Arduino Based Obstacle Avoidance Robot

Final Report for CS39440 Major Project

Author: Daniel Atkinson (daa9@aber.ac.uk)
Supervisor: Prof. Dave Barnes (dpb@aber.ac.uk)

20th April 2013
Version: 1.0 (Release)

This report was submitted as partial fulfilment of a BSc degree in
Computer Science (G401)

Department of Computer Science
Aberystwyth University
Aberystwyth
Ceredigion
SY23 3DB
Wales, UK

Declaration of originality

In signing below, I confirm that:

- This submission is my own work, except where clearly indicated.
- I understand that there are severe penalties for plagiarism and other unfair practice, which can lead to loss of marks or even the withholding of a degree.
- I have read the sections on unfair practice in the Students' Examinations Handbook and the relevant sections of the current Student Handbook of the Department of Computer Science.
- I understand and agree to abide by the University's regulations governing these issues.

Signature .....

Date .....

Consent to share this work

In signing below, I hereby agree to this dissertation being made available to other students and academic staff of the Aberystwyth Computer Science Department.

Signature .....

Date .....

Acknowledgements

I am grateful to my supervisor Dave Barnes for encouraging this project.

I'd like to thank my parents for supporting my work on this project as well as proof reading this report. I would also like to thank Martin Evans for also proof reading this report.

Abstract

The aim of this project is to physically build and write the software for a robot. This robot should be able to drive around within its environment under its own power without colliding with any obstruction. It should also be able to see areas of the environment using some form of sensor system in order to determine which direction to travel safely.

CONTENTS

LIST OF FIGURES

Chapter 1

Background & Objectives

I was first exposed to electronics in an academic environment in high school. This was only very basic circuitry, such as making a light flash by using simple integrated circuits. Being introduced to integrated circuits made building an electronic timer much easier, which was the first thing I produced using these small chips. This was very satisfying when it finally worked, a feeling I still get when something I make that works as intended.
Fast forward to college five years later and I am still fascinated by electronics. Still using these wondrous little chips to build more interesting circuits I built an audio amplifier whereby I input a waveform into the circuit, either generated by a signal generator or my guitar, and amplify it or smooth the signal to create a new sound, then output this amplified signal to a speaker. This distorted sound is similar to those created by a guitar amplifier that has built in effects or an specific effects pedal also used by guitarists.
At college I also took a computing class in which the programming language Visual Basic was taught as part of the course. Naturally the next step would be to combine the electronics with the programming knowledge. This took the form of a small blinking light project where I use a PIC (Peripheral Interface Controller) to flash an LED (Light Emitting Diode). A PIC is a small chip (Integrated Circuit) which can run small amounts of code to read inputs and control outputs on its various pins.
In the summer between the end of College and starting University I discovered a range of open source hardware microcontrollers called Arduino. These boards made combining program code and electronic hardware much easier by doing much of the base work for me. These microcontrollers have a large community, which has written all forms of libraries to interface the board with various pieces of hardware and control them with much less effort than would be needed when using a PIC. The PIC does have a large variety of libraries but the Arduino seems to have a much wider variety of what is supported and a very active community to help if you get stuck.
I have also had some experience using the pioneer research robot created by Adept MobileRobots [8] which are used by Aberystwyth University in the robotics lab. The experience with these robots was to use their ultrasonic sensors to try and avoid hitting some polystyrene boards. Due to the limited time available to use these robots the resulting code was not very effective or polished, but it has heavily influenced my ideas for designing my current project and further pricked my enthusiasm for robotics and all of the possible applications it has.

My main objective with this project is to produce a piece of hardware that can manoeuvre itself around an environment under it's own power without bumping into anything. This is to be built utilising the knowledge I have gained about electronics and programming from previous projects

and from the courses I have attended as part of my University degree.

Chapter 2

Development Process

2.1 Introduction

I chose to use the iterative and incremental approach to development. This is mainly because of how modular my project is. In theory, I can add more functionality with minor adjustments to the core system, thus making iterative/incremental very suited to my needs.
Each part of the system in an incremental strategy can be developed independently and slotted together as they reach completion.
Each iteration is a review of the previous which has been reworked and improved upon.
For a well functioning system it needs good design, quality programming and a good debugging process. After designing the initial system and writing a simple prototype it is then time for debugging it to get an indication of what the main flaws are. Once these flaws have been clearly identified a new design has to be drawn up to correct these issues. After writing the new version following the revised design the cycle continues in the same manner, design, write then debug.
Simply put:

1. Design robot
2. Build prototype
3. Test robot
4. Debug problems
5. Repeat until perfect

2.2 Modifications

No real modifications were made to this development process as it works for individuals and for teams without alteration.

2.3 Version Control

This is very useful in any project which involves managing code or documents digitally. It is even useful as a backup tool, to be safe from accidental deletions, hard drive failure or any number of other unfortunate occurrences (for instance a fire destroying your computer) as you can just

re-download the files, as long at you are not using a locally stored version control system and do have an off site repository.
There are other features that modern version control systems offer that are of more use in this type of project. Branching and merging are two of the most used features. These enable the user to make a branch within the project in which they can work on a specific feature independently of the main project. You may make multiple branches at the same time and work on different things all independent of each other. If you imagine a tree, where the trunk contains the current working state of a project, then if you want to change something or create something new you make a branch which shoots off from the tree but contains all the information that is in the trunk. You can then work on it independently from the trunk, even if you or somebody else makes another branch from the trunk, the changes made in your branch will not affect it. Once these are finished you can merge them back into the main project, this is a very nice feature version control systems offer as it performs most, if not all of this for you, instead of having to manually try and integrate each line of the branch files back into the main ones.
I have chosen to use Git for developing this project due to how powerful the merge feature is as well as a website called Github [6] which will host repositories for people. The website also has nice usage statistics and offer some private repositories to students. Github repositories are normally open to the general public unless you pay a fee for having non public facing ones. Being a student enables me to have a small number of these private repositories which let me control when I am ready to release a project to public viewing.

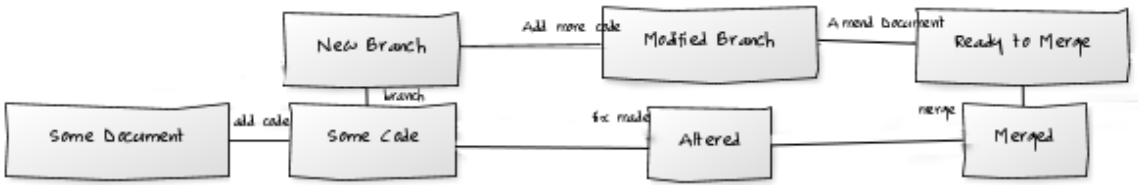

Figure 2.1: Branch Merge Diagram

Chapter 3

Design

3.1 Overall Architecture

The initial design for the robot is to produce a small vehicle capable of moving freely within an environment under it's own power with a platform for mounting the various systems. These systems should be a central control unit, motor control and the various sensors.

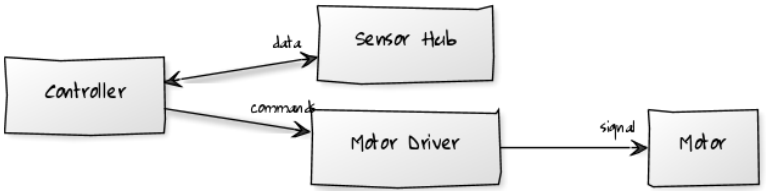

Figure 3.1: Basic system diagram

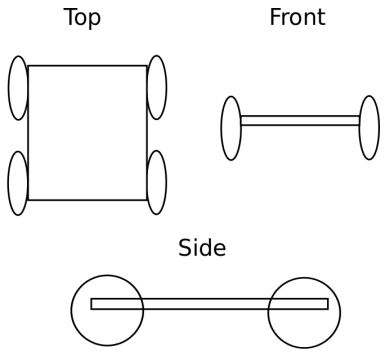

Figure 3.2: Initial design

The central control unit will be a microcontroller for ease of interfacing directly with hardware as well as keeping power consumption down. Keeping power consumption to a minimum is important so that the robot can be active for a longer period of time without needing to be recharged. This controller will interface with both a motor control system and the various sensors required to detect objects in the environment local to the robot.

