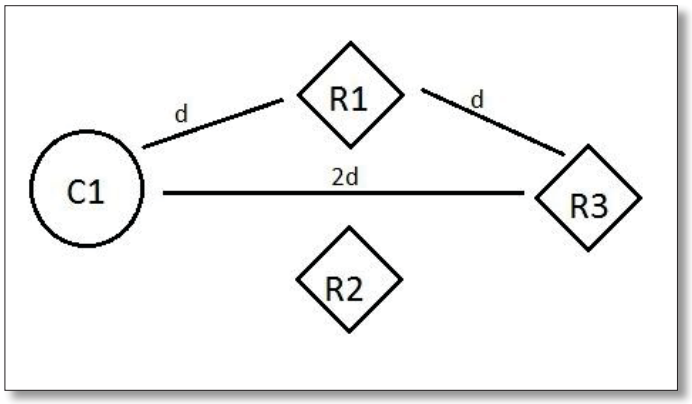


**Figure 7-1Graphical layout of the test network**

Initially R1 and R3 are turned on and a network discovery completed. The first test is to check that communications to R3 works through R1. Once the network is established, R1 is turned off and R2 is turned on. This tests the networks ability to determine if a path has become unavailable. If the path is unavailable the network will need to determine a new path, using the AODV routing mechanism described earlier, and send the packet through this new path.

The test showed that without manually updated, or informing the network that a node has become unavailable that it can successfully reroute frames when necessary. This showed 100% round trip frame delivery success and shows that the network has true ad hoc properties.

### 7.2.3. Relayed Communications

The next test important to checking the viability of the sensor and control network is the ability to send packets and have them travel through intermediate nodes reliability. The relayed communications need to be able to transmit and receive messages through a number of nodes and have the communications be reliable. With this in mind an experiment was designed that would test the ability to send from a fixed point to any node within a long chain. The chain would consist of Daughter nodes spaced far enough apart that they can only directly communicate with the adjacent nodes and no others. Figure 7-2 shows a map view of where each node was placed.

