A RoboCup Rescue Robot

Joshua Marshall

Supervisor: Prof Rick Middleton

20 October 2004

A thesis submitted in partial fulfilment
of the requirements for the degree of
Bachelor of Engineering in Computer Engineering
at
The University of Newcastle, Australia.

Abstract

RoboCup Rescue is an annual international competition designed to advance the state of the art in search and rescue robotics. To date the University of Newcastle has been involved with the associated RoboCup robotic soccer competition, achieving third place in the Legged League for the last three years.

The goal of this project is the design and construction of a robot to be used as an entry for the University in RoboCup Rescue 2005. It was designed to be a tracked vehicle with extensive sensors and a powerful onboard computer. Implementation of this design required mechanical, electrical and software engineering skills.

The robot will be developed to the point where it can be teleoperated, which is when a remote user is able to view all of the sensory data and send movement commands to the robot.

List of Contributions

- Research into RoboCup Rescue background and requirements.
- System design, including selection and integration of components.
- Mechanical design (with assistance from Russell Hicks).
- Electronic design of circuit boards for
  - a motor drive using a H-bridge,
  - an I^2C bus interface, and
  - microcontroller development.
- Construction, testing and debugging of these boards.
- Software design, testing and implementation of
  - a custom Linux installation – supporting all the required hardware,
  - embedded microcontroller software to interface with motors and sensors,
  - software to allow control of the robot over a network, and
  - development of a graphical user interface for monitoring and control.

Joshua Marshall

Prof Rick Middleton

Acknowledgements

This project would not have been possible without the assistance of many people.

Thanks go to Professor Rick Middleton for supervising my project. I am thankful for being given the opportunity to develop the University's entry for RoboCup Rescue. His support has been invaluable, whether it be in guidance on design issues or help in troubleshooting my mistakes.

The Robotic Research Infrastructure Block provided the funding from grants to purchase the many bits and pieces that made up the final robot.

Russell Hicks from the Electrical Engineering technical staff has been carrying out the mechanical construction. I am grateful for his assistance and expertise in this area.

My thanks also go to the other members of the technical staff, particularly Peter Turner, Steve Mitchell, Roy Murcutt, Ian Powell and Chris Jones, who all assisted me with my many little questions and offered helpful suggestions.

Dianne Piefke handled all the component orders from companies around the world, and handled all the problems with communication and shipping.

Stefano Carpin, from the International University Bremen in Germany, kindly sent his documentation and sample code for the range scanner.

Giuliana, thank you for your support in every aspect of life.
I couldn't do it without you.

— Josh

Contents

List of Figures

Chapter 1

Introduction

RoboCup is a series of international competitions and conferences, designed to advance the state of the art in intelligent robotics. Simulated real-life problems are proposed and solutions are attempted by teams from universities and companies from around the world. The current main focus is on robotic soccer, with the goal of developing a team of robots which are capable of defeating the human world champions by the year 2050.

While attempting to solve the problems presented by a soccer game results in advances in locomotion, machine vision and decision-making artificial intelligence, the end result is not directly applicable to a wider context. In an attempt to remedy this, the associated competition of RoboCup Rescue takes many of these advances and applies them to a practical problem.[1] The aim is for robots to navigate a series of artificial disaster arenas, which simulate collapsed buildings in the aftermath of an earthquake. While doing this, it is necessary that the arena be mapped and any simulated 'victims' located on this map for later rescue. The rationale behind this problem can be found in [1, 2]. In [3] Murphy discusses the challenge of urban search-and-rescue. He sees it as being one of the more compelling reasons to work on robotic AI, due to its humanitarian nature.

The rules for the competition state the vision of RoboCup Rescue:

> *When disaster happens, minimize risk to search and rescue personnel, while increasing victim survival rates, by fielding teams of collaborative robots which can:*
>
> - *autonomously negotiate compromised and collapsed structures*
> - *find victims and ascertain their conditions*

[1] See the RoboCup Rescue webpage at http://robotarenas.nist.gov/competitions.htm for current information on the competition.

- *produce practical maps of their locations*
- *deliver sustenance and communications*
- *identify hazards*
- *emplace sensors (acoustic, thermal, hazmat, seismic, ...)*
- *provide structural shoring*

... allowing human rescuers to quickly locate and extract victims. [4]

Success in mapping and locating victims in collapsed buildings will hopefully lead to a robot which can be used in real disaster situations. While this has been done already, with the highest profile case being the use of the CRASAR robots[5] after the collapse of the World Trade Centre Twin Towers in 2002, there are many limitations still to be overcome. These robots are currently remote controlled or *teleoperated*, over a communications link. The number of robots that can be used is limited as they require human operators, whether this is one operator per dozen robots or a dozen operators per robot. The use of human operators also requires communications links, wired or wireless, which restrict the movement of the robot. The latency of the link and human response time do not permit real-time operation. The need for costly manpower and training of operators is a significant issue.

The solution to this problem is to provide greater autonomy for the robots, which is done by using artificial intelligence methods on an onboard computer. While full autonomy is not yet possible, this may be undesirable, as this removes human control altogether. Rather, it is suggested that the important areas to autonomise are basic locomotion tasks and basic object recognition. The goal is to be able to monitor the robot and give high level commands, such as "head down that hallway", removing the need for fine-grained, individual movement control.

Attempting RoboCup Rescue differs markedly from research in industrial robotics, where there is a narrower problem domain. Rather than optimising a highly structured problem with clearly defined performance objectives, the goal is to perform 'well' in a loosely structured environment. The competition between teams and the lessons learned from attempts at this problem is what advances the knowledge in the field.

1.1 Previous efforts

Of the previous year's entries in the RoboCup Rescue competition, approximately half have been wheeled vehicles and half have been tracked [6]. A few alternative platforms have been tried, such as floating blimps or Sony AIBO robotic dogs, but these methods present

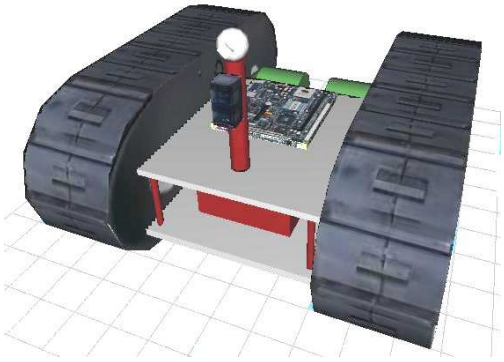


Figure 1.1: RoboCup Rescue Robot design

additional challenges and have not been successful to date. A variety of sensors have been used, such as sonar, video cameras, range finders, bumpers, and microphones. Sizes range from 100mm square up to 500mm square. Most of these robots are teleoperated over wireless links, which is to say that they have very little autonomy. No teams have done particularly well overall in the competition to date, although some have performed well at particular tasks, such as mapping. This does not appear to be an inherent limitation of the hardware designs, but rather of the controlling software.

By definition, the conditions in a disaster situation cannot be accurately predicted or controlled. In particular, wireless or wired links cannot be depended upon. With this in mind, over the next few years the competition will begin to include periods of radio blackout when teleoperation will not be possible, encouraging a move towards greater autonomy in the design of these robots.

This project will be an attempt to construct a robot that will be able to participate in RoboCup Rescue, with this need for autonomous behaviour as a central design criterion. Once the hardware design has been completed, the focus will be on providing for semi-autonomous behaviour. To this end, the control software will be designed such that it is independent of the platform it is running on, whether this be the development workstation or on the robot itself. The end goal is to provide a ready platform for a team to develop control software for a successful entry into RoboCup Rescue 2005.

1.2 Report outline

The remainder of this report will discuss the design and implementation of this RoboCup Rescue robot. The chapters correspond closely to the various stages in this process.

Chapter 2 will introduce the details of the RoboCup Rescue competition, and discuss

the necessary characteristics of a competing robot. Chapter 3 covers the design of the robot and the components chosen for each of the robot's subsystems, along with the construction techniques needed. Chapter 4 goes into detail on the process of implementing the various subsystems of the robot, which includes software, circuit designs and similar considerations. Chapter 5 will discuss the current state of the project, and what is expected to be achieved by the time of the Open Day presentation. Finally, Chapter 6 will state the conclusions reached by carrying out this project, and propose future directions for the University's involvement in RoboCup Rescue. The appendices consist of the circuit board schematics and layouts, and software listings.

Chapter 2

Background and Overview

The previous chapter gave an introduction to RoboCup Rescue on a conceptual and historical level. In order to build a robot capable of competing, it is necessary to first examine the specifics of the competition and design a robot with these in mind.

2.1 RoboCup Rescue in depth

The basic premise of RoboCup Rescue is for the competitor's robot(s) to be placed in a simulated disaster zone or arena. The robot will navigate the arena searching for simulated human 'victims.' A map is to be produced, displaying the structural features of the arena and the locations of the victims. Scoring is based on the quality of this map, with points being deducted for various violations of the rules. The following sections describe each of these components of the competition.

2.1.1 Arenas

The test arenas used in the competition are specified by the Intelligent Systems Division of the US National Institute of Standards and Technology. The goal behind standardising these arenas is *'to improve our ability to measure the capabilities of mobile robots.'* [7] The indoor arenas are divided into three categories based on increasing levels of complexity; yellow, orange and red. The former is a simple 2-D maze with no flooring problems, while the latter is an unstructured 3-D environment with shifting rubble and confined spaces. Figure 2.1 shows the increasing difficulty between the yellow, orange and red arenas.

(a) Yellow (b) Orange (c) Red

Figure 2.1: NIST standard rescue arenas. Taken from [7].

2.1.2 Human victims

Within the arenas are located a number of simulated human 'victims'. These are dolls or mannequins, thus having human forms. They are designed to imitate some of the characteristics of real humans. They may be making noise or moving, they are heated to simulate body heat, and they emit CO_2 gas.

These victims may be located anywhere within the arenas, ranging from lying on the floor to totally entombed within rubble. They also range in simulated 'age' from infant to adult. Each victim has a 'victim tag' which must be located and scanned by the robot. These contain victim data which must be presented on the map.

2.1.3 Rules & Regulations

Since RoboCup Rescue is a competition, there are a number of rules which dictate the final scoring for the comparison between robots. The rules are intended to deter the use of multiple robots and prevent false victim identifications, while promoting a multi-sensor approach. The accuracy of the final map of the arena, correctly identifying victim locations, classifying these victims and scanning their tags contribute positively towards the score, while collisions with the arena or victims subtract from it. An example of an ideal map is shown in Figure 2.2, which shows the walls and structural features of the arena, along with the victims locations and corresponding states.

Chapter 1 presented the vision for RoboCup Rescue. The current rules for the competition cover most of the first three points; negotiate compromised and collapsed structures, find victims and ascertain their conditions and produce practical maps of their locations. The remaining points will be of minor difficulty if these first tasks can be implemented with autonomous behaviour on behalf of the robot. However, this is not practical at this point in time, as current AI techniques don't provide for such a high degree of decision making and independence.

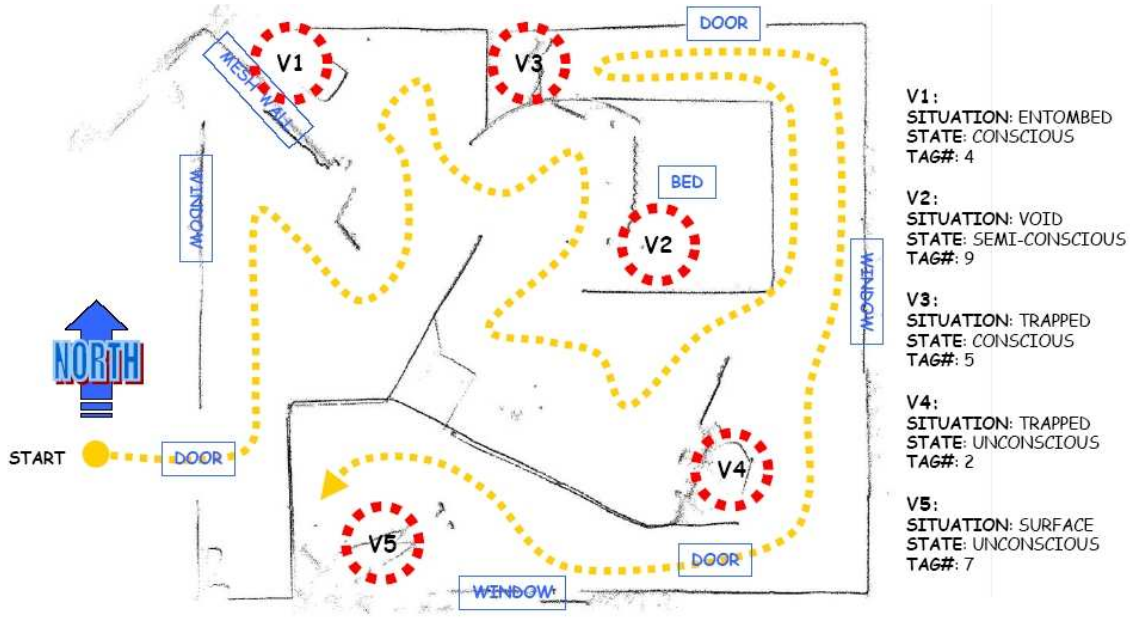


Figure 2.2: Ideal arena map, with marked walls and victims. Taken from [7].