3.2 Design Research

The various components that the project will need to come together into a finished product have many different possible options to choose from, all with their various strengths and weaknesses.

3.2.1 Materials

There were several considered materials for the robot chassis to be built of and the ideal material would be:

1. Extremely difficult to break so that it does not get damaged easily while in use.
2. As light as possible to make it easier on whichever drive system is used to move it.
3. To be as low cost as possible so that it is affordable to a student.
4. Something that does not conduct electricity which could possibly damage an electrical component mounted onto it.
5. Hard to bend to avoid being bent out of shape by any possible heavy load the robot may end up carrying or colliding with.
6. Easy to cut so that it is possible to craft into the required shape.

- Wood
  This would be the easiest material to make the chassis from as it is very cheap, easy to cut into the intended shape and easy to mount components on with either adhesive, nails or screws. Also the fact that it does not conduct electricity will help when mounting circuit boards to it.
- Plastic
  The lightest option. Good due to its low weight but may not be as strong as wood or a metal option and could bend or snap under the load of heavier components such as motors or a large power source. It can be more expensive than wood to acquire. There is a higher difficulty in cutting it into the desired shapes. It is also non-conductive, again useful to mount electronic components to. Plastic can hold a static charge which can be damaging to electrical components.
- Steel
  A stronger material that can withstand a much heavier load than wood of plastic. It may be able to withstand a heavier load before it is bent out of shape but is itself more heavy than the weaker options of wood or plastic. This extra base weight before adding anything else will put more strain onto which ever drive system that is used to drive the robot and may even need to use more powerful motors because of this extra weight. It is a very conductive material which means that a non-conductive mounting platform will also be needed to mount electronic components as to avoid damaging them.
- Aluminium
  A much lighter metal than steel, but still much heavier than wood or plastic or the same thickness. It can also withstand heavier loads than wood or plastic but it also much more difficult to cut. Again aluminium is a very conductive material meaning that a non-conductive mounting platform will be needed. It can also be used as a heat sink for the components

that can get very hot such as the motor drivers or the motors themselves. A heatsink is a material attached to something that gets very hot and conducts that heat. It generally has a large surface area to dissipate the heat into the cooler air around it, but it may also have a fan to blow/draw the hotter air away replacing it with air/gas with a lower temperature than that of the heatsink.

Aluminium seems to be the best all round choice being strong but not as heavy as steel. It can act as a heat sink if the motors are mounted directly to it. It is also not very expensive to buy in small amounts.
In addition to the aluminium base I have decided to use plastic for mounting components to the base as it is light, inexpensive and non conductive which is suitable for electronic components.

3.2.2 Actuators

Actuators are motors used for controlling movement of a system. The ideal actuator would be something that:

1. Has little to no power consumption which will increase the overall total runtime of the robot by decreasing the energy draw on the system.
2. Has as much torque as possible to make moving a larger mass less effort.
3. Weighs as little as possible to reduce the overall weight of the robot.
4. Is as low cost as possible again to be affordable to a student building this robot.
5. Has perfect precision in order to know exactly what position it is in at any time.
6. Have no limits on rotation as some actuators can only turn a set amount before having to turn back.
7. High speed range so that it can move as slow as needed or as fast as needed.
8. No wear and tear for easy or no maintenance.

- Servo
  Typical servos are a motor and a gearbox with a potentiometer, a voltage divider in this case used to determine how far a motor has turned, for feedback. These motors are great for controlling such things as the direction of sensors or moving very light devices. Servos are low voltage, typically 4.8 - 6 volts, and as such do not have much strength, which means that they are typically not good for driving larger equipment. Additionally most servos only turn up to 180 degrees or 360 degrees. In normal operation they do not turn continuously but can be modified to do so at the cost of losing the feedback of how far the motor has turned.

Figure 3.3: Servo Motor - robotshop.com

- DC Motor
  Direct current motor has a very simple operation: apply current to one side of the motor to make it turn, reverse the direction of the current to reverse the direction the motor turns. Changing the speed of these motors is simple, either change the voltage (keeping it within the devices tolerances) or turn the current supplied to the motor on and off at high speed where how quickly it is alternated determines the speed of the motor. Typically these motors are attached to a gearbox to gain more torque to drive much higher loads. Optical rotary encoders can be used to determine how much the motors have turned and how fast. An optical rotary encoder can be fitted to any motor to add the functionality of positional feedback. These encoders use a light based sensor to detect when the light changes in front of it, this can be used with a disc that has black and white lines on it. The change in color is detected, this along with how many times it changes and with what frequency this happens can determine the amount a wheel has turned and how fast it has done so.

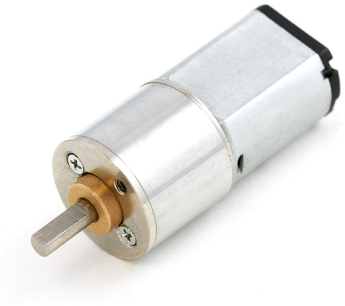

Figure 3.4: DC Motor - sparkfun.com - CC BY-NC-SA 3.0

- Steppers
  Stepper motors use an internal gear and a ring of magnets. These magnets pull the gear into position, powering the magnets in sequence which will turn the motor. Each part of this cycle is called a step. This means that a single step is a known amount of rotation. Using this type of motor ensures that you can accurately turn whatever is attached to the motor shaft by a known amount without any additional measuring equipment. But adding any such measuring equipment can add verification to determine if the motor has actually rotated by the amount expected.

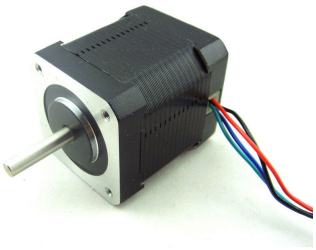

Figure 3.5: Stepper Motor - stepperonline.com

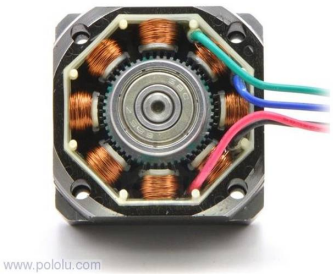

Figure 3.6: Stepper Motor Internal- robotgear.com.au

I have chosen to use stepper motors due to the ability to control the amount and speed of rotation with more accuracy than the alternatives. Stepper motors do come in high torque version which may be needed for this project as the chassis is made of metal which is a much heavier material. A stepper motor could be used with a chassis of any of the materials mentioned, it may struggle with steel depending on how thick of a piece is used. DC motors could also be used with all materials if in conjunction with a gearbox, but the additional system needed to measure and control the exact rotation of the wheels using this method puts me off of the idea. Greater power but less accurate control.

3.2.3 Drive Type

There are various different ways to move a mass around. The ideal drive system should be:

1. A very low mass to reduce overall load on the system.
2. Capable of omni-directional movement to increase the ease of moving within confined environments.
3. Perfectly stable so that the robot doe snot fall over or collapse.
4. Have high traction as to not slip on it's environment.

No single drive system is going to be perfect in all aspects.

- Wheels
  These are most conventional method of movement which are seen by most people every day. Robots that use wheels to enable movement are often very stable and can be using in many configurations.

- Two Wheels
  This is an unstable configuration which requires it to be in constant motion to maintain an upright position. When configured with the wheels side by side the center of gravity will be directly in the middle between the wheels underneath the axle where a weight is often positioned to help keep it upright as well as a tilt sensor to help compensate for movement.