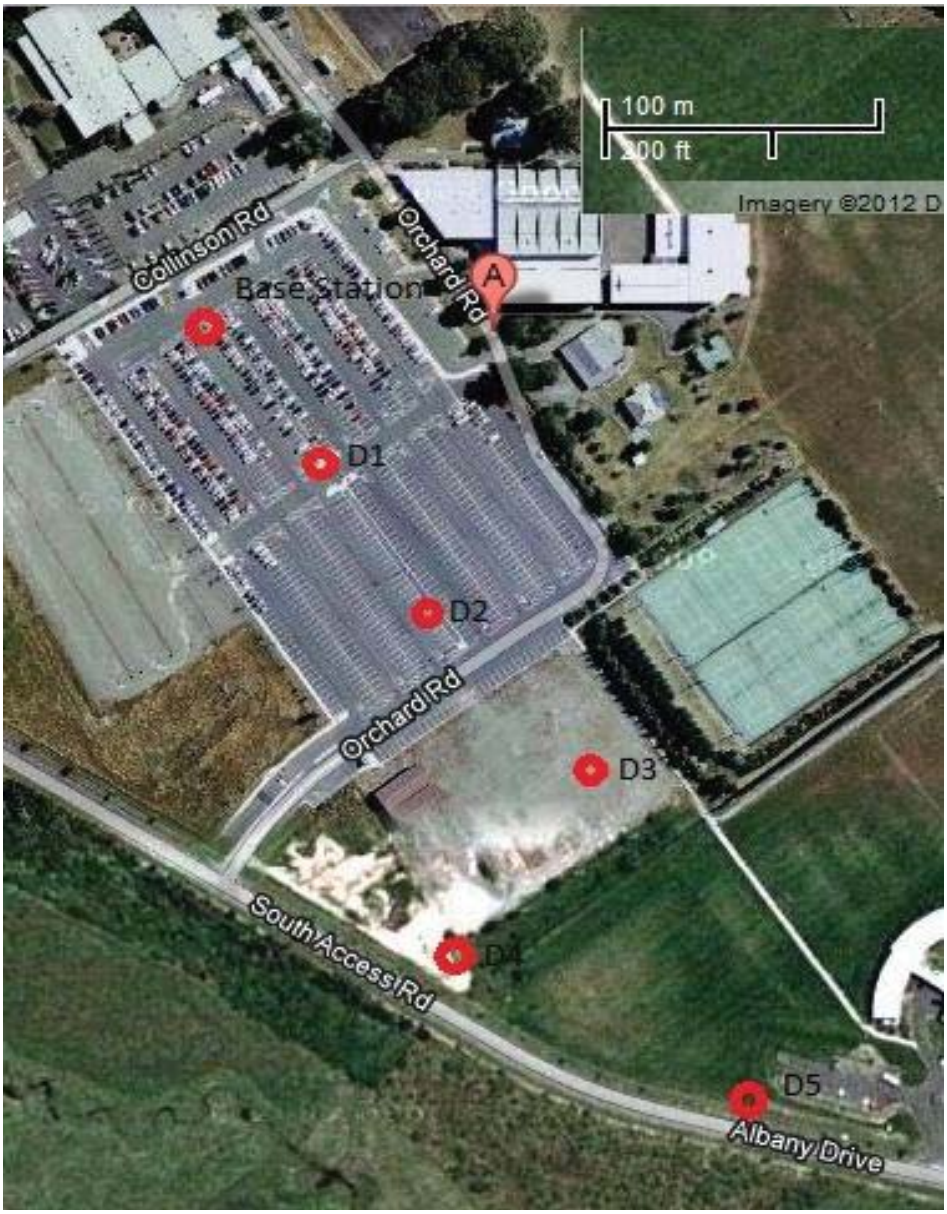


**Figure 7-2. Geographical layout of network elements**

The experiment was set up in the Massey University car park. The car park offers a large open area with line of sight. Line of sight is important as any object or obstructions can affect wireless communications. This experiment did not go to the extent of ensuring fully unobstructed Fresnel zones between each transmitter but ensured that each hops was a directly visible line of sight transmission. The experiment was completed in stages. Each stage consisted of the addition of an extra hop in the communications path. The base station was placed at a stationary location and turned on. The first node was turned on and moved sufficiently far away so that communications were not possible. This was done using the base station software and the node moved until the success rate was 0% on multiple test attempts. The base station also transmitted a network discovery command and the distance was determined by the 0% successful communications value and the inability to detect the node. The combination of these two factors gives very good certainty that the node is invisible to the base station. Once this distance had been determined an intermediate node was placed, approximately half way between the end node and the base station. This node was turned on and the basic communication tests repeated. This allowed the far node to again be detected and communications with it restored. The next node to be placed was required to only be visible to the final node. To achieve this, a similar process was undertaken, but the base station and the first node were left on. Once communications failed on multiple attempts the second node was turned on and then communications were tested. This showed that the first, second, and third nodes were all visible within the network and the communications between them and the base station were possible. This was repeated until all nodes were placed. The tests were undertaken to reach from the base station to each node. The message, a simple character, was transmitted to a desired address and the node was programmed so that it would echo this character. This would enable us to see a complete round trip of a message. For the message to return there would need to be no errors with the data.

| hops | % return |
|---|---|
| 2 | 100 |
| 3 | 95 |
| 4 | 85 |
| 5 | 100 |

**Table 7-3. Relayed communications results**

The results from this experiment are promising. The percentage of successful round trip transmissions is never below 85%. Over two hops the network had 100% success rate. At this distance there were never any issues with establishing connections, discovering nodes, or transmitting data. This coupled with the knowledge from a previous test about the maximum distance of point to point communications gives proof that a large area of operation can be achieved with minimal resources. A mere four nodes and a Mother would yield an operating area of approximately 100m in radius. In terms of search and rescue, which normally focuses on areas where people are expected to be found, this is an enormous operating area. At a range of 3 hops there were issues with communications. These were primarily in the first test and have brought the average down. There were issues with discovering the network and with transmission of data. These issues were also present at a range of 4 hops. These lows present a clear issue, not with multi hop communication but with stressing the maximum distance between nodes. The issue was fixed by varying the distance between the nodes. The shorter distances provide a much more stable communications channel and network discovery reliability is improved greatly. The results show that a small variations in distance, combined with all other factors affecting wireless communication, lead to huge issues with reliability. While this could be a major issue, if the system were designed to rely on a continuous chain of nodes, the current design and intent will see the nodes dispersed in high density over a relatively small area. Therefore the low success rates are not an issue.