2.2 System Requirements Overview

This project is split into two major goals. The first is to build a hardware platform for the robot with the requirements of the previous section in mind. This design will support teleoperation and eventually some degree of autonomous control. The second goal is to develop some software for basic teleoperation, which will display the status of the robot's sensors to a human operator and allow for the control of the robot's movement.

The design encompassed the whole robot, from mechanical construction up to the enabling firmware and communications software. In order to build a robot to be entered into this competition, the design was created with the competition guidelines in mind. It is expected that this design will be sufficiently expandable to encompass all the aspects of the vision for RoboCup Rescue when these are added to the actual competition.

Figure 2.3 gives a view of the major components in the system design. The robot requires sensors to perceive the world, and motors to move around in it. A control system, whether simple or complex, provides an interface to this sensing and movement. A power source is needed for cable-free operation and a physical structure for combining all these components into a single unit. The remote data link is needed for control and/or supervision.

The remainder of this chapter will discuss the areas of engineering knowledge that need to be applied to this design, and the particular aspects with which we will be concerned.

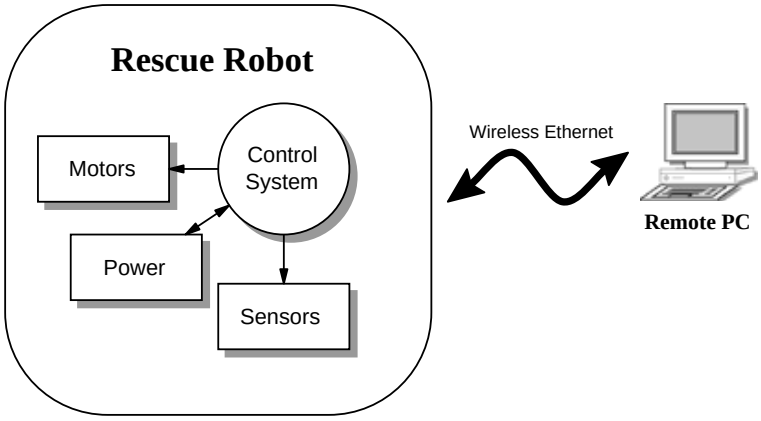


Figure 2.3: System diagram

2.2.1 Safety issues

This robot will be relatively large and heavy. It will also eventually move around in the environment according to decisions made by artificial intelligence algorithms, which could be faulty. Under these circumstances, it is critical that safety issues are considered, to prevent any loss of control. A kill switch will be fitted to the robot which will cause the immediate powerdown of all systems, to be used as a last resort. In the design of the individual components, steps will be taken to ensure that "runaway" subsystems will not cause a total loss of control. This will include the installation of watchdog timers on all control systems, both high and low levels, to initiate a system stop upon the loss of communication or detection of a fault. Chapter 4 will discuss the implementation of these safety mechanisms.

It is also important that those who will work with the robot be aware of these safety issues, and also those involving the manual transportation or movement of the final robot.

2.2.2 Electrical

The electrical components of the robot form the major part of the design of this project. The electrical subsystems making up the robot include sensor interfacing, a power supply, motor drive and a computer system for the overall control.

Sensors

The use of sensors in robotics requires the proper selection of devices for the application domain and interfacing these to the control systems. The particular sensor required for an application depends on the signal to be measured. See [8, ch 7] for an overview of the different kinds of sensors that can be used.

The RoboCup Rescue rules necessitate the sensing of visual and audio cues, temperature

and carbon dioxide levels. In addition, some mechanism is necessary for mapping the arena and locating the 'victims'. This latter problem is the more difficult one. We will use a video camera, a range scanner and odometry obtained from the motors to attempt this task. To detect the former, various off-the-shelf sensors can be used, such as a standard microphone, noncontact thermometer and CO_2 sensor. These will use various interfaces to connect to the main computer system, with some custom electronics required.

Power Electronics

The principal applications of power electronics in the project are for the power supplies and controlling the drive motors.

The various electronic components of the robot will require differing supply voltages. The robot will be powered from a rechargeable battery for mobility. To obtain the other voltages requires DC-DC conversion. The computer system will be powered from an off the shelf power supply.

The motors will be driven using MOSFET H-bridges. Pulse width modulation (PWM) drive waveforms will be provided by a microcontroller. This allows for continous speed control of a motor using only a single bit digital signal. This will allow the robot to move forward and backward and turn along arbitrary curves, which includes turning in place. See [8, §6.7] for details on how this is done.

2.2.3 Mechanical

The environment that the robot is to be subjected to is varied and unpredictable. It is possible that falls from some height may occur. Therefore the mechanical construction of the robot needs to be solid, and the individual components will need to be securely attached and resistant to collision and vibration.

The method of locomotion of previous contestants has been mostly either wheeled or tracked. A wheeled design that is capable of surmounting obstructions, steps and inclines is very complex. Therefore this design will use a tracked vehicle, which is simpler to implement and analyse.

These tracks will be driven by electric motors and appropriate gearing, which shall allow for a top speed of a moderate walking pace. The more important characteristic is sufficient torque to climb the expected inclines and obstacles.

Since the components of the robot will be numerous, it is necessary that there be sufficient room to mount all of these, and devices such as a camera and range finder will need

to be mounted high on the chassis to allow for an unobstructed view.

As was mentioned earlier, the transportation of the robot involves some safety issues due to weight and bulk. It is essential that all heavy parts be firmly attached and for the robot to have an easy means of being lifted.

2.2.4 Computer & Software

Once the hardware systems have been implemented, it is the software that determines the capability of the robot. This is the open ended side of the design, as complex artificial intelligence routines will be required in order to manoeuvre around the arenas and locate the 'victims'.

Many of the previous entries for RoboCup Rescue have used small microcontrollers as the main robot control system. Unfortunately, doing this reduces the amount of autonomy available, as these devices do not provide enough processing power to interpret the sensory data, particularly vision. They must therefore rely on a human operator or some other remote processing capability. To avoid this problem, this robot will incorporate a full x86-compatible computer onboard.[1] Linux was chosen as the underlying operating system, as it is an open platform, free of charge, and a wealth of information exists on all aspects of development using this OS. The computer can then be programmed using standard software development techniques and existing control and AI libraries can be reused.

It is planned for there to be three major software components to the system. These are the embedded software in the peripheral controllers for the sensors and motors, the low level interfacing code in the onboard PC and the higher level control and intelligence software.

Interfacing

The sensors and motors will be interfaced using several microcontrollers. These devices will be connected to the main computer using a serial bus. The software for these will require interfacing to the bus and interacting with the external, mostly analog components. It will be written in assembly language, and manage such tasks as analog to digital conversion and PWM generation for motor control.

The low level interfacing code will run on the PC and gather together all the sensory input to be sent to the control layer. This will be data from the microcontrollers mentioned earlier, a video stream from the camera and range data from the range finder. It will receive motor control commands from the control layer and send these through the serial bus to

[1] For a project with a similar approach, see [9].

the microcontroller. Any calibration or signal processing will be done in this layer, which will be written in C and run as a process on the main computer.

Control

The control system software will receive the sensory data and use it to implement the functions of navigation, mapping and victim location and identification. In the initial stages of development this will present the sensory data to a human operator, who can then control the robot. It can be written in any higher level language, currently targeted to be Python, which is a high-level scripting language, often referred to as executable pseudocode. It is simple to learn and allows for rapid development. Code can be updated while it is still running, and any computationally intensive tasks such as vision can be easily recoded in C for speed.[2]

The control software will be able to run on any computer which is connected through a network (wired or wireless) to the robot or on the robot itself. The former will allow for ease of implementation and testing during initial development. As software modules such as vision are developed, these can be moved to the robot's computer for execution. This will give some degree of autonomy once a significant portion of the control software is running on the robot.

This abstraction will be obtained by the interfacing code communicating with the control systems through the networking layer of the operating system. Once it is desired to run the code on the robot, the data will be still be taken from the network interface, but in a loopback mode.

[2]See `http://www.python.org`

Chapter 3

System Design

This chapter describes the components chosen for each of the robot's subsystems, within the limitations set by the rules for the competition. Figure 3.1 is an overall block diagram of the components of the robot. The following sections will discuss each of the components chosen for the robot, along with the motivation for each particular choice.

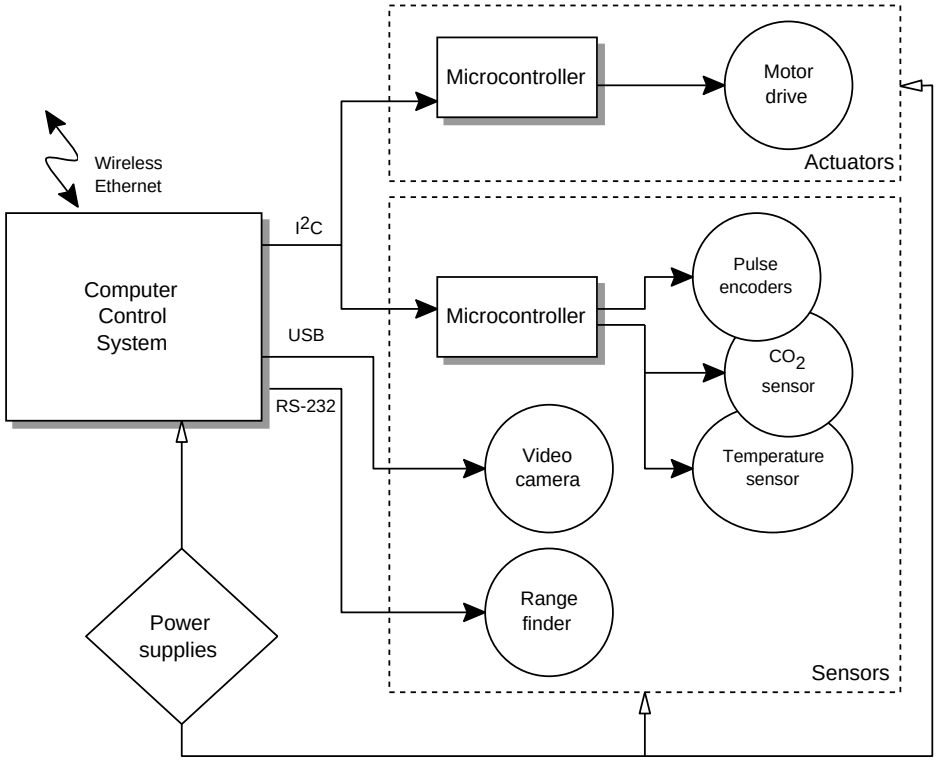


Figure 3.1: Robot system diagram

Figure 3.2: EPIA M10000 system platform

3.1 Hardware Components

3.1.1 Computer control system

As was mentioned in the previous chapter, this robot will have a full x86-compatible computer on board. The VIA EPIA M10000 platform has been chosen for this task, as it provides processing power sufficient for vision processing and basic artificial intelligence methods with its 1 GHz CPU, low power consumption figures of 25W peak and 10W idle, and a small form factor of 170mm square. (See Figure 3.2)

The other components of the computer system include 512 MB of double data rate RAM, a 20 GB 2.5″ hard drive and a Netgear WAG311 wireless Ethernet card. These specifications ensure that computing power and storage should not be a limiting factor in designing advanced AI algorithms. The computer can interface to a network either through a wired Ethernet cable or via 802.11b/g/a[1] wireless Ethernet.