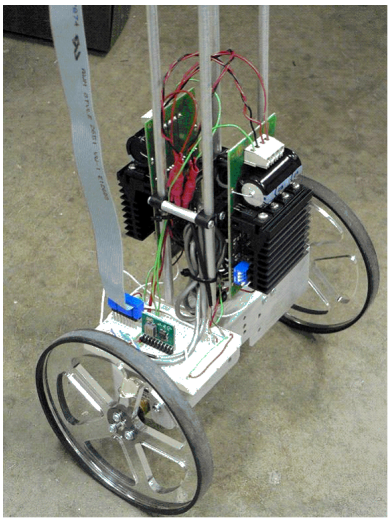

Figure 3.7: 2 Wheel Gyroscope - ni.com

  Another way for a two wheels to be arranged would be like a bike where again it will need to be in constant motion to remain upright.

- Three Wheels
  Most commonly two drive wheels and a single free turning wheel for stability. To turn the wheels are either turned in the opposite direction of each other or in the same direction just as different speeds using the free turning wheel to stop it falling over or being pulled along the ground as long as the center of gravity if within the three wheels.

- Four Wheels
  All four wheels can have independent control for tank style movement. This is harder to keep in a straight line due to having to try and keep all four wheels turning at the same speed otherwise risk being inefficient and also causing additional friction dragging one side back. An advantage to having all four wheels operated independently is that if using in conjunction with 'omni' wheels. These wheels have rollers on them enabling then to traverse in a lateral direction if the front and back pair are rotated in the opposite direction of each other.
  Car type configuration is also four wheels but only either the front or the back pair are powered and either the front of back pair are able to turn. Both the wheels that are able to change direction and the powered drive wheels can be the same pair. It will require some kind of servo system to turn the axis which is designated at the turning wheels.

- More than Four Wheels
  Very good for uneven terrain as each wheel can rise and fall independently with less effect on the main structure. These are far more complex to operate but if the terrain

is very rocky then it may be the best choice.

- Tracks
  These are very stable and have the best traction compared to wheeled variants as long as the surface it is used on is not smooth, sand is a good place to use tracks due to the large surface area compared to wheels as they will not sink while wheels will. Very high friction with the ground especially when turned as it has to operate the tracks in the opposite direction of each other forcing its way around in a circle.

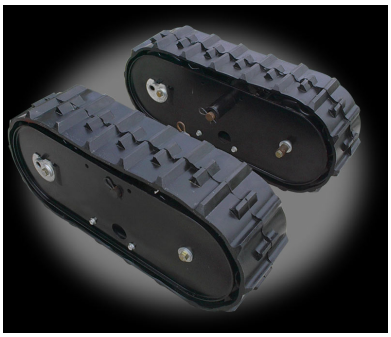

Figure 3.8: Tracks - robotcombat.com

- Legs
  These are the most uncommon of the movement types. Due to the cost of developing systems using this type of movement and the overall complexity of controlling them.
  - Single Leg
    The most difficult to control due to having to balance its load upright on a single point and having to lean in the direction it needs to go without falling over and 'hop' to move around without falling over when landing.

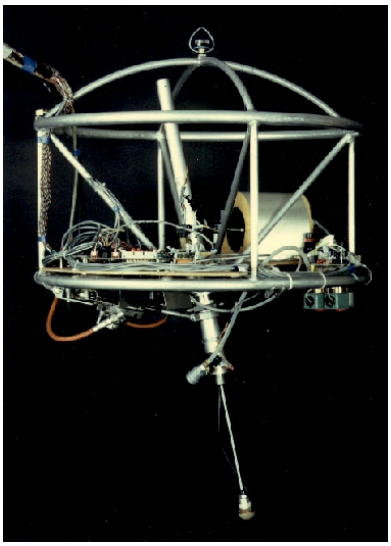

Figure 3.9: 3D One Legged Hopper - mit.edu

  - Two Legs

Extremely difficult to control and there are very few working practical applications of a two legged robot. The main issue for controlling this is keeping it upright and not overbalancing, also if knee type joins are added the complexity significantly increases. Also the mere fact that work on two legged robots is being done by a cutting edge engineering company like Boston Dynamics [2] is evidence that this kind of system is a fair bit beyond the scope of an undergraduate student project.

Figure 3.10: NAO - aldebaran-robotics.com

- More than Four Legs
  Having more legs can increase stability. This is because most of the legs can be firmly on the ground while the others are moving into their new position. Turning can be more complicated to control than the walking already is. The servos used to control each of these legs and all the joints the legs may have may need to be quite strong or at least have strong teeth as to not get stripped under the weight of whatever payload the robot may have on top of all of these legs. This is based on how insects are constructed where their body is vastly bigger than its legs, and as such it has many legs to compensate for this and distribute the load.

I have decided to use a four wheel configuration to take advantage of the level of traction four wheels will give the robot as well as the added stability over the two or three wheeled configuration. The configurations with more than four wheels have added complexity in how to control all of the extra wheels and the advantages of having the added wheels are not needed in this project. Also the legged variety of drive systems are well beyond the scope of this project and the added complexity would not bring any real benefits tot he robot for it's intended purpose of avoiding obstacles in the environment.

3.2.4 Sensors

Sensors are needed so that the robot is able to see what is in the environment around it, enabling it to see, calculate and avoid obstacles.
An ideal sensor would be:

1. Infinite range, being able to see as far as possible and as close as possible.

2. Can no be interfered with, able to be used in any environment and nothing can interfere with the readings it provides.
3. Little to no processing required to acquire and interpret the data provided by the sensor.
4. Lightweight as to not increase load on the drive system.
5. Small so that many can be fitted into a small space for maximum obstacle sensing ability.
6. Cheap as possible to be able to afford many.
7. High resolution as to be able to determine what exactly the objects in the environment are.

Such a wonderful sensor does not exist but there are many options.

- Bump Skirt A bump skirt is multiple touch sensors such as buttons arranged like a skirt around the edge of the robot. When the robot bumps into something the button is pressed telling the robot where it bumped into something. From this it will be able to move in the appropriate direction to get away from the object it hit. This is a very bad way of avoiding an obstacle because to know that something is there it will have to hit it first.
- LDR
  An LDR is a light dependant resistor. A small resistor that changes its resistance depending on how much light it is exposed to. This could be used to detect if the robot is very close to bumping into an object and avoid it as the object got closer and possibly cast a shadow onto the sensor reducing the amount of light the resistor can detect, similar to a physical bump skirt which activates when something touches it.

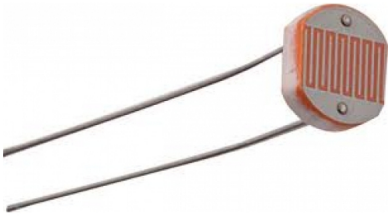

Figure 3.11: Light Dependant Resistor - robotics.org.za

- Camera
  A camera could be used to detect objects in front of it by using various image processing techniques to analyse the data it provides. This method is good because it can potentially map a relatively large area in a single image. However it requires more processing to do so, which can be slow and result in colliding into an object or being stuck in a tight space before the system has finished processing data from the camera. This method would force me to use a more powerful processor to overcome the limits of the lower speed processor models but it adds complexity to the project as well as additional cost and a higher power consumption.