# 8. Conclusions and Future Work

This research has been very successful. The result of this work shows an effective prototype system. The concept of a multi-tiered robotic search and rescue system is a very viable concept. The research has made steps toward developing an initial prototype of the Mother robot and Daughter nodes. The testing has shown the efficacy of these systems as well. The network concept that is required to tie all of the individual components has been tested and shown that the concept can work, and has the ability to provide a reliable communications with a range far exceeding that of the task.

## 8.1. Mother Robot

The Mother Robot development is an early stage prototype. It provides very basic functionality and provides a platform for further research. The mechanical design is simple, but effective. It has provided a stable platform that allows the electronic systems to be added to and expanded as research progresses. The current design was never intended for use in hostile, dangerous environments and so further research into locomotion and chassis design will further enhance the system's ability to survive in hazardous environments. These changes in mechanical design will require adjustments to the motor control and driving systems. The current design of these had this eventuality in mind. To make this change easier there is a simple interface between the motor controller and the single board computer. This means that as long as the new controller supports this interface, and the command structure associated with it, the controller can be swapped at any stage. This interface, RS485, also allows provide the ability to quickly install further motors without the need for redesigning the system. It provides a communications bus that allows further devices to be attached and the communications protocol allows these to easily be indexed and controlled. To ensure the controller is compatible with the driver there is a simple 5 signal interface that is used. By maintaining both of these interfaces the system provides a simple method for interchanging one, or both, or the motor subsystems. This will allow rapid change to the systems locomotion. These features combine to create a very robust prototype design that will aid researchers in their work.

## 8.2. DAUGHTER NODES

The Daughters nodes are simple wireless sensors that show a level of intelligence that helps keep the system free from excess data transfers. This intelligence helps the Daughter node determine the difference between minor fluctuations in the environment and serious changes in the readings. This is another application of distributed intelligence in the system, but primarily it has been designed to prevent network issues. The major issue that this intelligence prevents is data being transmitted all the time. If the Daughters were to continually update the user with every measurement value then the network bandwidth would quickly be used up and transmission of control data to the Mother robot made impossible. The use of basic intelligence has shown that the Daughters can distinguish between natural fluctuations in measured values and actual dangerous changes. In conjunction with this the Daughters have shown the ability to accurately measure environmental factors and transmit this data to the user. Future work will include testing a range of algorithms to determine whether the changes in measure values are relevant. These Daughters are a basic template and can easily be modified to be suitable for most tasks, additions or changes to the sensors will allow the Daughters to measure numerous environmental factors. Further research will be conducted into making these nodes into fully fledged robots. That means the Daughters will be able to explore or interact with the environment; primarily by a form of locomotion. This will give the system much greater flexibility in the types of tasks it can undertake.

## 8.3. USER BASE STATION

The user base station was a simple user interface that was designed for technical testing and debugging. The system will need much research to improve the human machine interaction and make it more intuitive. The system, as it stands, can only be operated by a trained professional who is familiar with its workings and errors. The system does not provide an interface that can be understood by looking at it and also allows users to execute actions even though they may be detrimental. A major issue with adoption of this type of system will be its usability. It needs to be simple enough to be used by someone at a glance, but powerful enough that all the functions can be used to their potential. This is a difficult challenge and will require a lot of research. Future research, with this challenge in mind, will require that error messages are simple to understand and the problem easy to fix. The major area of research needs to

focus on how to make the operation of this complex, multi robot system simple enough that zero training is required and that mistakes are difficult, if not impossible, to make. The ability to use this system will make or break it in a real world setting such as urban search and rescue.