The computer has been chosen to run Linux as the operating system, as doing so will allow easy access to the underlying hardware. Since the sensors will be using a wide range of interfaces; the I^2C bus[2], RS-232 serial, USB[3] and the onboard audio; this is an important factor in the software design. The only platform dependent code will be this interfacing code. The various modules will be implemented as separate processes, and communicate using the network interface. For more details on Linux and other software, see the following chapter.

[1]The rules for RoboCup Rescue this year dictate that only 802.11a can be used during the competition. This is to avoid conflict with the other RoboCup competitions, which are using 802.11b, in the 2.4 GHz band. 802.11a is in the 5 GHz band, and provides a maximum of 54 Mbps bandwidth.

[2]inter-integrated circuit bus

[3]Universal Serial Bus

3.1.2 Sensors

The RoboCup Rescue rules specify that when identifying the 'victims', maximum points will be derived from making the identification using as many as possible of the following: visual means (human form, clothing, reflective tape, locator strobe, victim tag), detecting CO_2 emissions, body heat, audio (tapping, voice, locator alarm) and motion (moving arms and fingers).

To this end, we will be equipping the robot with sensors to cover all of the possible cues. All the vision and motion signals will be detected through the use of a video camera and processing of the resultant images. A microphone will be connected to the system's audio input to detect the sound cues. A CO_2 gas sensor and infrared thermometer cover the remaining possibilities. As well as detecting victims, the video camera along with an additional LED range finder will be used for navigation and mapping the arenas. We will obtain odometry data from pulse encoders on the drive axle, but since a significant amount of slippage[4] may result during turns, an onboard electronic compass will be used to determine an absolute bearing.

Video camera

Vision is the most demanding component of most autonomous robotic devices. This is in terms of computational power required as well as software complexity. This topic is discussed in [8, §8.5.2].

The digital video camera to be used is a Logitech QuickCam Pro 4000 USB device. (See Figure 3.3) This is one of the more advanced 'webcams', with a capture resolution of 640 by 480 pixels at 15 frames per second (fps).[5] It was chosen for these capture capabilities along with the ease of interfacing to the computer system over the USB bus. It will be mounted to a mast at the highest point on the robot, which is expected to be approximatedly 30cm off ground level. Interfacing to this camera is done entirely through software and will be covered in the following chapter.

Range finder

A LED-based range finder ([8, §7.12.2]) is to be mounted below the camera to supplement the video imagery with range data for mapping and navigation. The device to be used

[4]The treads on tracks need to *slip* during turns, since the opposing sides travel different distances. This also results in higher drive current to the drive motors when slippage is occurring.

[5]Or alternatively; 320 by 240 pixels at 30 fps.

Figure 3.3: Video camera

Figure 3.4: Range Finder

is a Hokuyo PB9-11, which has a range of 0–3m. (See Figure 3.4) The readout from this range finder gives the distance in a 162° arc, in 1.8° segments. It interfaces to the computer through a serial RS-232 connection and is powered off a 24V supply. The data will be used to provide a map of what is immediately in front of the robot, and integrated into a map of the whole arena. This is also useful to determine the paths that the robot can traverse, by finding the width of a particular passage. This device will be mounted directly below the video camera, allowing for easier integration of the visual and range data.

Audio

The video camera also contains a small microphone, which can be used for detecting any audio cues from the victims. Should this microphone be of low sensitivity, an electret microphone and amplifier circuit can be added larger, which would connect through the computer system's onboard audio interface.

CO_2 sensor

In order to detect the CO_2 emissions that will be given off in order to simulate a victim's breathing, the robot will have a CO_2 gas sensor. This will be an MG811 device which responds to local carbon dioxide concentrations in the range of 350-10000 ppm[6]. (See Figure 3.5) The output signal from the device is buffered and amplified before being fed into an analog to digital converter on a microcontroller.

Heat sensor

A noncontact infrared thermometer will be mounted near to the video camera in order to provide a temperature readout. The rules state that a victim will have a temperature of

[6]parts per million

Figure 3.5: CO_2 gas sensor

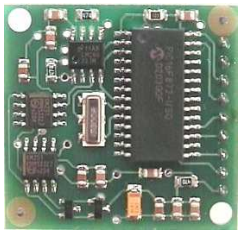


Figure 3.6: Electronic compass

37°C, which means that the thermometer will only be of use when the ambient temperature is much lower than this value. The device has an accuracy of ±2°C. The area over which the temperature is taken will be able to be correlated with the vision data.

Odometry

In order to accurately map an arena we require some way of determining the paths the robot has traveled, along with the range and video data. To this end we need to keep account of the robot's movements through velocity measurement. This data can then be integrated to give the displacement. Traditionally this is a very difficult task, as the errors and noise in velocity data measurements accumulate in the process of the integration to obtain distance values. In order to avoid this, we will provide several sensors to collect different forms of odometry data.

Pulse encoders ([8, §7.3.2]) will be provided on each drive axle. These measure the number of rotations of the axles through a small magnet and magnetic field sensor. When the robot is moving forward or backwards, without the tracks slipping, this system will provide accurate distance data.

During turns the rotation of the drive axles may not be a valid representation of the distance the tracks are traveling due to slippage. For this situation, an electronic compass is provided. This device interfaces via the computer's I^2C bus, and provides an 8-bit readout of the current magnetic field. In the absence of other magnets, this gives the direction of the robot relative to magnetic north. The device used is a Magnetic Compass CMPS03, which is accurate to $3 - 4°$. (See Figure 3.6)

When the above two signals are combined with the range data and processed vision information, accurate positioning should be possible. Some future extensions are possible, if this proves to not be the case, which are discussed in Chapter 6.

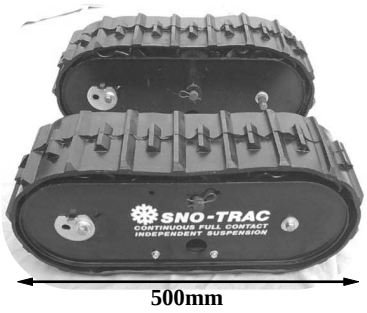


Figure 3.7: Tracks and treads

3.1.3 Locomotion

Locomotion will be provided through a pair of tracks and treads. These are surplus units that were intended for small snow plows, with two pairs to a vehicle. (See Figure 3.7) These tracks are 500mm in length, 220mm in height and 120mm wide. They will be driven using a pair of 200W electric motors. These are motors that are intended for use in small electric scooters. They have a rating of 2750 RPM under load, which will be stepped down using a chain and sprocket. The largest sprocket which can be mounted inside the tracks has 80 teeth. The motor has a drive sprocket with 11 teeth, giving a step down ratio of 1:7.3. This will result in a maximum speed of around 4 meters/second, which is faster than is desirable for safety reasons. The implementation chapter will discuss this issue further, including details on methods for limiting the maximum speed.

Using an H-bridge drive with PWM waveforms allows for smooth acceleration and movement. These control waveforms will be generated from a microcontroller connected to the I^2C bus.

3.1.4 Power

As the robot will be operating remotely, it is necessary that it carry its own power source. This will be supplied through the use of 2 sealed lead acid cells, each 12V. This gives a total power supply of 24 volts.

The power to be used is through several separate paths. The first is to the drive motors. The H-bridge drive will connect the batteries directly across the motors. Appropriate switching noise reduction measures will be used. The second path is to the computer system, through a small switching power supply which connects directly to the motherboard. This powers the computer, hard drive and network card. There is also an auxiliary 5 volt connector, which can be used to power some of the peripherals. The range scanner requires 24V, and it will be connected directly to the batteries. A linear regulator will be used to

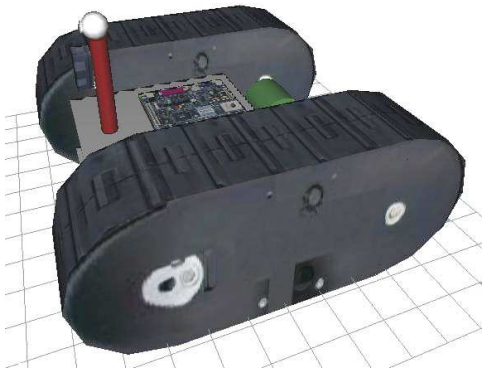


Figure 3.8: Chassis and component placement mockup

power the CO_2 sensor, which requires 6V.

3.2 Construction

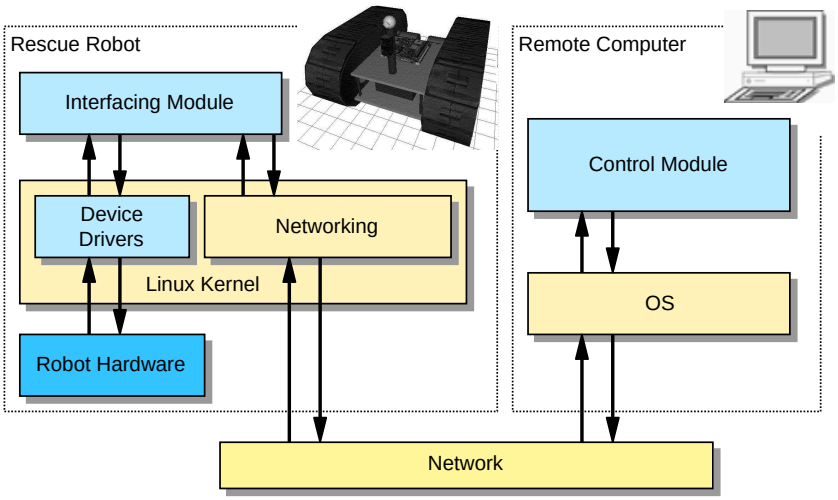
All of the above components are to be mounted on a chassis, which will be bolted between the tracks.

The preliminary design was to have the various components with be mounted on several levels. One level would contain the drive motors and electronics, along with the power supplies. Another will contain the computer system, whose largest parts are the motherboard and the hard drive. Finally a mast will be on top, with the video camera and range finder mounted on it. Figure 3.8 is a mockup of what the final robot should look like, omitting the smaller components. This design changed slightly in the final construction, with the motors being placed inside the tracks. Section 4.5.2 discusses the construction process.

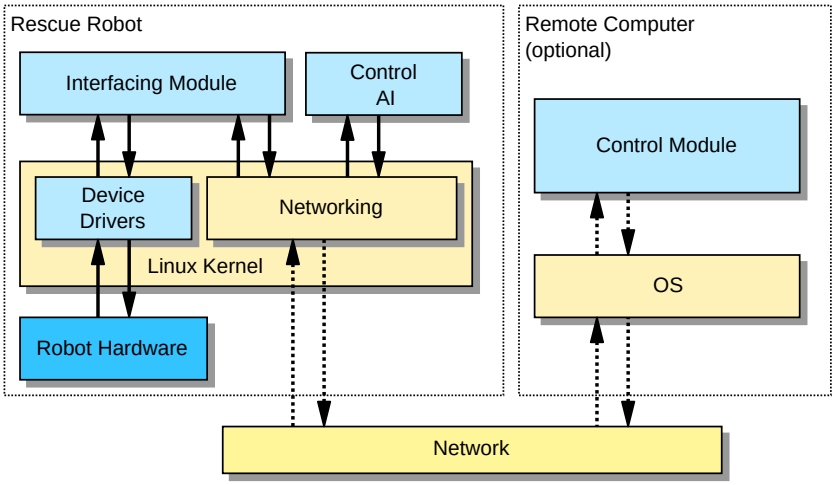
The overall dimensions of the robot are determined primarily by the tracks. The length of the robot will be 500mm, which is the length of the tracks. The width will be approximately 450mm, of which 240mm is the tracks. The widest other component is the motherboard at 170mm, and 20-250mm will be allowed for the mounting plates. The total height will be approximately 350mm, with only the camera and rangefinder on a mast raised over the top of the tracks.