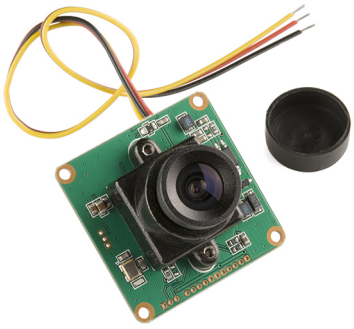

Figure 3.12: Camera Module - sparkfun.com - CC BY-NC-SA 3.0

- Infrared
  Used to detect distance from an object. An emitter and a receiver pair linked to work like the light dependent resistor but using infrared instead of normal visible light. Depending on the intensity of infrared picked up by the receiver it can be used to determine the distance from the source of the reflection. Ambient infrared can effect readings as there is infrared radiation emitted from the sun and is everywhere. This extra radiation other than the amount emitted by the sensor is unneeded and unwanted and as such if it arrives at the receiver it will provide false readings, therefore this can be very inaccurate.

Figure 3.13: Infrared Sensor - coolcomponents.co.uk

- Sonar
  An emitter style approach. It emits an ultrasonic wave to bounce off of whatever surface is in front of it. The time taken from emitting the wave until receiving the wave determines how far away the object is. This method comes with its drawbacks. Due to how sound waves behave when they interact with the environment by bouncing off of it. If the surface is angled or curved the sound can bounce away from the receiver, either not reaching it at all giving the possible false reading that there is nothing in front of it, or it could bounce off of multiple surfaces back to the receiver giving a false reading that an object is there but further away due to the sound taking longer than it should have to reach the receiver.

Figure 3.14: Ultrasonic Sensor - coolcomponents.co.uk

A combination of both sonar and infrared logically seems like a good idea. One can compensate for the others weaknesses. Use the sonar to compensate for ambient infrared and the infrared can be used to compensate for sonar bouncing around the environment. Hopefully this will reduce the number of false readings produced. This does not remove the significant weaknesses of ambient infrared or the sonar echo but should help identify which readings are being affected by comparing the two different readings taken by the different sensor types.

3.2.5 Control

The robot will need a controller, that connects the software to all the hardware.
The ideal control unit would be:

1. As low power consumption as possible to increase overall total runtime of the robot.
2. As fast as possible to be able to handle any amount of processing ti may be required to do, such as analysing vast amounts of sensor data in real time.
3. As much memory as possible to be able to store all the data it will be processing as it works on it.
4. Easy to program with support for any language the user may want to write in.
5. If it to be mounted on a circuit board to be used within a larger integrated system then it should be in DIL format for easy soldering to a board via the through hole method where the legs of the chip are placed through holes in a circuit board and soldered in place.
6. Access to general purpose input and output pins, these are to interface directly with other pieces of hardware.

If a unit like this actually existed it would be perfect but the real suitable choices are as follows.

- PIC
  Peripheral Interface Controller. Very low cost microcontroller with a small easy to learn instruction set and support serial communication/re-programming. They also come in a DIL package (dual in-line) making them easy to incorporate into through-hole printed circuit boards as the legs of the chips can fit through these holes and be soldered (held in place with a low melting point conductive metal alloy.) into place.

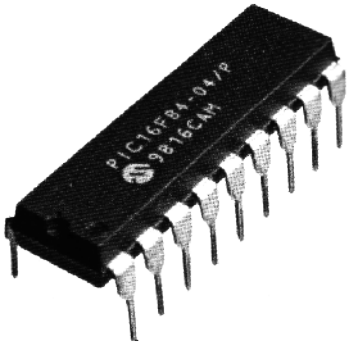

Figure 3.15: PIC - circuitstoday.com

- Arduino
  An open source hardware board that is cheap but not as cheap as a PIC. These are very popular among hobbyists due to them being very easy to use and having a vast collection of community written libraries to interface with all different types of hardware.
  Arduino uses C or C++ programming language for development, this can be a good thing due to the fact that it provides the programmer with a large amount of control over exactly how the program runs but is more complex to use than some other languages such as python or java which handle a large amount of the low level operations for the developer.

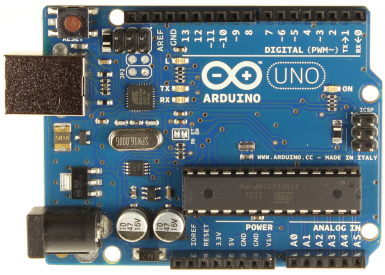

Figure 3.16: Arduino arduino.cc

- Netduino
  This is also an open source electronics prototyping platform but instead of being based on C and C++ it is based on the .Net Micro Framework which is Microsoft's version of an embedded framework. This could be useful to somebody who is very familiar with a .NET language such as C#. The main disadvantage is that this framework requires a faster processor to run and as such requires more power to operate making this option have higher overheads on power and processing than the PIC and Arduino alternatives.
  These boards also cost more than the Arduino and PIC, and have much less community support.

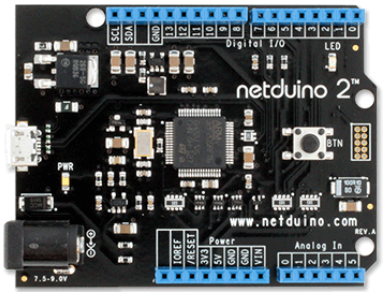

Figure 3.17: Netduino - netduino.com

- Motherboard
  A small motherboard that can be found in a home computer or a netbook/laptop. These have the widest variety of applications. It can support most operating systems and programming languages but come at the hefty price of power consumption. Compared to microcontrollers, a full motherboard draws a very large amount of power to run. Also they take far longer to power on due to running an operating system, unlike microcontrollers that have the code compiled down and run directly on the hardware itself. This can also be done with these larger boards but is far more complex to do and is out of the scope of this project.
  The cost of such boards is also very high as they are far more complex pieces of electronics.

Figure 3.18: Atom Motherboard - intel.co.uk

- Raspberry Pi
  The Raspberry Pi is a credit card sized computer that is an embedded platform for Linux and various other operating systems. It is very cheap and runs much faster than most microcontrollers. It does also have the downside of long startup times due to running a full desktop style operating system on such a compact board. This could also be compiled to run directly on the ARM processor this board uses but is also out of the scope of this project. Unlike normal motherboards this little board has some GPIO (general purpose input output) pins for interacting directly with various pieces of hardware like the microcontrollers do.

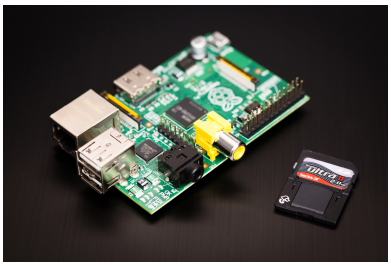

Figure 3.19: Raspberry Pi - raspberrypi.org

3.2.6 Power Source

As this robot is intended to move around freely the power source should be:

1. Unhindered by power and data cables, the power source cannot be supplied by a wall power outlet, it has to be self contained. This means it will have to be a battery.
2. The battery will have to be several cells or a single high output cell due to the size of motors intended.
3. It will need as many Amp hours as possible for longer run times. This could be achieved with several cells linked together in series (end to end) to increase voltage and/or link more together in parallel (side by side) to increase amp hours (runtime).
4. It will have to be small enough to fit on the robot.
5. It should be light as to not increase the load on the drive system too much.

The possible options are:

- Lithium Polymer
  LiPo batteries come in up to 11.2 volt packages, which is not quite high enough for some higher voltage motors which I may be using such as the high end steppers. These batteries are much lighter than some other alternatives and are common in embedded devices. The amp hour ratings of these batteries can range from 100 milliamp hours to around typically 6 amp hours in the size packages that would be suitable for a small robot.

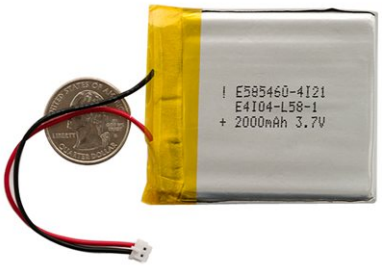

Figure 3.20: Lithium Polymer Battery - robotshop.com

- Lead Acid
  A lead acid battery is a choice with a high output. A single battery can output 12 volts and can be found in high amp hour packages such as 1 - 40amp hours compared to the lithium alternatives which are around 0.1 - 6 amp hours. Due to already needing a high power motor for the chassis, a higher weight battery is not too much of an issue and provides the benefit of the higher power output to drive the more power hungry motors.