## 8.4. NETWORK

The network is the core of the system. It provides a wireless communication link between the Mother robot and the user base station. The network also creates a wireless sensor network that provides the user with data from the Daughter nodes and helps create a map of environmental factors in the area. This concept of a dispersed network has proven very effective and tests have shown that a wide ranging network can be quickly established and provide effective communications. While there were some difficulties in testing, due to placing the nodes at the most extreme range from teach other possible, the system performed as expected. These issues were removed by decreasing the distance between the nodes in the tests. This small change in distance, usually under 5m, brought the reliability of the network back to 100%. While this shows a weakness of the ZigBee system, that a few meters can make a huge difference, it will not be an issue for our concept. Our concept relies on a small operating area and high density dispersion of the Daughter nodes. This will ensure that the distance between any two nodes is far below the maximum and prevent this loss in reliability. Future work in this field will be required to determine the best approach to deploying the nodes and improved algorithms for routing and data management. This research provides proof that the multi-robot urban search and rescue concept is plausible and can, with further work, become a valuable piece of equipment in urban search and save lives.

# References


[1] D. P. Stormont and M. D. Berkemeier, "Blue Swarm 2 . 5 A Step Toward an Autonomous Swarm of Search and Rescue Robots," *Artificial Intelligence*, pp. 1–5, 2003.

[2] N. Swarbrick, "Search and rescue," *Te Ara*, 2011. [Online]. Available: http://www.teara.govt.nz/en/search-and-rescue/. [Accessed: 22-Dec-2011].

[3] Anon., "Urban Search and Rescue (USAR)." [Online]. Available: http://www.fire.org.nz/About-Us/Our-Organisation/Pages/UrbanSearchandRecue.aspx. [Accessed: 22-Dec-2011].

[4] Anon., "Historic World Earthquakes." [Online]. Available: http://earthquake.usgs.gov/earthquakes/world/historical.php. [Accessed: 22-Dec-2011].

[5] R. Murphy, "Trial by Fire," *IEEE Robotics & Automation Magazine*, no. September, pp. 50 – 61, 2001.

[6] D. Carnegie, "A Three-Tier Hierarchical Robotic System for Urban Search and Rescue Applications," *2007 IEEE International Workshop on Safety, Security and Rescue Robotics*, no. September, pp. 1–6, Sep. 2007.

[7] R. R. Murphy, M. Ausmus, M. Bugajska, T. Ellis, T. Johnson, N. Kelley, J. Kiefer, L. Pollock, and E. F. Avenue, "Marsupial-like Mobile Robot Societies," in *Third Annual Conference on Autonomous Agents*, 1999, pp. 364 – 365.

[8] N. P. Papanikolopoulos, S. A. Stoeter, D. G. Krantz, K. B. Yesin, M. Gini, R. Voyles, D. F. Hougen, B. Nelson, and M. D. Erickson, "Enlisting Raners and Scouts for Reconnaissance and Surveillance," *IEEE Robotics & Automation Magazine*, no. December, pp. 14–24, Dec-2000.

[9] B. Le Comte and G. Sen Gupta, "Characterisation of a Motor Driver with Over- Current Protection and Speed Feedback," in *5th International Conference on Automation, Robotics, and Applications*, 2011, pp. 127 –132.

[10] B. Le Comte and G. Sen Gupta, "Two-Tier Wireless Robotic System for Urban Search and Rescue," in *Electronics New Zealand Conference*, 2011, pp. 8 – 12.

[11] B. Le Comte and G. Sen Gupta, "Distributed Sensors for Hazard Detection in an Urban Search and Rescue Operation," in *IEEE International Instrumentation and Measurement Technology Conference*, 2012, pp. 2385 – 2390.

[12] D. Schneider, “Robin Murphy: Roboticist to the Rescue - IEEE Spectrum,” *IEEE Spectrum*, 2009. [Online]. Available: http://spectrum.ieee.org/robotics/artificial-intelligence/robin-murphy-roboticist-to-the-rescue/0. [Accessed: 22-Dec-2011].

[13] C. Gates, “Robot reveals cathedral quake damage | Stuff.co.nz,” *Stuff*, 2011. [Online]. Available: http://www.stuff.co.nz/the-press/news/christchurch-earthquake-2011/5278552/Robot-reveals-cathedral-quake-damage. [Accessed: 22-Dec-2011].

[14] H. Donnell, “Robot always a ‘long shot’ - academic - Pike River mine blast - NZ Herald News,” *NZ Herald*, 2010. [Online]. Available: http://www.nzherald.co.nz/pike-river-mine-blast/news/article.cfm?c_id=1503000&objectid=10689548. [Accessed: 22-Dec-2011].