3.3 Software Architecture Overview

There are three levels to the control of the robot; interfacing to the electronic components to produce raw sensory data, transforming this data into a usable form for human or machine interpretation, and higher-level monitoring and control. Figure 3.9 gives an overview of the interactions between these components. The first diagram from the figure shows the goal for



(a) Teleoperated



(b) Autonomous

Figure 3.9: Software architecture overview

this current project, with the sensory data being passed over the network to a monitoring computer. The second shows the ultimate target for RoboCup Rescue, where the robot performs most of the decisions and the remote computer is reduced to a monitoring and supervisory role.

In the initial design, the flow of sensory data is from the hardware, through the relevant device drivers and to the interfacing module. Here some basic processing occurs, such as converting between raw integers and the corresponding units, and then this data is sent over the network to the remote computer, where the control module receives it. Control commands are sent in the opposite direction. Although it is outside the scope of this project, the second diagram shows the addition of a control module to the robot and how this results in decreased control and data traffic over the link between the computers.

3.3.1 Data capture and transmission

Towards the goal of this project, the role of the software onboard the robot has two tasks. The first, interacting with the actual components, is carried out both by custom microcontrollers and by device drivers running under the operating system. The second, transmitting this data over the network, will be done by a few programs running on the computer. The following chapter discusses the implementation of both these tasks.

3.3.2 Remote operator interface

As has been stated previously, the goal of this project is to bring the design of the robot to a point where it can be remotely controlled. To this end, a program will be written for the operator's computer which will display all the sensory data in a graphical user interface. The operator will be able to control the robot using either the keyboard or a joystick.

Chapter 4

Implementation Details

This chapter discusses the details of carrying out the implementation of the design. The subjects covered include the implementation of safety mechanisms, the development of the various types of software, interfacing the sensors through hardware and software, electrical considerations and mechanical construction.

4.1 Safety mechanisms

We begin the discussion of the implementation details by covering the safety mechanisms used to avoid any damage to person or property. As was previously discussed, the large mass of the robot along with prototype software and possible autonomy may result in a loss of control. This, along with considerations pertaining to protecting the robot itself from other causes of damage, resulted in the following features in the design.

4.1.1 Kill switch

A wire will be strung across the back of the robot between two anchor points. This will connect directly to the battery. If this is pulled, the power will be shut off immediately. This is intended as a last resort in the case of loss of control, but it is anticipated to also be needed during development in the case of bugs in the software.

4.1.2 Watchdog timers

The microcontroller software controlling the motor drives will have a watchdog timer as a part of its central design. This requires that the controlling software send a 'keep-alive' signal at a set interval. Should this fail to be received, the drive system is shut down. This will be set to around a second, which will avoid loss of control caused by issues such as

infinite loops in the control software. A similar system will also be implemented in the remote control application on the remote computer. The operator must constantly select a command to move the robot, otherwise movement commands will not be sent. This should prevent a momentary distraction from causing damage.

4.1.3 Fuses and current monitoring

A fuse block will be connected between the batteries and the motors and computers. The fuses will be rated at 150% of the maximum current draw. Both motors, the range scanner and the computer will have individual fuses. This will lessen the chance of any short-circuit damaging the electronic components.

While not implemented at this time, a future addition will be current monitoring to detect stall conditions. This is useful for more than safety issues, and is discussed in section 4.4.3. Once this is implemented, it should prevent any chance of burning out the motors.

4.1.4 Chassis construction

The chassis for the robot will be built in such a way that no fragile components will be exposed. The underside of the robot will be a steel plate protecting the motherboard, and the only protruding devices above the tracks will be the camera and range scanner. The two tracks will be joined by a solid steel plate, providing for rigidity and ease of transportation.

4.2 Software

The computer system was chosen with running Linux in mind, since this operating system is open source and well documented in most aspects. This simplifies future development for the robot, since code can be developed in any programming language and can be tested and debugged on the development computer before executing it on the robot.

The software running on the robot consists of two main layers. At a lower level, the operating system and device drivers communicate with the robot's hardware and provide networking functionality. Algorithms to process the sensory input and control the motors run at a higher level, and can run either on the robot or on the development computer, communicating over the wireless network with the robot's computer.

4.2.1 Operating system

A "Linux from Scratch" (LFS) [10] install was chosen for the computer's operating system. While many different distributions of Linux exist, these are mostly targeted at the desktop

computer environment, sporting graphical user interfaces and a large number of preinstalled software packages. A LFS install allows for the total customisation of a particular computer, and using this method allowed for the installation of only necessary software. This improves the startup time and resource utilisation, necessary in this embedded environment. It is also an excellent learning experience for understanding the workings of a Linux system, which facilitates future development and any continuing maintenance.

One of the graphical desktop distributions, Fedora Core, was initially installed on the computer and used to bootstrap the installation of the Linux from Scratch install. Once this initial install was successful, the instructions in the LFS book were followed to install the required supporting software and a Linux kernel.

Linux OS installation

The process of installing LFS is described in [10]. It consists of several distinct phases, which deal with the preparation for a new system, building the required software and installing the OS components.

The preparation stage involves building a set of temporary development tools on a separate hard drive partition which will hold the completed system. The idea is to obtain a toolchain independent from the host installation. The major packages installed are GCC (GNU Compiler Collection), Binutils (linker, assembler, etc) and various patch, testing, compression and scripting tools, along with their supporting libraries. These are statically linked[1] to remove any dependencies on the host's native toolchain. This is similar to the process of setting up a cross-compiler for another computer architecture, but instead targets the same architecture as the host.

Once this initial toolchain has been installed, it is used to compile many of the same tools again, this time with dynamic linking. A Linux directory structure is set up on the partition, and a `chroot` 'change root' command is executed, which causes subsequent commands to act upon the new partition rather than the host installation. At this point, all the software needed for a basic system is installed. This includes all of the packages from the previous section, along with Glibc (the core C library on a Linux system), filesystem utilities, login and terminal interaction programs, along with all the necessary support packages.

The next step is to install system 'bootscripts' which determine what programs are executed at startup. The scripts will be modified later to start up our custom robot software.

[1] Static linking includes a copy of any library code in the actual executable. Dynamic linking uses a 'shared library', of which only one copy is held on the system (or in memory) and used by every application that links to or uses this library.

The final step is to compile a Linux kernel. The kernel is the core of the Linux operating system, controlling aspects such as memory management, process scheduling, hardware interaction and networking. This requires the selection of many configuration options. This compilation was repeated several times over the course of the project, due to the omission of particular options required later, such as those for wireless networking. The final kernel was built as a modular kernel[2], allowing the drivers for the hardware to be loaded later. This also allows the unloading and reloading of drivers modules during testing.

Once all these steps are carried out, the boot loader configuration is updated to include the new partition, and the system is rebooted. At this point, a basic command line is available and not much more. The additional software needed is covered in the following sections.

Support software

In addition to the base LFS install, a number of packages described in the "Beyond Linux From Scratch" (BLFS) book were also installed for support purposes. These included an SSH server, which allowed access to the operating system's command line over the network and also the removal of the keyboard and monitor from the computer. With the wireless network enabled, this eliminated the need for any cabling to the robot.

4.2.2 Hardware drivers

Some of the hardware added to the computer was not supported natively by the Linux OS base install. This then requires the installation of some further device drivers and interfacing software. These included the wireless Ethernet card, the onboard I^2C bus and the video camera.

Wireless Ethernet card

The 802.11b/a/g Wireless Ethernet card is a Netgear WAG311 PCI card. This particular card was chosen because it was known to be supported by the MAD-WIFI driver.[3] The installation of this driver required the enabling of wireless networking in the kernel, the installation of the Wireless Tools to view and control the wireless connection status and finally the installation of the driver itself.

[2]The other option is a monolithic kernel, where the kernel and all the device drivers are compiled into a single large file.

[3]Available: `http://sourceforge.net/projects/madwifi/`

Once this installation was carried out, a wireless router (Linksys WRT54G) was setup to allow the robot's computer to attempt to connect to a wireless network. Once connected, the results of listing the configuration information are shown in Section C.1. At this point, the wired Ethernet connection was removed, and the computer could be accessed wirelessly.

I^2C bus interface

The Linux kernel comes with basic support for the I^2C bus, however this support is insufficient for development purposes. Updated versions of the `i2c` and `lmsensors` packages[4] were installed. The former allows for accessing the I^2C bus through a file-like interface. The latter allows reading the computer's onboard temperature and voltage monitors, along with utilities for scanning the bus for I^2C devices.

These packages were used in the development of the software to access the electronic compass, and will also be used to verify the working of the PIC I^2C interfacing firmware. The primary reference for programming the I^2C interface was [11].

Video camera

The Logitech camera requires a non-standard driver, the `pwc` and `pwcx` driver modules.[5] The 'pwc' stands for '**P**hilips **W**eb **C**amera', which is the chipset this camera is based on. The first driver is open-source, and allows for basic access to the camera. The second module is the **pwc**-e**x**tended driver, and allows the use of the high resolution and high framerate modes of the camera. It contains proprietary Philips information on the compression routines, and is only available as a prebuilt binary module.

The driver requires the basic USB modules to be present. Once installed, the `camsource` program was used to view the video data, as described in Section 4.2.3.

4.2.3 Sensor and Motor Interfacing

The previous section has dealt with the installation of the operating system and drivers for the various hardware devices. Once this is completed, it is necessary to have userspace programs (as opposed to kernel space) to interact with these drivers and retrieve the sensory data or send commands to the actuators. This section will cover the software obtained and written for these tasks.

[4] Available: `http://secure.netroedge.com/~lm78/`

[5] Available: `http://www.smcc.demon.nl/webcam/`

Video streaming

Once the drivers for the web camera have been installed, a software application to retrieve the video data is necessary. Some of the options researched include the programs `ffmpeg` and `camsource`.

The former is the general purpose library of audio-visual codecs for Linux, along with supporting software. As a part of the package, a streaming server `ffserver` is included. The installation of this package was attempted, however it is known to be currently non-functional, and therefore out of consideration. However, in the future its usage should be reattempted, as it offers MPEG-2 & 4 compression, which would be useful for streaming the high resolution, high framerate video to a remote computer.

`camsource` is a specialised program for streaming web-camera video to a web browser. It encodes the individual video frames using JPEG compression and sets up a small http server to serve these across the network. The video can then be viewed using any web browser, albeit at a lower frame rate of 3–5 fps and with a long latency of half a second. This will suffice for monitoring, however other options for higher quality video streaming will be investigated later. This latency should not affect the basic teleoperation significantly, since the speed of the robot will be restricted.

Range scanner

The Hokuyo PB9-11 range scanner connects to the computer via a serial RS-232 connection. An application program is provided to access the device from Microsoft Windows, however the protocol is not documented. We are grateful to Stefano Carpin[6] for providing us with his reverse-engineered specification for the protocol, along with sample C code to setup and retrieve data from the device.

Currently this code opens a 38.4 kbps[7] serial connection to the range scanner, and reads back 91 range values, covering the 162° field of view. The device disconnects after a set number of readouts, and it needs to be reconnected in order to receive a constant stream of data. This software will be rewritten in Python, so as to send the data directly over the network to the monitoring computer.

[6] School of Engineering and Science, International University Bremen, Germany

[7] kilobits per second

Electronic compass

The CMPS03 compass is connected directly to the I^2C bus on the motherboard. To communicate with the device, a simple C program was written, based on sample code for another device from the same manufacturer.

To read the heading, two registers on the device are read over the bus. This is done using the Linux `ioctl`[8] API[12, §6.7]. These two registers, representing the high and low bytes of the total value, are combined in software. The result is a number ranging from 0 to 3599, which is the current bearing in degrees multiplied by a factor of 10.

See Section C.2 in the appendix for the source code to perform these operations.

Sensory data & Motor control

The microcontrollers to input data from the remaining sensors and control the motors also connect to the I^2C bus. Therefore they are accessed in a similar manner as the compass from the previous section. As the microcontroller software is not yet operational, (see Section 4.2.4 for details) we are limited to an overview of the operation of this software.

The sensory data will be read from the microcontrollers in an identical fashion to the compass. Read commands over the bus for various registers will retrieve this data. Once implemented, this is how the data from the temperature sensor, pulse encoders, CO_2 detector and power management circuits will be read.