Figure 3.21: Lead Acid Battery - kestrelelectricalsupplies.co.uk

3.2.7 Wheels

As I have chosen a wheeled solution for the drive system of the robot the wheels supporting it must be appropriate.
The perfect wheels would be:

1. Made of extremely light materials thus being easier to turn.
2. To be very strong to support all of the weight of the robot on them.
3. To have a high level of traction as not to slip on the environment it may be traversing.
4. As cheap as possible to keep overall costs to a minimum.
5. Be easy to mount to the drive shaft of the motors being used to move the wheels.

- Off Road
  Having chosen to use stepper motors for the reason of increased accuracy over the less precise DC motors it would be a bad idea to go for wheels which have insufficient traction and as such off road style wheels which have a wide and deep tread with spikes for additional grip would be a suitable solution.
  These wheels measure 120mm in diameter and 60mm wide which results in it being harder to turn and will require a stronger motor to move.

Figure 3.22: Offroad Wheel - sparkfun.com

- Colson Wheel
  These come in sizes ranging from four inches to twelve inches in diameter with a tread width of between two and three inches. They have a rubber exterior for grip with an iron core for mounting. They are shock absorbent but this should not be an issue for this project due to the slow speeds that it is intended to be moving at. The iron core makes these wheels very heavy weighing around four pounds for the smaller wheel and up to sixteen pounds for the larger versions.

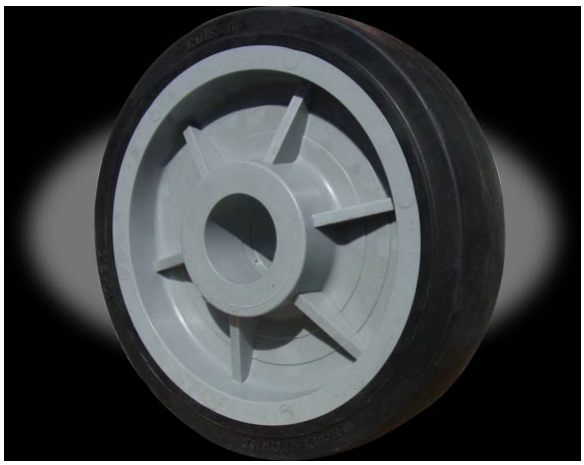

Figure 3.23: Colson Wheel - colsoncaster.com

- Omni Wheel
  These wheels have been briefly described in the four wheeled drive system section but here I will go into more detail.
  They have less grip then other wheel types due to the rollers all around the wheel. Their primary advantage is that when two or more are used in conjunction with each other and are

rotated in different directions they can move whatever device they are mounted to directly sideways, this is going against the way the wheels are actually turning. This feature of the omni wheels is a major advantage if the robot gets stuck in a tight space that it cannot turn to get out of or even just a tight turn that it's normal turning circle is not sufficient to traverse.

Figure 3.24: Omni Wheel - sparkfun.com

3.3 Feedback Interface

While operating the robot, if there is any unexpected behaviour it would be nice to have some form of interface to see what the robot thinks the environment looks like.
An ideal solution for this should conform to the following guidelines:

1. There cannot be any cables trailing from the robot to the feedback device because you do not want to have to follow the robot around it's environment just to be able to read what it thinks it is seeing, so a wireless solution would be good.

2. Be lightweight so that carrying it is not a nuisance.

3. Low power consumption so that it does not need to be near a power source all the time. Having a run time the same as or longer than the robot itself would be ideal.

4. Easy to read display which is also low power.

5. Intuitive user interface so that it is very easy to use and enables the user to just use it without any instruction.

All that is needed is a small microcontroller as learned from the previous sections that these are low powered, easy to use and have support for various different pieces of hardware.
Lithium ion batteries are a good fit for this sort of device due to their low weight, low voltage and small size.
A small display is needed.

- Liquid Crystal Display
  These are low powered easy to use devices for displaying information. They are used in everyday machines such as a digital display on an oven or microwave, the screen on a calculator or a vending machine even the screens used in mobile telephones. These screens are very cheap and easily available in all sorts of types and sizes.

Figure 3.25: Liquid Crystal Display - coolcomponents.co.uk

- E-Ink Display These displays are unique in that they only use power to change what they are displaying, once the display has changed it used no more energy to keep the screen displaying it's current image. The main downside to these is getting hold of one that has not been removed from another device such as an e-reader. Some of these displays are also flexible and could be used to wrap around the wrist as a type of watch display. They are also very expensive for a simple display but the power consumption aspect makes them a very attractive option for this type of device.

Figure 3.26: E-Ink Display - dataprotectioncenter.com

A wireless module of some description will be needed to transmit the information from the robot to the feedback device. This device should ideally conform to these guidelines:

1. Have as much range as possible so that no matter how far away the robot goes it can still receive the information from it.
2. Be able to transmit through any material so that no matter what obstacles get between the robot and the feedback device it can still receive information.
3. Consume as little amount of power as possible to increase the run time of the device.
4. Be physically small to keep the size of the device as small as possible.

5. Be as light as possible to keep the overall device weight to a minimum for comfort as the user may be holding it for prolonged periods of time.

The options for this type of component are:

- Wifi
  This is the most common wireless medium for transmitting large amounts of data. It has some low powered modules with a consumption rate of only 100 milliwatts and this low power module can transmit data at 1Mbps (mega bits per second). The downside is that these low power versions are expensive costing around fifty dollars a piece.

Figure 3.27: Wifi Module - antenova-m2m.com

  If a USB connection is available with a host that has the appropriate drivers to run a USB wifi module then a cheaper version is available at around fifteen dollars but these have a much higher processing overhead for any device that is capable of using such a module which in turn uses more power. The module itself is still around 100-500 milliwatts of power usage.

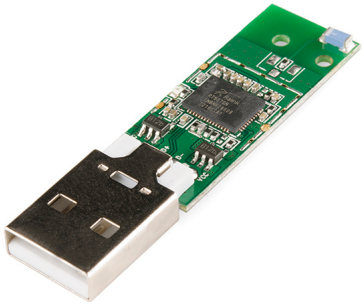

Figure 3.28: USB Wifi - sparkfun.com

  Both of the Wifi module types have a maximum range of around one hundred meters.

- Bluetooth This type of communication is commonly using in mobile telephones to enable such things as hands free calls in an automobile. Low power Bluetooth units that transmit data at around 1200 bits per second typically use only 24 milliwatts of power in doing so. This is much lower than the low power version of wifi but it has severe range limitations of only being reliable within 10-20 meters. The Bluetooth modules available for use in hobbyist projects such as this are rather pricey costing around forty dollars each.

Figure 3.29: Bluetooth Module - sparkfun.com

- Zigbee These little wireless radios are perfect for small, low powered wireless projects. They run on as little as 3.3 volts with an output of 1 milliwatt. The downside is that it's data transfer rate is only around 250 killobits per second, just a quarter of the embedded wifi module. The zigbee module more than makes up for the lack of data transfer with its increased range of up to 100 meters. These are also one of the cheaper options costing around twenty dollars each. These such modules are called xbee which is just an implementation of the zigbee protocol which is typically used for mesh networking but still works for single point to point communications.

Figure 3.30: Xbee Module - sparkfun.com

An Arduino Fio is a good fit as it is small, low powered, has both lithium polymer battery socket and an xbee wireless module socket built in making it the obvious choice for this type of device as most of the connection work has been done already just leaving the screen and a user input method to be added.
A small LCD (liquid crystal display) can be connected to the Arduino to display information it receives over the integrated xbee socket.
A user input interface can be added by using some small push buttons and wiring them to some of the input pins provided on the Arduino Fio.