[15] G. S. Virk, Y. Gatsoulis, and M. Parack, “Mobile Robotic Issues for Urban Search and Rescue,” in *The International Federation of Automatic Control*, 2008, pp. 3098–3103.

[16] A. C. Schultz, “Robot Competitions Enhance Search and Rescue Technology While Raising Awareness and Attracting Funds,” *Technology*, no. September, pp. 26–33, 2002.

[17] R. M. Voyles, S. Member, and A. C. L. Member, “TerminatorBot : A Novel Robot with Dual-Use Mechanism for Locomotion and Manipulation,” pp. 1–8, 2004.

[18] R. Molfino, M. Zoppi, and L. Rimassa, “Rescue robot module with sliding membrane locomotion,” *Industrial Robot An International Journal*, vol. 35, no. 3, pp. 211–216, 2008.

[19] S. Hobby, “Sandia Labs’ Gemini-Scout robot likely to reach trapped miners ahead of rescuers – Sandia Labs News Releases,” 2011. [Online]. Available: https://share.sandia.gov/news/resources/news_releases/miner-scou/. [Accessed: 22-Dec-2011].

[20] D. J. Spero, “Department of Electrical and Computer Systems Engineering Technical Report A Review of Outdoor Robotics Research A Review of Outdoor Robotics Research,” 2004.

[21] H. Wang, Zhelong, Gu, “A review of locomotion mechanisms of urban search and rescue robot,” *Industrial Robot An International Journal*, vol. 34, no. 5, pp. 400– 411, 2007.

[22] J. Meltzer, R. Gupta, and S. Soatto, “Simultaneous localization and mapping using multiple view feature descriptors,” *2004 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) (IEEE Cat. No.04CH37566)*, vol. 2, pp. 1550–1555.

[23] D. Schleicher, L. M. Bergasa, R. Barea, E. Lopez, and M. Ocafia, “Real-Time Simultaneous Localization and Mapping using a Wide-Angle Stereo Camera and Adaptive Patches,” pp. 2090–2095, 2006.

[24] S. Thrun and D. Fox, “A Real-Time Algorithm for Mobile Robot Mapping With Applications to Multi-Robot and 3D Mapping,” no. April, 2000.

[25] M. Montemerlo, S. Thrun, D. Koller, and B. Wegbreit, “FastSLAM : A Factored Solution to the Simultaneous Localization and Mapping Problem,” pp. 593–598, 2002.

[26] S. Di Tore, F. D’elia, P. Aiello, N. Carlomagno, and M. Sibilio, “Didactics, movement and technology: new frontiers of the human-machine interaction.,” *Journal of Human Sport & Exercise*, vol. 7, no. 1, pp. S178–S183, Jan. 2012.

[27] D. J. MCFARLAND and J. R. WOLPAW, “Brain-Computer Interfaces for Communication and Control.,” *Communications of the ACM*, vol. 54, no. 5, pp. 60–66, May 2011.

[28] I. Sample, “Brain implant allows paralysed woman to control a robot with her thoughts,” *The Guardian*, 2012. [Online]. Available: http://www.guardian.co.uk/science/2012/may/16/brain-implant-paralysed-woman-robot-thoughts. [Accessed: 24-May-2012].

[29] “da Vinci® Surgery - Features.” [Online]. Available: http://www.davincisurgery.com/davinci-surgery/davinci-surgical-system/features.html. [Accessed: 28-May-2012].

[30] Anon., “The Advantages of Tethering — Robotics Trends,” *Robotics Business Review*, 2010. [Online]. Available: http://www.roboticsbusinessreview.com/blogs/view/the-advantages-of-tethering. [Accessed: 23-Dec-2011].

[31] Wifi Alliance, “Discover and Learn | Wi-Fi Alliance,” 2012. [Online]. Available: http://www.wi-fi.org/discover-and-learn. [Accessed: 04-Jun-2012].

[32] SIG, “Bluetooth Basics,” 2012. [Online]. Available: http://www.bluetooth.com/Pages/Basics.aspx. [Accessed: 04-Jun-2012].

[33] ZigBee Alliance, “ZigBee Specification Overview,” 2012. [Online]. Available: http://www.zigbee.org/Specifications/ZigBee/Overview.aspx. [Accessed: 04-Jun-2012].