The motor control will be carried out by sending two numerical values to the microcontroller. These will specify the speed of the left and right motors, respectively. If these are byte-wide values, then these will be in a range of ± 128, which is sufficient granularity to specify enough differing speeds from full forward to full reverse.

There will be no checks on this output data in the user program. Rather, the microcontroller software will perform these checks, such as ensuring that speed limits are not being exceeded. This is to ensure that AI software under development will not be able to cause unsafe behaviour from the robot.

4.2.4 Microcontroller software

The software which will be in the microcontrollers is also called firmware. It is expected once it has been fully developed and the bugs removed through testing, to remain static and rarely changed. Development will done in PICmicro assembly code in the Microchip MPLAB

[8] I/O control

development environment. The device data sheet [13] provides most of the information necessary for the development, and a useful development reference is [14].

I^2C bus interfacing

Since the primary job of the microcontroller is to interface signals to the main computer, the bus interface is the most important component in this chain. The devices were chosen primarily for their onboard I^2C interface, for which sample code [15] is provided by Microchip. An application program 'Application Maestro' generates the necessary subroutines to access the bus, and the remaining task is to provide the data to be sent by these routines.

Motor drive control

As has been specified earlier, the motors will be driven through an H-bridge circuit with PWM waveforms. The task of the motor drive microcontroller is to generate these pulse waveforms [16, 17]. The computer will send a byte for each of the motors, specifying a value from -128 to $+127$, for full reverse to full forward.

This value will not be used to directly calculate the 0–100% duty cycle PWM waveforms for the drive. Rather, first software speed limiting will be applied. As will be noted in the section on the physical drive chain, the top speed is possibly upwards of $4ms^{-1}$. This is an unsafe speed, so the duty cycle of the PWM will be restricted, possibly to the 25–75% range.

The resulting value will be used as a target value, and the duty cycle will slowly be ramped up to this value over approximately a second. This is to prevent sudden current surges, which could disrupt the electronic circuitry with inductively induced noise or possibly destroy the drive FETs. Once the current sensing and monitoring has been implemented, we can allow a quicker ramp, and use the monitored current value to prevent surges.

Analog sensor reading

At this stage, there are several analog sensors. These are the CO_2 detector, the temperature sensor and the current shunts for power monitoring. However, the analog signal conditioning circuits for these devices have not yet been built, and therefore these sensors will not be integrated by the end of this phase of the project.

However, it is possible to outline how this will occur. The sensors will be connected to differential amplifiers, followed by any necessary filtering, which will then connect to the analog to digital conversion inputs on the microcontroller(s). These signals will be sampled

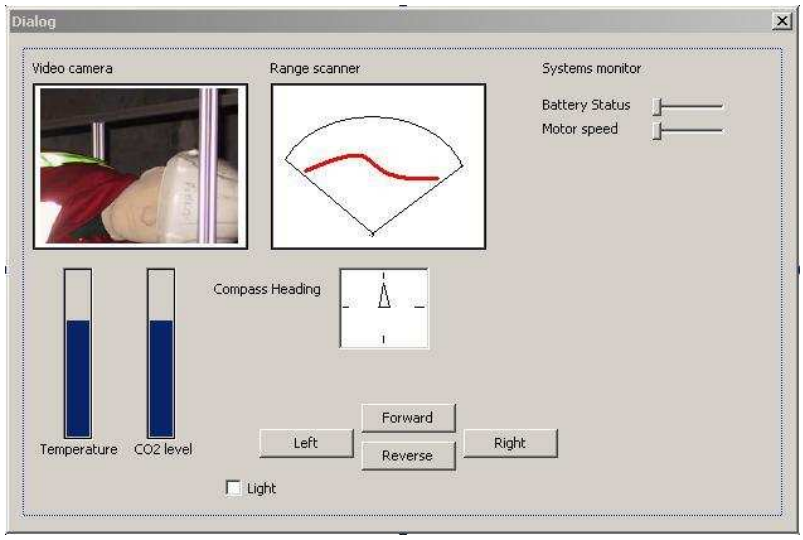


Figure 4.1: Robot control graphical user interface

at up to a 1kHz rate. These values will be stored and transmitted to the computer when the microcontroller is next polled over the I^2C bus.

4.2.5 Remote operation software

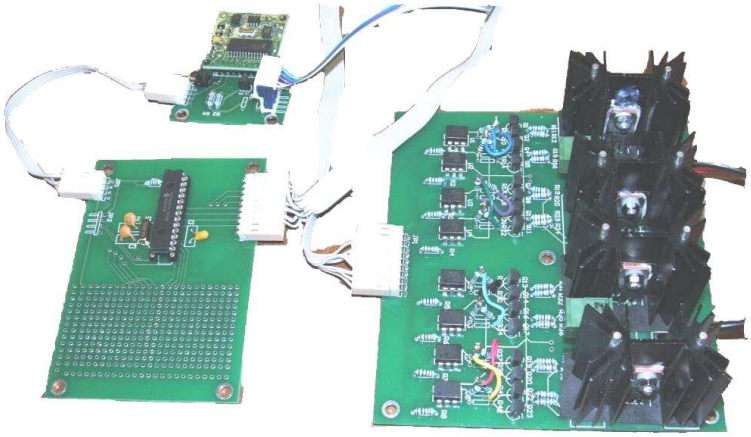
The previous sections have dealt with the process of retrieving the data from the individual sensors and sending this data over the wireless network. A remote computer is therefore required to receive this data, and display it to a human operator. Motor control commands will be input by this operator, and sent back to the robot. A prototype of the interface is shown in Figure 4.1.

This program will be written using Python and will be able to run either on Windows or Linux PCs. The graphical user interface will use the *wxPython* toolkit[9], and interaction with a keyboard or joystick will use the *Pygame* toolkit[10]

4.3 Circuit Board Designs

Three custom circuit boards were designed for this project. These provide the functionality of connecting to the analog sensors and controlling the motors. The rationale behind separate circuit boards rather than a single monolithic system interface board was modularity. It was anticipated that problems would be encountered in the separate components, and having separate modules allowed for easier testing and correction of any errors, in the

[9] See `http://www.wxpython.org`
[10] See `http://www.pygame.org`



(Clockwise from top left: I^2C interface with compass, H-bridge drive and microcontroller development boards.)

Figure 4.2: Completed circuit boards

restricted time frame for achieving full functionality. This did prove to be the case and this modularity allowed for the easier repair of these problems.

It is anticipated that in the future, once the separate circuits are working, these boards will be redesigned using SMD components on a single circuit board. This will reduce the space taken up by the interface boards by a factor of 3 or 4.

Figure 4.2 shows the completed circuit boards, which are described in the following sections.

4.3.1 Microcontroller Support

The primary job in interfacing the analog sensors and actuators was to provide a method for the signal conditioning, analog to digital conversion and sending the resultant data back to the main computer. This task was carried out by a Microchip PIC16F876 microcontroller, which is an 8-bit device with 8 Kwords of program memory, 368 bytes of RAM and 256 bytes of storage EEPROM in a 28 pin DIP package. The most significant features of this device are an onboard I^2C interface and 5 A/D converters. While a PWM controller is also present, this is restricted to two channels, which is insufficient for driving two motors in the push-pull manner that the H-bridge drive requires.

This circuit board provides the basic setup for the microcontroller. Provided are dual I^2C ports, an 8 lane I/O connector and a small prototyping area; which can be used for implementation of the analog circuitry, along with basic circuit testing and debugging. The remaining components are the 20 MHz crystal oscillator and power supply decoupling. The full circuit is shown in Appendix A.

4.3.2 I^{2}C Voltage Level Translator

The EPIA motherboard provides an I^{2}C interface connector onboard. This is a 3.3V interface, and one component required a 5V I^{2}C bus; the electronic compass. It was decided to make all the peripherals connected to the I^{2}C bus run on 5V to simplify interfacing the microcontrollers, and use a logic level translator between the motherboard and the remaining devices. A sample circuit using two FETs to provide this level translation is supplied in [18], which was used as a basis for this circuit. Also included are connectors for the peripherals, a 5V power connector and a socket for connecting the electronic compass. Appendix A contains the full schematic for this circuit.

It was later discovered that this voltage translation is not necessary. Since the drivers on all outputs are open collector, these only need to be connected together. The resulting logic high of 3.3V is input as high on the PIC microcontrollers as well. (This was the original concern that lead to the design of this board.) Therefore, the transistors were removed from the board and the connections bridged with links. The board is still used for connecting the compass and for providing the I^{2}C signals and power to the PIC development boards described above.

4.3.3 H-Bridge Motor Drive

The motors are driven by a H-bridge circuit which, when driven by PWM waveforms from a microcontroller, allows for full control over the speed of the individual motors. Since the motors to be driven are 200W at stall, with a 24V source voltage, this circuit is designed to switch up to 10A per motor. The circuit is active-off, such that the motors are disabled whenever there is no signal from the microcontroller.

Figure 4.3 shows a quarter of the total circuit, which drives one side of one motor. On the connector, pins 2-10 are the digital outputs from the microcontroller. Pairs of these outputs are used to drive the input LEDs of two optoisolators; U1 and U2. The optoisolators will be off whenever the inputs are the same, and each individual opto will be on when its input is high and the second input is low.[11] This assures that the pair of optoisolators are never on simultaneously, which prevents short-circuiting the power supply through the FETs. Table 4.1 shows a logic table for the drive circuit.

The drive MOSFETS, Q3 and Q4, are enhancement mode P-channel and N-channel devices, rated at 23A. The drive circuit for the FETs consists of two totem-pole PNP and NPN BJT pairs (Q1-Q2 & Q4-Q5). These are biased such that they drive the FET gates

[11] Thanks to Rick Middleton for suggesting this drive circuit.

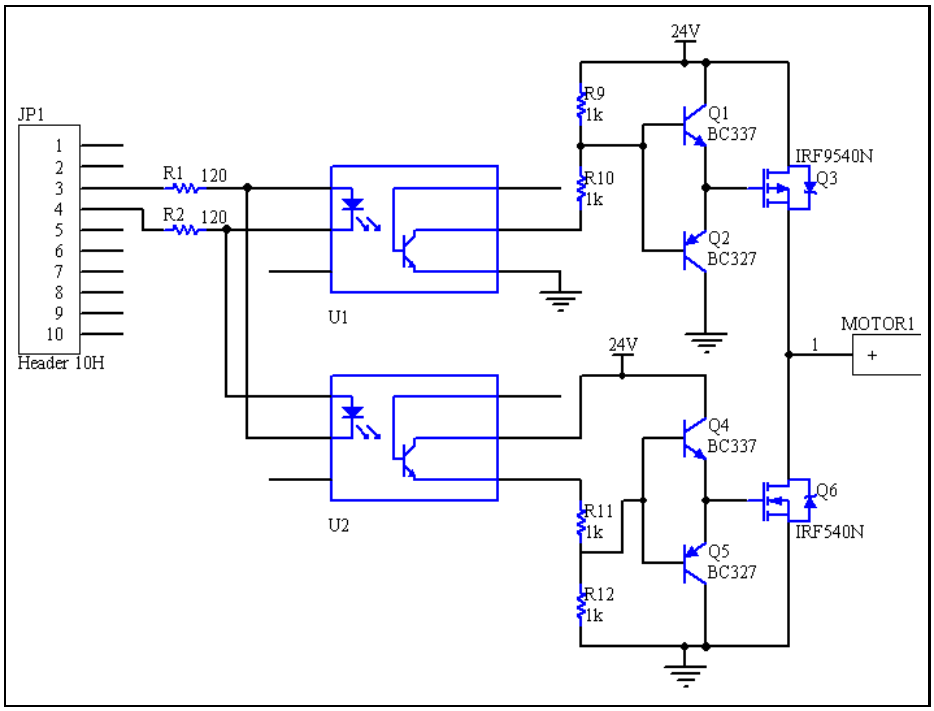


Figure 4.3: H-bridge circuit schematic section

Input 1	Input 2	Optoisolator 1	Optoisolator 2	Q1	Q2	FET Q3 (P-ch)	Q4	Q5	FET Q6 (N-ch)
L	**L**	off	off	on	off	**off**	off	on	**off**
L	**H**	off	on	on	off	**off**	on	off	**ON**
H	**L**	on	off	off	on	**ON**	off	on	**off**
H	**H**	off	off	on	off	**off**	off	on	**off**

Table 4.1: H-bridge logic levels

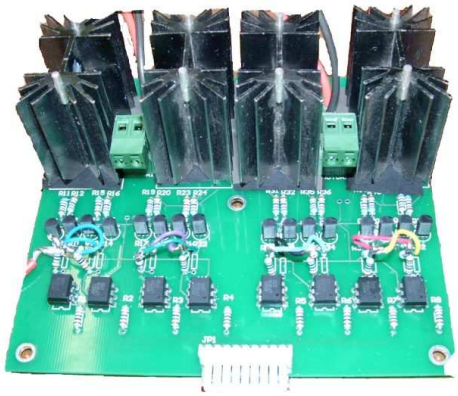


Figure 4.4: H-bridge circuit board

to 24V (for the P-channel FET) and 0V (N-channel) when the optoisolators are off, and to 12V when they are on[12].