Chapter 4

Implementation

4.1 Prototype

Building a prototype will quickly highlight the main flaws in the initial design. This is not the same as the intended final version and in this case is not even the same materials as I have chosen. It does however confirm the basic design but is made from much cheaper sourced components. I already had an Arduino Uno (the basic prototyping model) from my interest in the technology before I attended university, so this was an easy component to acquire quickly and is perfect for a prototype. I had no chassis built or any materials to make one so I found a cheap and easy to assemble one online at a hobbyist electronics retailer. This kit also included some very small DC motors with gearboxes and wheels, very convenient little package. The Arduino along with the chassis kit and a small infrared sensors, a 9 volt battery and some jumper wires and a prototype was put together in a short period of time.

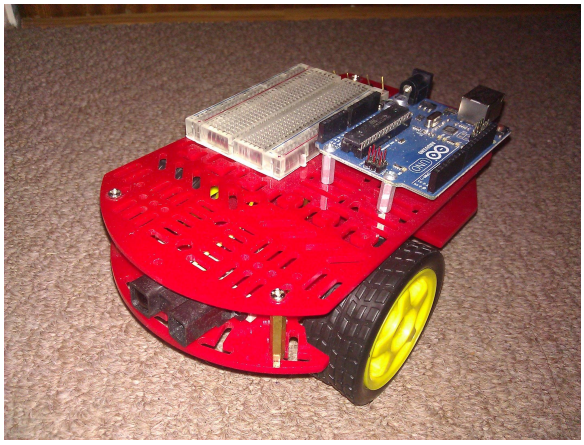

Figure 4.1: Prototype mkI

Wiring up the components was fairly easy due to there only being two motors and a single infrared sensor. The sensor just has 3 pins, ground and positive power pins as well as a signal pin. This is basically set up like a resistor, you supply power to the positive pin, attach the ground to the ground power rail of the system and just read the value being transmitted back on the signal pin. The difference between zero up to the amount of power being given to the sensor, in this case the data sheet specified 5 volts as the upper edge and that is what I supplied the sensor with, gives an indication of how far it is from an object. With this sensor the higher value returned is

actually how close the object is and the lower number indicates it is further away. This is due to the fact that the reading received is indicating how intense the amount of infrared getting back to the sensor is.
The harder part of this was getting the motors to run safely, this issue was discovered when putting together this prototype and testing each part as it was implemented. The Arduino I use for prototyping can only output a regulated voltage of 3.3 or 5 volts. The motors supplied with the chassis kit do not run very well at this voltage and struggle to move on thick carpet. As the supply I am using to power the Arduino is a 9 volt battery this was sufficient to run the motors at an acceptable level, the only problem is supplying this to both the Arduino and the motors. As the microcontroller is needed to control when and how fast the motors are to turn, simply wiring the power supply directly to the motors is a bad idea as they will just spin constantly due to always having power.

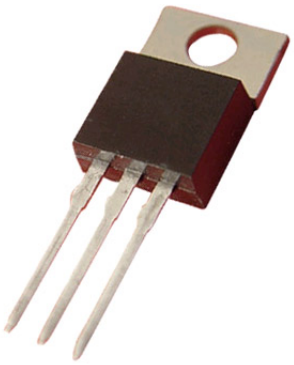

Figure 4.2: Transistor - zmescience.com

This could be solved using a transistor (a semiconductor device used to switch electrical signals) by supplying it with the higher voltage, connecting it to the motor and when a signal voltage from the Arduino is received it switches to the higher voltage allowing the motor to turn. This is a very handy little component which is at the core of modern day electronics, but to use it in this fashion would need a lot more complicated circuitry as to ensure that this higher voltage does not damage other components in the circuit. Another solution would be to use a chip known as a h-bridge.
This chip also acts like a switch but with the addition that it can change the currents direction meaning that you can not only control when the motor is on or off but also the direction it turns without additional complex circuitry.

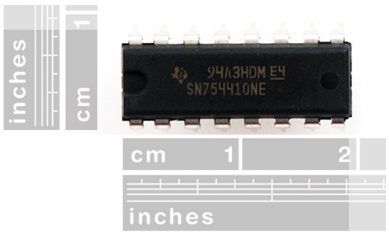

Figure 4.3: H-Bridge - sparkfun.com

I discovered that the h-bridge chip also has its issues, as it generates heat when high currents

are passed through it so if the motors are working hard more current will be drawn and the more heat the chip will generate and possibly burn out. There is again the issue of having no protection for the rest of the circuit. An option that would solve this issue is a full motor driver board but the cost if these is many times the cost of the components to make the circuits myself. For example a h-bridge chip an assortment of diodes, capacitors, resistors and transistors costs around £5 while a fully built board costs around £20-30.

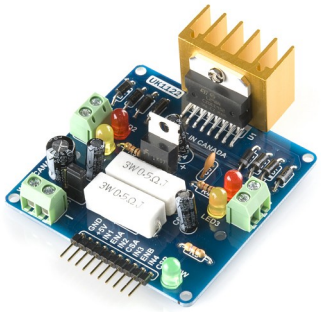

Figure 4.4: Motor Driver - sparkfun.com

I decided to build a simple motor driver using the h-bridge chips, effective for a simple prototype.
With only a single infrared sensor the only logical place to mount it would be to have it facing directly forwards. After writing the code to control the motors and process readings taken from the front mounted sensor the logic to test the concept is very simple. Just check if there is something close in front and if there is just turn and check again, if there is not just keep moving forwards. The logic looks like this:

```
if(sensor_range < value)
{
        motors.turn.right(45);
}else
{
        motors.move.forward(1);
}
```

Figure 4.5: Prototype Code Exert

This seems to work quite well, it does drive forwards and it does turn when an object comes in range of the infrared sensor. This is the desired behaviour but there is a problem. If the robot turns away from one object and into another, if that second object is too close by the time it comes in front of the sensor then it can not be seen by the sensor. The sensor that I have fitted to the prototype has a maximum range of 150cm, but it also has a minimum range of 20cm meaning that is blind to any object closer than this minimum range.
In the figure the red signifies the blind area of the sensor and the green is the visible area. If the robot were to turn right to avoid the wall in front of it then it would collide with the other wall and not be able to detect it.

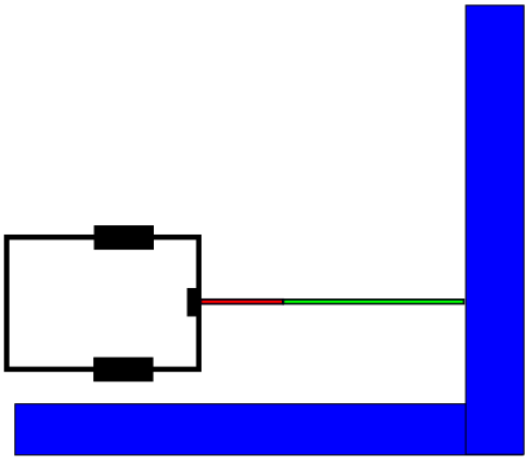

Figure 4.6: Collision Illustration

Another issue with the prototype robot is with the motors. Even though I am supplying each motor with the same voltage they do not turn at the same speed, this is due to the lack of feedback with controlling them. Also the fact that it is built from a very cheap kit is a probable reason for how uneven the speed of the motors is. This all leads to the robot driving in a curve as opposed to the intended straight line, also contributing to the sensor problem of turning into an object putting it within the sensor blind spot.