[34] IRobot, “Ground Robots - 510 Packbot,” *iRobot Corporate*, 2011. [Online]. Available: http://www.irobot.com/gi/ground/510_PackBot. [Accessed: 22-Dec-2011].

[35] IRobot, “Ground Robots - 710 Warrior,” *iRobot Corporate*, 2011. [Online]. Available: http://www.irobot.com/gi/ground/710_Warrior/. [Accessed: 22-Dec-2011].

[36] E. Ackerman, “Japan Earthquake: iRobot Sending Packbots and Warriors to Fukushima Dai-1 Nuclear Plant - IEEE Spectrum,” *IEEE Spectrum*, 2011. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/industrial-robots/irobot-sending-packbots-and-warriors-to-fukushima. [Accessed: 22-Dec-2011].

[37] E. Rohmer, T. Yoshida, K. Nagatani, and S. Tadokoro, “Quince : A Collaborative Mobile Robotic Platform for Rescue Robots Research and Development,” in *The 5th International Conference on the Advanced Mechatronics*, 2010, pp. 225 – 230.

[38] K. Ohno, T. Kawahara, and S. Tadokoro, “Development of 3D laser scanner for measuring uniform and dense 3D shapes of static objects in dynamic environment,” in *IEEE International Conference on Robotics and Biomimetics*, 2008, pp. 2161 – 2167.

[39] K. Hatazaki, M. Konyo, K. Isaki, S. Tadokoro, and F. Takemura, “Active scope camera for urban search and rescue,” in *2007 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2007, pp. 2596–2602.

[40] Anon., “Swarmanoid project,” *Swarmanoid*, 2010. [Online]. Available: http://www.swarmanoid.org/swarmanoid_hardware.php. [Accessed: 23-Dec-2011].

[41] E. Ackerman, “Swarmanoid Robot Teams Up with Itself to Steal Your Books,” *IEEE Spectrum*, 2011. [Online]. Available: http://spectrum.ieee.org/automaton/robotics/artificial-intelligence/swarmanoid-robot-teams-up-with-itself-to-steal-your-books. [Accessed: 23-Dec-2011].

[42] E. Ackerman, “Swarmanoid Robot Teams Up with Itself to Steal Your Books - IEEE Spectrum,” *IEEE Spectrum*, Aug-2011.

[43] D. A. Williamson, “The Development of a ‘ Mother ’ Agent for a Hierarchical Multi-Robot Urban Search and Rescue System,” Victoria University of Wellington, 2007.

[44] B. Howarth, “Rubble-Bot Design,” 2006.

[45] Digi International, “XBee ® / XBee-PRO ® ZB RF Modules Digi International Inc .,” 2010.

[46] R. Grossmann, J. Blumenthal, F. Golatowski, and D. Timmermann, “Localization in Zigbee-based Sensor Networks,” in *1st European ZigBee Developers Conferences*, 2007.

[47] Digi International, “X-CTU,” 2012. [Online]. Available: http://www.digi.com/support/productdetail?pid=3352. [Accessed: 04-Jun-2012].

[48] NZ Standards Council, “NZS 4121:2001 Design for Access and Mobility - Buildings and Associated Facilities,” *New Zealand Standards Council*, 2001. [Online]. Available: http://shop.standards.co.nz/scope/NZS4121-2001.scope.scope.pdf. [Accessed: 22-Sep-2012].

[49] Jaycar, “12V DC Reversible Gearhead Motors - 160RPM.” [Online]. Available: http://www.jaycar.co.nz/productView.asp?ID=YG2738&keywords=12v+gear&form=KEYWORD. [Accessed: 21-May-2012].

[50] R. Thomas, “Omni-Directional Mobile Platform for The Transportation of Heavy Objects,” 2011.

[51] Microchip, “dsPIC30F2010 Specifications,” 2010. [Online]. Available: http://www.microchip.com/wwwproducts/Devices.aspx?dDocName=en010329. [Accessed: 14-Mar-2012].

[52] Anon., “OSMC project info,” *Robot Power*, 2011. [Online]. Available: http://www.robotpower.com/osmc_info/. [Accessed: 04-Jan-2012].