The remaining three-quarters of the circuit, as shown in Appendix A, is identical to this section, with one to drive the other side of this motor, and the other two to drive the second motor. The full circuit allows for three states for each motor, which are driven forward, driven backward or electrical braking. Switching between these states using PWM waveforms allows for full speed control of the motor.

Power calculations for the dissipation in the FETs indicated that this is relatively low. This is due to the use of this particular drive circuit, since the totem-pole BJT pairs ensure quick switching through the rapid charging and discharging of the gate capacitance. The maximum dissipation per FET was calculated at approximately 4 watts at a 20 kHz switching rate. The heatsinks have been overspecified, but this should help to prevent any faults during initial testing from immediately destroying the power transistors.

The completed circuit board is shown in Figure 4.4.

4.4 Power considerations

Given that the robot has to operate without any restricting cables, it was necessary to provide a power source on the robot. This was chosen to be rechargeable batteries, since this is the only viable technology option at present. It would also be useful to provide power usage monitoring for fault detection and future development.

[12] $V_{GS(max)} = \pm 20V$

4.4.1 Battery selection

The choices considered for the power source were the various forms of battery technology. These included sealed lead acid, nickel cadmium, nickel-metal hydride and lithium-polymer cells. A comparison between these different chemistries is given in Table 4.2. The total weight for a 24V, 7.2Ah battery and the total cost is given as well, as these are the parameters used to decide which chemistry to use.

While SLA technology has the lowest energy density, it also has the lowest cost per watt-hour. Since the robot already has a weight of 20 kg in the tracks alone, an extra few kilograms is not a significant concern. Therefore the choice was made to use the SLA gel cells on the basis of cost. These are shown in Figure 4.5, and two will be used to obtain the 24V desired.

Battery life

The capacity of the battery determines the amount of running time available for the robot. Table 4.3 shows estimated power consumption and running time. This is calculated for 2 batteries with a 7.2Ah rating. These calculations take into consideration the derating curves for the batteries, which explains the low running time available at maximum power consumption. This short duration is the motivation behind providing a power consumption measurement ability, discussed later in this section. The maximum power draw is dependently mostly on the motors, and it will be important to be able to detect a stall and/or an over-current condition.

At an average power consumption level, these batteries give a running time of approximately an hour, which should be sufficient for the competition. It is expected that the robot should be able to remain powered on while the batteries are recharged, limiting the amount of down time.

4.4.2 Power supplies

Once the battery voltage has been decided, it remains to provide power supplies for the electronic components. The 24V supply voltage was chosen primarily for the motors and also for the range scanner. The computer system requires voltages of 12, 5 and 3.3 volts. These were catered for by purchasing a small switching power supply which snapped directly into the power socket on the motherboard, and accepts an input from 11–30 volts. This allows us also to charge the batteries while the robot is running, since this requires 28V which is still within the input range. The total power output from this supply is a maximum of

Figure 4.5: Sealed lead-acid gel cell battery

Chemistry Type	Energy Density Wh/kg	Cost $/Wh	Total weight kg	Total cost AU$
Sealed Lead Acid (SLA)	50	0.50	3.5	80
Nickel Cadmium (NiCd)	65	2.30	3.2	400
Nickel Metal Hydride (NiMH)	70	2.30	3.1	400
Lithium Polymer (LiPoly)	140	4.30	0.7	740

Table 4.2: Battery technologies compared. Prices are approximate.

Component	Power (watts)		
	min	avg†	max
Motors	0	50	400
Computer System‡	20	30	60
Range Scanner	0	4	6
Support Electronics	5	8	10
Total	25	92	476
Estimated time available (hours)	5	1	<0.1

†Estimated usage.

‡This includes the processor, motherboard hard drive, and wireless Ethernet card.

Table 4.3: Power consumption by component.

Component	Weight (kg)
Tracks ×2	20
Motors ×2	4
Batteries ×2	7
Chassis	3
Electronic components	3
Total	37 kg

Table 4.4: Robot weight breakdown

80 watts, which is ample to power the computer and the custom interfacing logic.

4.4.3 Power monitoring

If we are able to measure the power consumed by the robot, this provides us with a few useful features. Primarily this can be used to detect a fault condition. If the robot is unable to move, the motors will stall and draw their full rated current of 11 amps each. It would be advisable to avoid this situation to prevent running down the batteries (which will occur quickly at a 22A load) and overheating the motors. This information could also be used to modify the design for lower power consumption in the future.

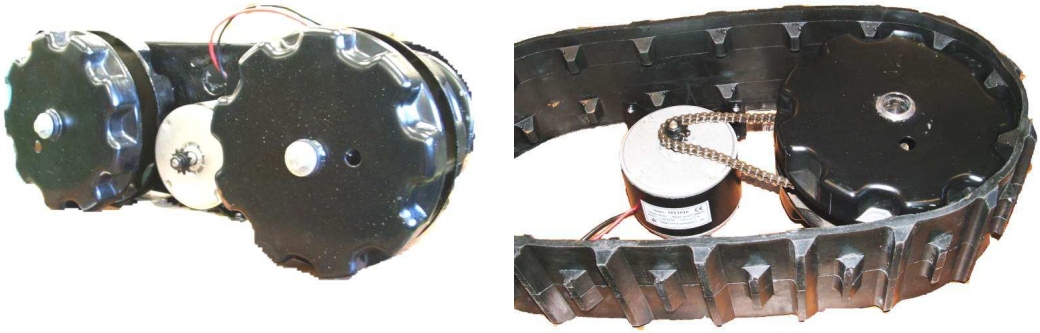
This power consumption reading will be generated by placing current shunts inline with the power leads. One shunt will be placed on each motor, and one on the electronics. A microcontroller with a differential amplifier will be able to read the resulting voltages. However, the implementation of this system will not be carried out for the project, and will be left as an future extension.

4.5 Mechanical construction[13]

This section covers aspects of the mechanical design and construction for the robot. This includes details on mounting components on the chassis and gearing for the motors and drive wheels.

The largest components to be mounted are the motors and batteries. The motors are mounted inside the tracks on the outer side, after a section has been cut out of the steel. This leaves the area between the tracks for the other components. See Figure 4.6 for a photo of the mounting process.

[13]The mechanical design details and construction is being carried out by Russell Hicks, of the Electrical Engineering technical staff.



(a) Drive hubs & motor

(b) Track, motor & chain

Figure 4.6: Tracks and motor mounting.

The dimensions of the robot remain as was specified in the design chapter. The length of the robot is 500mm, which is the length of the tracks. The total height is 350mm, with only the camera and rangefinder over the top of the tracks. The width is 450mm.

4.5.1 Motor drive system and gearing

The tracks are driven by the motor through a chain and sprocket. The motor sprocket has 11 teeth, and the drive sprocket 80, so there is a step-down gearing of 1 : 7.3. This reduces the 2750 RPM maximum at the motor shaft to 377 RPM or 6.3 revolutions per second at the track wheel. Since the wheel has a diameter of 220mm (including the track) and a circumference of 691 mm, this then gives a top speed of approximately $4.3ms^{-1}$ or $15^{km}/_{h}$. A preferable speed, for safety reasons, would be around 2–$3ms^{-1}$ or walking pace, hence the need for the software speed limiting described in Section 4.2.4.

The drive sprocket was the largest that can fit inside the hubs without coming into contact with the rubber track. This, along with the immediate availability of these drive sprockets and chain, was the reason behind this choice in drive gearing. If this setup does not offer sufficient torque, confirmed through field testing, a later improvement would be to use idler sprockets and two chains to gear down the motor further. This was not attempted initially due to the limited space inside the tracks, making this option more difficult in construction.

4.5.2 Component mounting details

The batteries will be placed between the tracks at the rear of the robot, with their terminals exposed for the ease of connecting a charger. Forward from the batteries, the fuse block and current shunts will be mounted on a vertical plate, such that the minimal length of wiring

is needed for the high current connections. The motor drive board will be mounted on a second vertical plate forward of the batteries.

At the front of the robot, the motherboard will be mounted in an enclosure towards the ground with a plate underneath. On top of this will go a second plate, with the hard drive mounted underneath with damping foam, to isolate it from any significant vibrations. The range scanner and video camera will mounted above this top plate, which will have a view unobstructed by the tracks or other components. This will also allow for the later mounting of a panning platform for the camera.

Chapter 5

Results and Outcomes

The following pages discuss the current state of the project, and the goals achieved towards the culmination of the project. Also included are details of the expected status of the project by Open Day, along with information on what will be presented.

5.1 Project status (as of 20 October, 2004)

Most of what has been discussed in the implementation chapter has been completed at this stage. The following sections distinguish between the fully completed tasks, those that remain to be completed and tasks that will be completed after the assessment date.

5.1.1 Completed tasks

The system design and selection of the components for the robot was a large undertaking, but this has been completed.

The computer system is functional, with the operating system working as required. This required a significant amount of time invested in both preliminary research, installation and troubleshooting. All of the necessary device drivers have been installed and function correctly. The wireless Ethernet card connects to the wireless LAN and the computer can be controlled over this link using SSH. The `camsource` program is running and streaming video data back to the remote computer, which is a significant portion of constructing the remote control program.

The circuit boards have been completed and basic tests have been carried out. The I^2C interface board works as described, with the connection to the compass functional. Correct bearing data has been successfully read back from the device.

The original design for the motor drive board was erroneous, but a substantial mod-

ification of the FET drive circuit was possible on the circuit board. The h-bridge board switches correctly, but has not yet been connected to the motors. The protection against short-circuiting the power supply functions as expected.

The design originally called for driving the tracks with cordless drills, which would be purchased with rechargeable batteries for power. This plan was changed for the 200W motors due to the improved power and torque available and the easier integration with the remaining components. The price difference was insignificant.

The choice in battery technology and voltage was left until later in the project, due to uncertainties about the motors, and in the voltages and currents that would be required. The final selection of SLA gel cells represents what seems to be the best balance in ratings, weight, price and availability.

5.1.2 Tasks to be completed before assessment

There are some tasks remaining that are expected to be completed by Open Day. Some of these are the completion of the mechanical construction, the implementation of motor drive code on the microcontrollers and the remote operator interface. These are the most important tasks, as the proposed demonstration requires these to be completed.

The completion of the mechanical construction is contingent on time being available to Russell Hicks for this job. The estimate is for this to be complete by November 15, in time for the demonstration. However, the components can be tested separately before they are integrated. It is expected that speed and control tests will be carried out in the week prior to the demonstration.

The software for the PIC microcontrollers has been researched but only partly written. The two critical portions of code to be written are the I^2C bus interface and the PWM waveform generation, to allow for the control of the motors.

Finally, a prototype of the remote graphical control system has been drawn up, but is not yet functional. This will be implemented by Open Day, and will allow the viewing of the video stream, range data and other monitoring data. If the motor control is functional, it will allow for remote control of the robot's movement.

Other tasks that will be completed within the month include interfacing to the range scanner, the completion of the data broadcast communication software. Testing of the integrated components also remains to be carried out.

5.1.3 Incomplete tasks

The largest uncompleted task was the interfacing circuits for the analog sensors. This was of lesser importance, since this data is not necessary for the teleoperation of the robot, the major goal of the project. The code used for the motor drive microcontroller will require only the addition of A/D control routines to be used to interface to the CO_2 gas sensor, thermometer and lighting for the camera.