4.1.1 Prototype Evaluation

The main issues with the prototype are the motors and the lack of more than a single sensor.
The motors did not run at exactly the same speed causing the robot to turn constantly as well as having no feedback system connected to the motors or the wheels and as such the robot nor the user know what position the drive system is in or how far the robot has moved.
Having just a single sensor facing directly forwards highlighted the need for more sensors covering a wider area in front of the robot due to when turning objects may be moved into the blind spot of the sensor and not be seen causing a collision.
These issues were taken into consideration when designing the second iteration of the robot.

4.2 MK-I

The next step was to build a version based on the design and using the things that were learnt from the prototype to improve it.
First thing to do was to build a sturdy chassis for all the systems to be mounted onto. I bought several sheets of aluminium to but cut into a similar shape as the small red plastic prototype.

Figure 4.7: Aluminium Sheet

I chose this shape mainly because of how much I liked working with the kit used in making the prototype. Even though the first version was not perfect, the sensor was successful to a certain degree, it just needed more of them to cover the blind spots. The curved shape of the front where the sensors mounted makes sense because then each sensor reading will provide the same relative distance from the robot unlike if it was square than the edge would be closer to the object in the environment than the robot thinks.

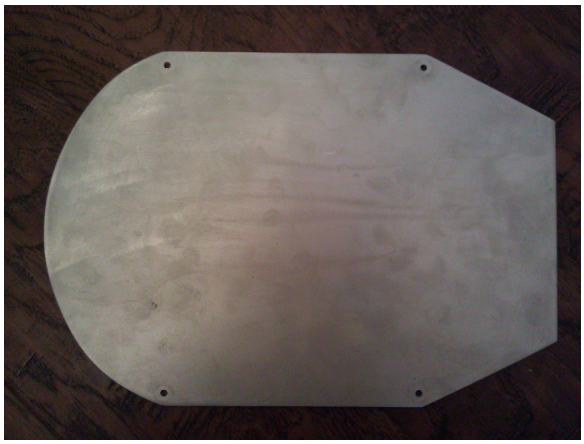

Figure 4.8: Aluminium Sheet Cut

Next was to think about how to mount the motors to this aluminium base plate. After measuring the size of the stepper motors I bought to use as part of the robots drive system I also purchased a strip of aluminium to be cut and shaped into a motor mount to be attached on the underside of the baseplate.

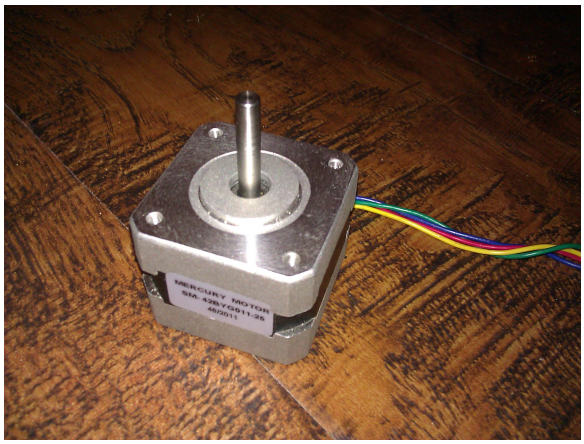

Figure 4.9: Chosen Stepper Motor

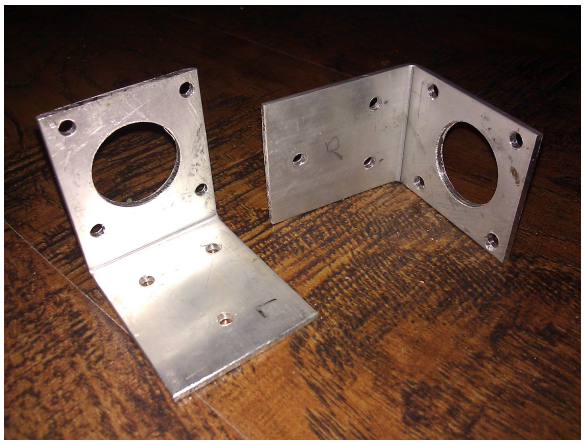

Figure 4.10: Motor Mounts

These mounts are just the cut strips of aluminium that have been bent so that they form a 90 degree angle and holes drilled through them to accommodate the motor drive shaft and for screw holes to attach these mounts to the baseplate.
Another alteration to the original design is the fact that I have chosen to use only two drive wheels instead of the four shown in the design document. The reasoning for this was that it is easier to turn with just the two wheels as it will be able to almost turn on the spot rather than having to perform a multi directional turn, moving backwards and forwards turning a different direction for each just like how a driver would turn a car around in a tight space. Also the kit used in the prototype only used two drive wheels with a caster wheel for stability and was in fact very stable, more than stable enough for this project and removes some complexity from the drive system.

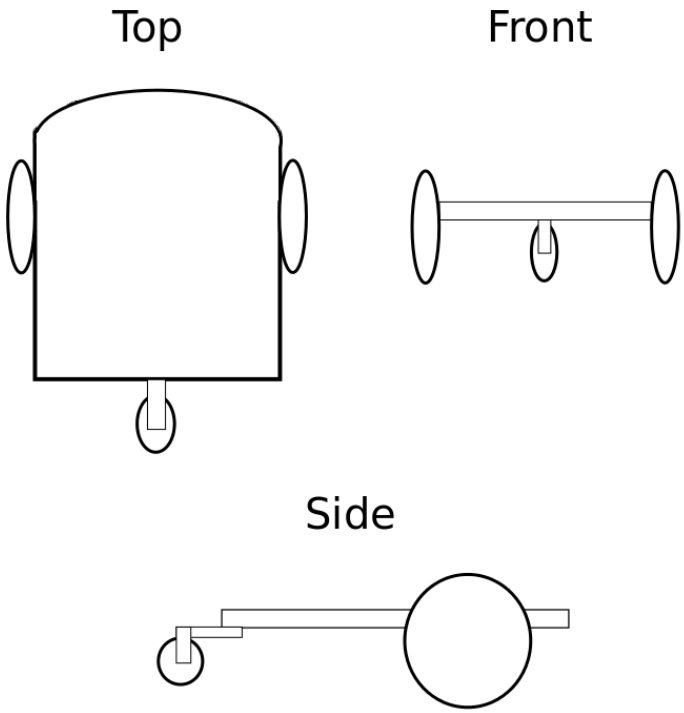

Figure 4.11: Two Wheel Drive Design

Now that the design has been reduced from four drive motors down to only two it will need some other wheels to keep the robot stable. A single rear caster wheel worked well for the prototype and if I went with two rear stabilising wheels then the car turning in a tight space issue would still apply and because of this a single wheel has been chosen.
This wheel again has been mounted on a strip of aluminium bolted hanging over the rear end of the also aluminium baseplate.

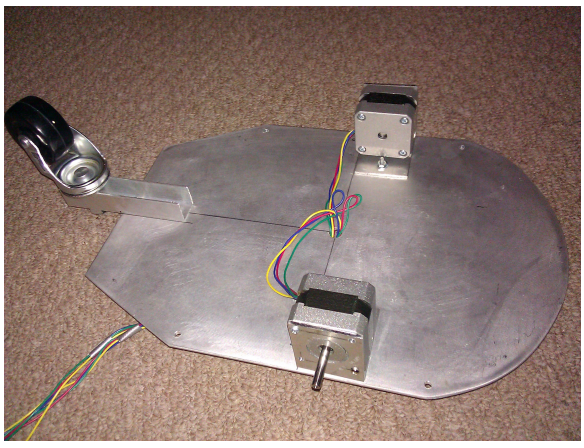

Figure 4.12: Baseplate Underside

Next was to attach the stepper motors to the cut out motor mounting plates and attach the chosen off road wheels to them. The wheels are attached to the motor shaft with the use of a

hollow shaft attached to the wheel and a grub screw through that hollow shaft to clamp onto a flat edge that I cut into the motor shaft for grip.