[53] Anon., “HIP4081A Full Bridge Driver with Charge Pump,” *Intersil*, 2011. [Online]. Available: http://www.intersil.com/products/deviceinfo.asp?pn=HIP4081A. [Accessed: 04-Jan-2012].

[54] Anon., “IRF1407 Datasheet,” *Internation Rectifier*, 2011. [Online]. Available: http://www.irf.com/product-info/datasheets/data/irf1407.pdf. [Accessed: 04-Jan-2012].

[55] Anon., “ACS712 Hall Effect Based Linear Current Sesnor IC,” *Allegro Microsystems*, 2011. [Online]. Available: http://www.allegromicro.com/Products/Current-Sensor-ICs/Zero-To-Fifty-Amp-Integrated-Conductor-Sensor-ICs/ACS712.aspx. [Accessed: 04-Jan-2012].

[56] HobbyKing, “HobbyKing R/C Hobby Store : 900MHZ 1500mW Tx/Rx & 1/3-inch CCD Camera NTSC,” 2011. [Online]. Available: http://www.hobbyking.com/hobbyking/store/__13443__900MHZ_1500mW_Tx_Rx_1_3_inch_CCD_Camera_NTSC.html. [Accessed: 20-Mar-2012].

[57] Mindkits, “Pan-Tilt Bracket,” 2011. [Online]. Available: http://www.mindkits.co.nz/store/mechanics/pan-tilt-bracket. [Accessed: 22-Mar-2012].

[58] Hobby Hanger, "DN Power HTX3700-4S 3700mAh 30C," 2011. [Online]. Available: http://www.hobbyhangar.co.nz/4s-lipo/power-htx37004s-3700mah-p-3523.html. [Accessed: 28-Mar-2012].

[59] RoBoard, "RoBoard -- The Heart of Robotics," 2011. [Online]. Available: http://www.roboard.com/RB-110.htm. [Accessed: 19-Feb-2012].

[60] Texas Instruments, "TI Launchpad," 2012. [Online]. Available: http://e2e.ti.com/group/msp430launchpad/w/default.aspx. [Accessed: 29-Feb-2012].

[61] Texas Instruments, "MSP430G2553." [Online]. Available: http://www.ti.com/product/msp430g2553. [Accessed: 29-Feb-2012].

[62] Anon., "MQ-6 LPG Sensor Datasheet." [Online]. Available: http://www.sparkfun.com/datasheets/Sensors/Biometric/MQ-6.pdf. [Accessed: 27-Nov-2011].

[63] National Seminconductor, "LM94022 - 1.5V, SC70, Multi-Gain Analog Temperature Sensor with Class-AB Output." [Online]. Available: http://www.national.com/pf/LM/LM94022.html#Overview. [Accessed: 16-Feb-2012].

[64] HobbyKing, "HobbyKing R/C Hobby Store : Zippy Flightmax 1000mAh 2S1P 15C," 2011. [Online]. Available: http://www.hobbyking.com/hobbyking/store/__6472__Zippy_Flightmax_1000mAh_2S1P_15C.html. [Accessed: 22-Feb-2012].

[65] Sparkfun, "XBee 2mW Wire Antenna - Series 2 (ZigBee Mesh)." [Online]. Available: http://www.sparkfun.com/products/10414. [Accessed: 01-Mar-2012].

[66] D. Norman, *The Design of Everyday Things*. Doubleday Business, 1990, p. 272.

[67] Mindkits, "Thumb Joystick," 2011. [Online]. Available: http://www.mindkits.co.nz/store/components-ics-breakout-boards/Thumb-Joystick. [Accessed: 20-Mar-2012].

[68] Sparkfun, "XBee Explorer USB - SparkFun Electronics." [Online]. Available: http://www.sparkfun.com/products/8687. [Accessed: 20-Mar-2012].

[69] NZ Standards Council. (2001). NZS 4121:2001 Design for Access and Mobility - Buildings and Associated Facilities. *New Zealand Standards Council*. Retrieved September 22, 2012, from http://shop.standards.co.nz/scope/NZS4121-2001.scope.scope.pdf

# Appendices

The appendices are located on the companion CD included with this work. It contains a record of the designs, software, and all published papers.