A microphone and speaker were to be connected to the computer's onboard sound system, however this requires the difficult installation of a driver for the OS and complex application software for recording, playback and streaming over the network. As with the previous uncompleted sensors, this is not required for the robot teleoperation and was omitted due to lack of time.

The pulse encoders on the drive wheels were also not implemented. This task relied heavily on the actual mechanical construction of the drivechain. Since this has not yet been completed, the pulse encoders have not been attempted.

5.2 Demonstration proposal

The proposed presentation for this project on Open Day will involve a demonstration of the current capabilities of the robot. It is anticipated that the robot will be able to be teleoperated from a remote computer. Any questions about the design, construction process and the future goals for the project will be answered.

Ideally, it was anticipated that a suitable demonstration of the robot would be to ascend stairs under remote control. Whether this is possible will be unknown until the evaluation of the fully integrated robot. Should the mechanical construction not be complete, only a demonstration of the wireless monitoring ability will be possible.

Chapter 6

Conclusion

This report has covered the design and construction of a robot for the RoboCup Rescue competition. The preceding chapters have covered each of the steps in this process; design, implementation and results. Here this information is briefly summarised, and the future goals for the project and possible extensions are presented.

6.1 Project summary

The current state of the project is showing the potential of the design, with the remote operator being able to view the video stream and monitoring data from the robot. The evaluation of the robot's capabilities awaits the completion of the mechanical construction and motor drive system.

The lessons learned from this project include an appreciation of the many areas of engineering that enter into the construction of a robot. The need to consider every subsystem of the final product when selecting components and their integration was also demonstrated. A deeper understanding of the tasks involved in system design, operating system installation, circuit board design, and mechanical design issues was gained.

6.2 Extensions

The next stage in this project will be the preparation for RoboCup Rescue 2005 in Osaka, Japan. The completion of any outstanding tasks will be the first goal. After these are completed, work on artificial intelligence routines to allow some degree of semi-autonomy will begin.

Following the assessment of this project, a few tasks will remain from the original design. The remaining sensors will be interfaced and integrated into the system. Pulse encoders

will be added to the motors to allow speed measurement. If the tracks lose traction only occasionally, these can also be used for obstacle collision detection.

If required, some additional odometry sensors could be added for localisation. These include an accelerometer to determine the tilt of the whole vehicle, useful when travelling up and down stairs, and a GPS unit for exact positioning, although this may not work in a collapsed structure without line of sight to the sky.

The video camera and range scanner would be more useful if they could be mounted on a pan-tilt platform. Data gathering in a 360° radius could then be carried out without movement of the entire robot. The latency issue in using the `camsource` package will need to be addressed for accurate teleoperation. The thermometer may be scrapped in favour of a passive infrared (PIR) sensor, similar to those used in home security systems, for detecting body heat.

Some other ideas that arose during the project include adding a power management microcontroller, which can be used to monitor and collect data on power usage patterns. This would be connected to the I^2C bus, and convert the signals on the current shunts. The remote user interface could then be used to monitor instantaneous power usage.

The remote interface software can also be enhanced to show more monitoring signals from the computer or historical data. Once AI routines are developed, this software will also show the state of the robot's world model and decision making process.

The artificial intelligence routines for the robot will be developed along a similar architecture to that of the University's RoboCup team. The required tasks will be divided into modules such as vision, localisation and world modelling, behaviour and locomotion control. Development on these will be able to occur simultaneously. If the AI code is developed using a scripting language such as Python, software components will be able to be replaced on the fly.

Each of these tasks will initially be carried out by a human operator. The first modules to be developed will be vision, localisation and world modelling. The scoring for the competition is contingent on preparing a good quality map of the arena, so this will be a primary goal. Work on visual and sensory detection of the 'victims' will also be required. The more advanced modules such as behaviour and advanced locomotion will not be implemented initially, with these higher level tasks being carried out by the operator.

References

[1] H. Kitano and S. Tadokoro, "RoboCup Rescue: A Grand Challenge for Multiagent and Intelligent Systems," *AI Magazine*, vol. 22, no. 1, pp. 39–52, Spring 2001.

[2] H. Kitano, S. Tadokoro, I. Noda, H. Matsubara, T. Takahashi, A. Shinjou, and S. Shimada, "RoboCup Rescue: Search and Rescue in Large-Scale Disasters as a Domain for Autonomous Agents Research," in *Proc. of the IEEE Conference on Systems, Men, and Cybernetics*, 1999.

[3] S. Murphy, *Introduction to AI Robotics*. MIT Press, 2000.

[4] (2004) Robocup Rescue Competition Rules. Robocup. [Online]. Available: http://robotarenas.nist.gov/RoboCup-AAAIRescueRobotCompetitionRules2004(v1).pdf

[5] J. Casper and R. Murphy, "Human-Robot Interactions during the Robot-Assisted Urban Search and Rescue Response at the World Trade Center," in *Proc. of the IEEE Conference on Systems, Men, and Cybernetics*, vol. 33, no. 3, 2003, pp. 367–385. [Online]. Available: http://crasar.csee.usf.edu/research/Publications/CRASAR-TR2003-2.pdf

[6] *Robocup 2003: Proceedings of the International Symposium*. Robocup, 2003.

[7] A. Jacoff, B. A. Weiss, and E. Messina, "Test arenas and performance metrics for urban search and rescue robots," Intelligent Systems Division, National Institute of Standards and Technology, United States Department of Commerce, Tech. Rep., 2003.

[8] S. Niku, *Introduction to Robotics*. Prentice Hall, 2001.

[9] (2004) Open Automaton Project. [Online]. Available: http://oap.sourceforge.net

[10] G. Beekmans, Ed., *Linux From Scratch*. Cheap Bytes, 2004. [Online]. Available: http://www.linuxfromscratch.org

[11] D. Walters. (2004) Mr. Davbot's Musings: The I^2C Interface. [Online]. Available: http://www.mobilerobotics.org/robot/index.php?option=content&task=view&id=123&Itemid=44

[12] M. Mitchell, J. Oldham, and A. Samuel, *Advanced Linux Programming.* New Riders Publishing, 2001. [Online]. Available: http://www.advancedlinuxprogramming.com/

[13] *Microchip PIC16F87X Data Sheet*, 2001. [Online]. Available: http://ww1.microchip.com/downloads/en/DeviceDoc/39582b.pdf

[14] M. Predko, *Programming and Customizing PICmicro Microcontrollers.* McGraw-Hill, 2001.

[15] S. Bowling, *Using the PICmicro SSP for Slave I^2C Communication*, 2002. [Online]. Available: http://ww1.microchip.com/downloads/en/AppNotes/00734a.pdf

[16] M. Palmer, *Using the PWM*, 1997. [Online]. Available: http://ww1.microchip.com/downloads/en/AppNotes/00564b.pdf

[17] O. Röpcke, *PWM, a Software Solution for the PIC16CXXX*, 1997. [Online]. Available: http://ww1.microchip.com/downloads/en/AppNotes/00654a.pdf

[18] H. Schutte, *Bi-directional level shifter for I^2C-bus and other systems*, 1997. [Online]. Available: http://www.semiconductors.philips.com/acrobat/applicationnotes/AN97055.pdf

Appendix A

Circuit Diagrams

This appendix contains the circuit schematics for the PIC development, I^2C level translator and H-bridge motor drive boards. These were drawn up and converted to circuit board layouts using Altium Protel DXP 2003.

Figure A.1: PIC Microcontroller circuit

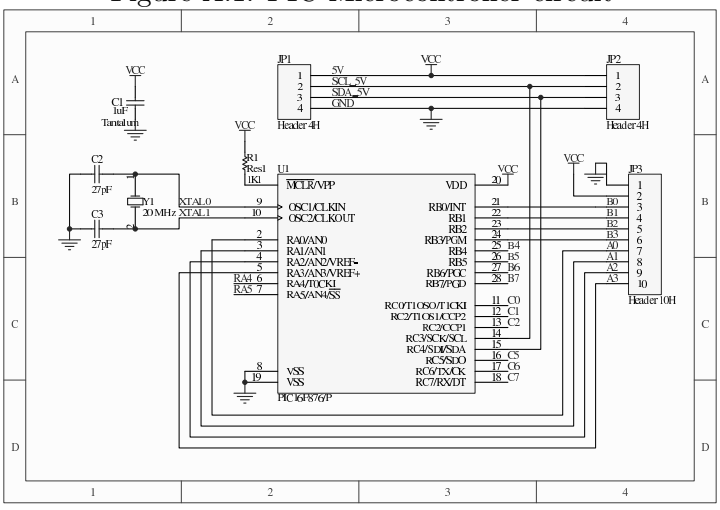


Figure A.2: I^2C Level Translator circuit

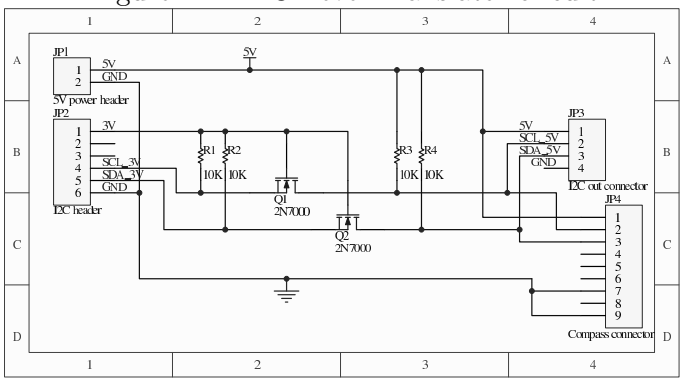
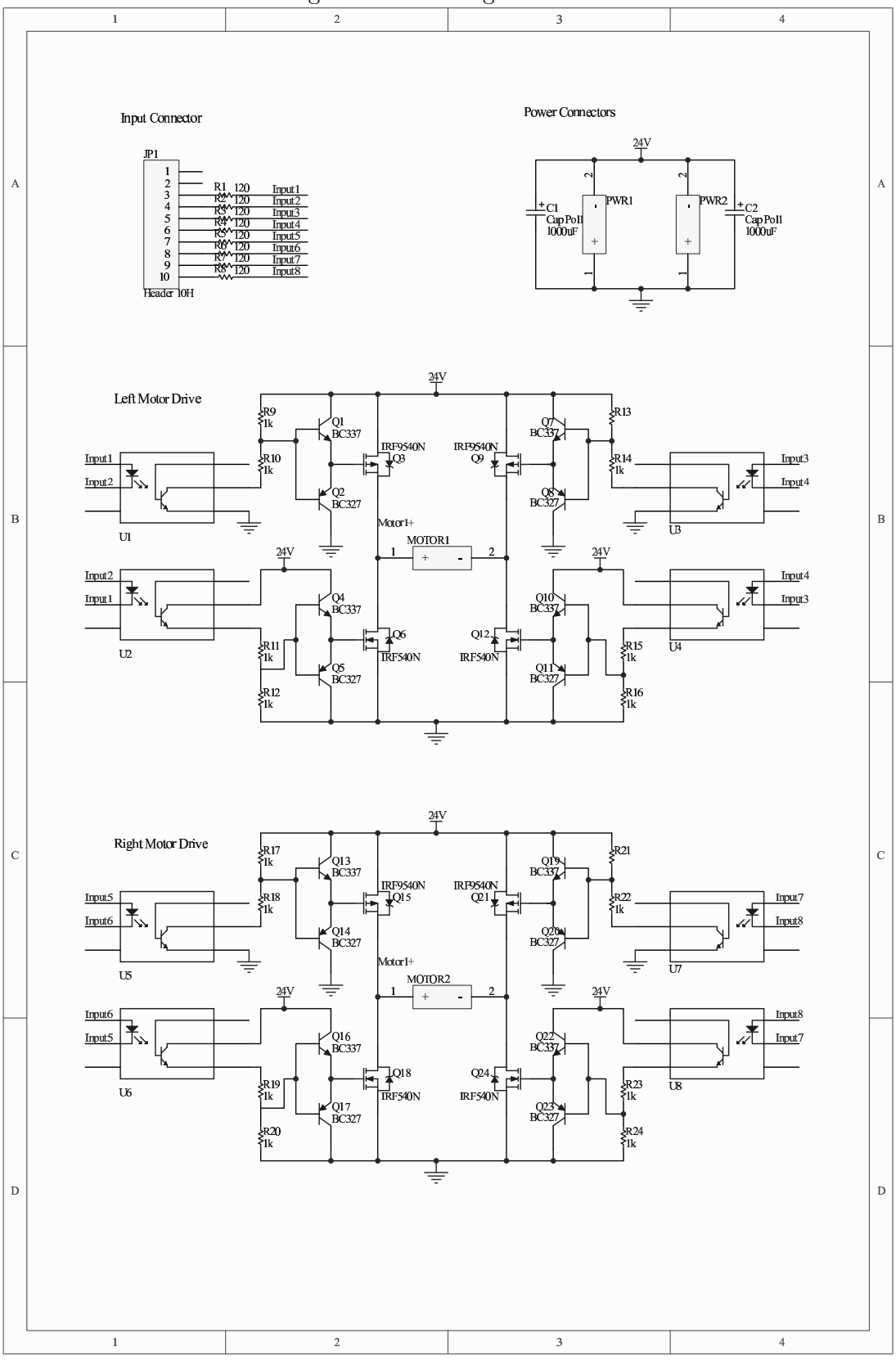


Figure A.3: H-bridge circuit



Appendix B

Circuit Board Layouts

This appendix contains copies of the circuit board layout, with tracks and component placement. These reflect the schematics in the preceding appendix.

Figure B.1: PIC Microcontroller board layout

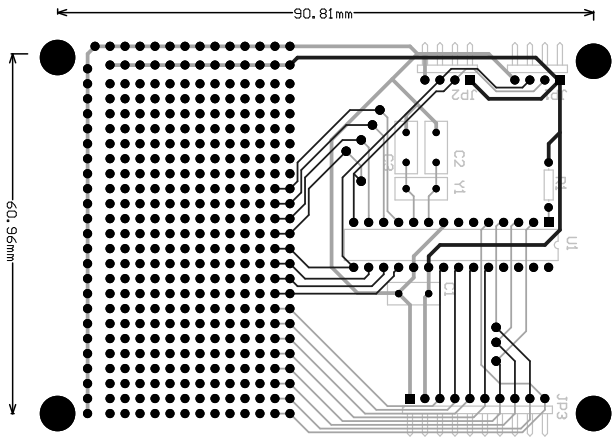
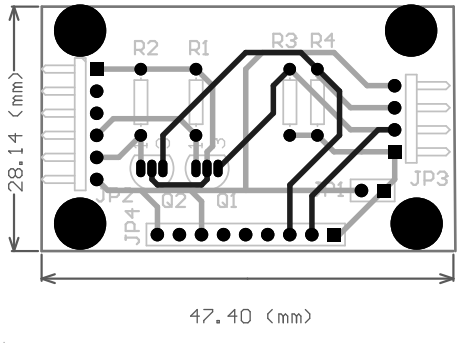
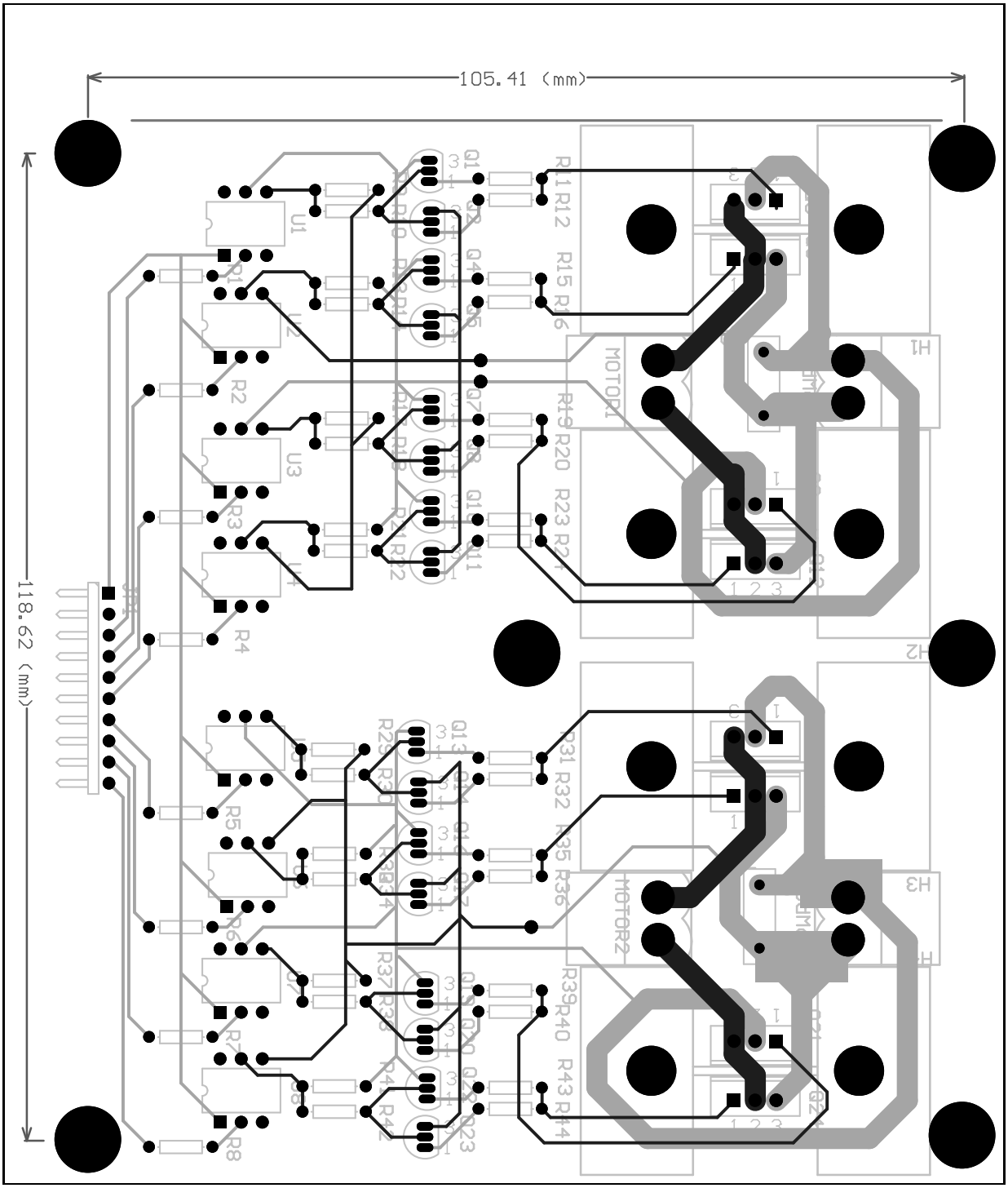


Figure B.2: I^{2}C Level Translator board layout



After it was discovered that the I^{2}C level translation was not necessary the transistors and pull-up resistors were omitted and the data and clock lines were bridged. This board is still necessary for the compass mount and interfacing to the microcontroller boards.

Figure B.3: H-bridge board layout



This circuit layout is from an older version of the circuit which did not function correctly. However, the board was modified to reflect the new circuit diagram, with the exception of the inputs within each pair being swapped.

Appendix C

Interfacing Software

C.1 Wireless Ethernet

The following lines show the output for the `ifconfig` and `iwconfig` commands when run on the wireless network interface `ath0`. The first command shows the networking information common to all interfaces, wired or wireless, such as IP address and packets sent and received. The second shows the wireless related information such as frequency, link quality, etc.

```
root@jericho:~# ifconfig ath0
ath0      Link encap:Ethernet  HWaddr 00:09:5B:C7:0D:AE
          inet addr:192.168.0.237  Bcast:192.168.0.255  Mask:255.255.255.0
          UP BROADCAST RUNNING MULTICAST  MTU:1500  Metric:1
          RX packets:569 errors:14 dropped:0 overruns:0 frame:14
          TX packets:193 errors:0 dropped:0 overruns:0 carrier:0
          collisions:0 txqueuelen:199
          RX bytes:62947 (61.4 Kb)  TX bytes:20011 (19.5 Kb)
          Interrupt:10 Memory:de87e000-de88e000

root@jericho:~# iwconfig ath0
ath0      IEEE 802.11  ESSID:"robocuprescue"
          Mode:Managed  Frequency:2.462GHz  Access Point: 00:0F:66:AA:1C:D8
          Bit Rate:36Mb/s   Tx-Power:off   Sensitivity=0/3
          Retry:off   RTS thr:off   Fragment thr:off
          Encryption key:off
          Power Management:off
          Link Quality:66/94  Signal level:-29 dBm  Noise level:-95 dBm
          Rx invalid nwid:0  Rx invalid crypt:0  Rx invalid frag:0
          Tx excessive retries:0  Invalid misc:0   Missed beacon:0
```

C.2 Electronic compass

The following code shows basic usage of the I^2C bus to communicate with the CMPS03 compass.

```
/* compass.c
 *
 * Software to read from the CMPS03 magnetic compass.
 *
 * By Josh Marshall, joshua dot marshall at studentmail dot newcastle dot edu dot au
 * October 15, 2004.
 *
 * Adapted from code for the SP03 speech synthesiser
 * By Bram Stolk, b.stolk at chello.nl
 */

#include <errno.h>
#include <string.h>
#include <stdio.h>
#include <stdlib.h>
#include <unistd.h>
#include <fcntl.h>
#include <linux/i2c-dev.h>

/* Note that the documentation for the compass states its address as 0xC0.
 * However, this includes the low bit which specifies read or write.
 * Linux i2c does not include this bit in this address, so the actual
 * address is 0xC0 shifted down, 0x60.
 */
#define CMPS03_ADDR 0x60

/* The important registers on the compass. Internal/test registers omitted. */
#define CMPS03_SOFTWARE_REVISION 0x0
#define CMPS03_BEARING_BYTE 0x1
#define CMPS03_BEARING_WORD_HIGH 0x2
#define CMPS03_BEARING_WORD_LOW 0x3
#define CMPS03_CALIBRATE_CMD 0xF

int main(int argc, char *argv[]) {
   char *end;
   int res,file;
   int e1;
   char filename[20] ;
   long funcs;

   int heading_byte, heading_word_h, heading_word_l;
   int bearing_long, bearing_degrees;

   sprintf(filename,"/dev/i2c-0");
   if ((file = open(filename,O_RDWR)) < 0) {
     e1 = errno;
     if (e1 != ENOENT) {
       fprintf(stderr,"Error: Could not open file '%s' : %sn",
               filename,strerror(e1));
       if(e1 == EACCES)
         fprintf(stderr,"Run as root?n");
     }
   }

   if (ioctl(file,I2C_SLAVE,CMPS03_ADDR) < 0) {
     if (errno == EBUSY) {
       printf("device is busyn");
     }
```

```
    printf("Got error: %d\n", errno);
    exit(0);
  }

  /* Get software revision number */
  res = i2c_smbus_read_byte_data(file, CMPS03_SOFTWARE_REVISION);
  if (res < 0) {
    printf("Cannot read software revision leveln");
  } else {
    printf("Software revision level: %02xn", res);
  }

  /* Loop and read from the compass. */
  while (1) {
    /* The heading byte is 0-255 for the 360 degrees. */
    heading_byte = i2c_smbus_read_byte_data(file, CMPS03_BEARING_BYTE);
    if (heading_byte < 0) { printf("Error reading from compass."); exit(1);}

    /* The high resolution heading is given in two registers, and is 10 * the
     * heading in degrees, ie 359.9 degrees reads as 3599. */
    heading_word_h = i2c_smbus_read_byte_data(file, CMPS03_BEARING_WORD_HIGH);
    if (heading_word_h < 0) { printf("Error reading from compass."); exit(1);}
    heading_word_l = i2c_smbus_read_byte_data(file, CMPS03_BEARING_WORD_LOW);
    if (heading_word_l < 0) { printf("Error reading from compass."); exit(1);}

    /* Combine the two bytes, and get the heading in degrees. */
    bearing_long = heading_word_h * 256 + heading_word_l;
    bearing_degrees = bearing_long / 10;

    printf("Bearing: %d \n", bearing_degrees);

    /* Wait for a while. */
    usleep(200000);
  }
}
```