4.2.1 Manufacturing Parts

Before starting this project I spent some time building a three dimensional printer. This device is just like an ordinary printer but creates three dimensional physical objects.
An ordinary printer uses a print head to dispense ink onto a sheet or paper or card to leave a permanent mark on it which after a lot of small applications of the ink will build up letters or a picture. A three dimensional printer works in a very similar way. Plastic is fed into the print head where it is heated up above the melting point of whichever plastic is being used then extruded out onto a heated print bed. While this melted plastic is being extruded the head is being moved around to draw out the shape of whatever is being printed. This is currently still essentially two dimensional, what happens next is that once the first layer is finished with either the print head moves up or the print bed moves down and the next layer is started. This is how this type of printer works, after the first layer is printed onto the heated bed, the next layer is printed on top of the previous one. This process continues as many times as necessary until the layers are built up into the full object.
This process is very useful as it wastes no materials as it is an additive process as opposed to other computer aided manufacturing methods which are generally subtractive. This means that other machines cut away excess off of a block of a selected material when this printer only adds material that is needed.
The reason I built this printer is to produce one off custom components for any project I might need such a capability. Instead of trying to source the exact component I want of the right size and shape, or having to modify something else, the idea of just being able to create a three dimensional model on a computer of the object I want to create and then just be able to make it at a whim is a very attractive prospect. These parts do not only have to be plastic, there are 3D printers can can even print metal, some of these parts are used in aircraft [9] and even formula one [7] cars. These parts were made using a power type 3D printer which uses the same concept as the printers available for consumer purchase but instead of extruding melted plastic from a heated nozzle they use an extremely fine powder which is dusted over the print area and melted in the correct areas with lasers.

Figure 4.13: 3D Printed Aircraft Part - technologyreview.com

The printer I built had many printed components as parts of its own construction. These parts were made out of a blue plastic. This printer was bought in kit form from a company called Felix Printers [4] in the Netherlands and took about a day to construct and a further day to configure and get working to an acceptable quality of parts produced by it.

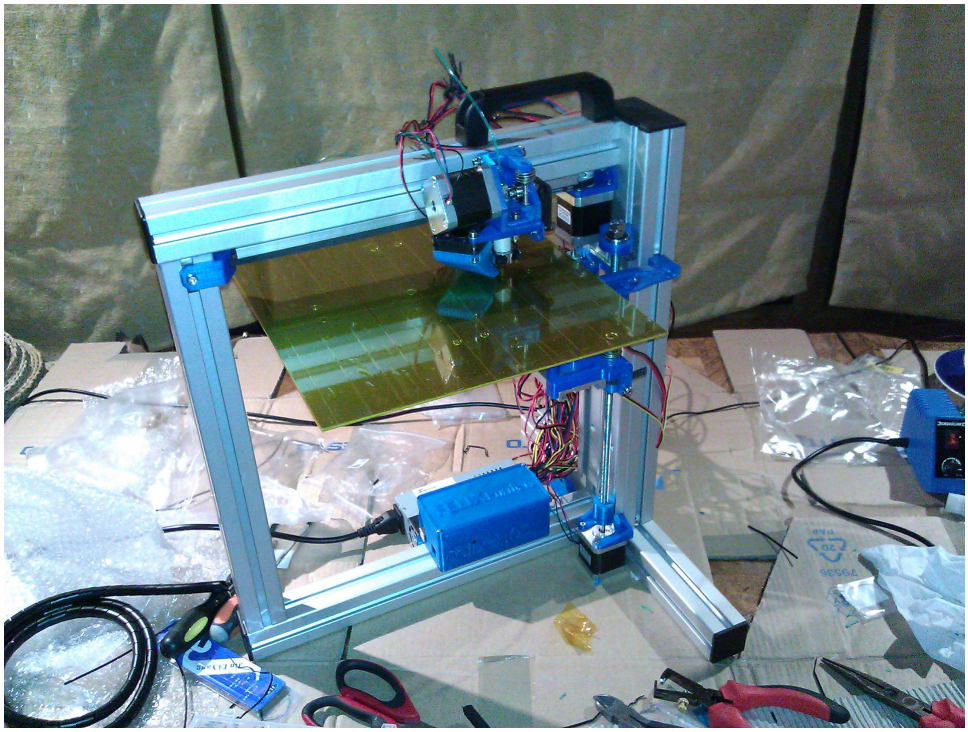

Figure 4.14: 3D Printer

The first component for the robot that I constructed with this printer is a mount for the microcontroller. The mount is use to attach the component to the chassis and to insulate it from the metal base to avoid short circuits between varies part of the component and between it and other com-

ponents that may also be mounted on the same base plate. This mounting plate was constructed by the printer in less than half an hour and weighing under 25 grams and costing me less than 50 pence.

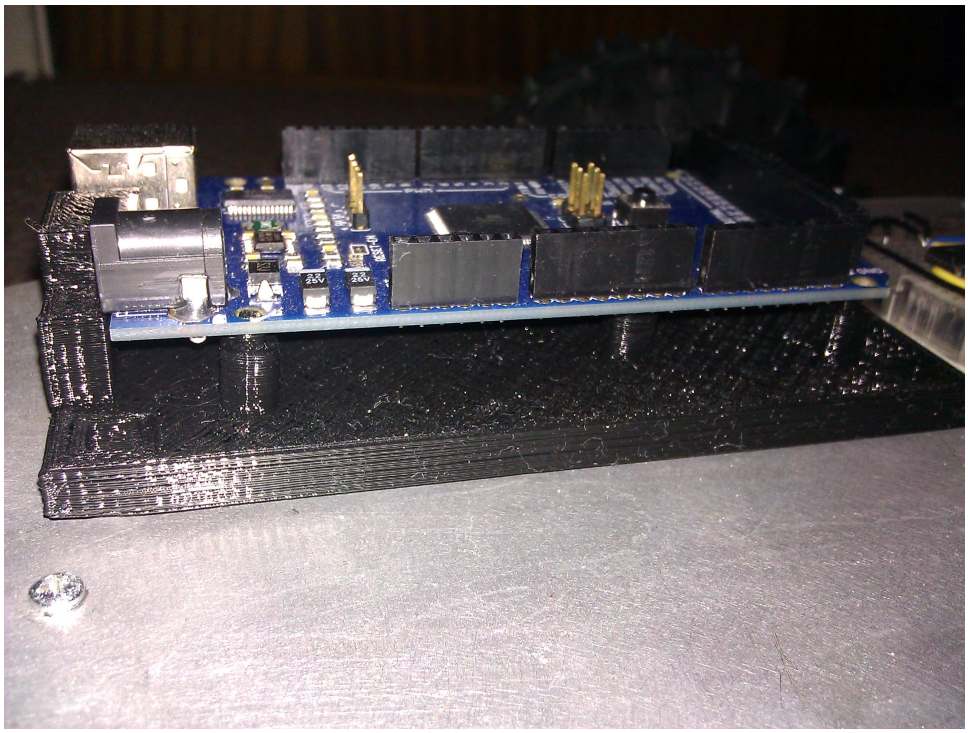

Figure 4.15: Printed Mounting Plate

Eventually the printed mounting plate will be attached to the base plate with bolts but at this stage of development it is just help in place with adhesive tape. There is a prototyping breadboard used for the h-bridge motor driver also fixed in place with the adhesive tape. With the drive wheels attached to the motors and mounted to the base plate, the rear caster wheel set in place, the Arduino microcontroller, the motor driver breadboard taped to the base plate and the motors wired up it was ready to write new motor control code as the prototype used a pair of DC motors and this iteration is using a pair of stepper motors.
To control a stepper motor there is a library to do so provided in the default set that the Arduino development environment supplies. In the code which pins of the Arduino are being used to connect to the stepper motor has to be specified and then the library enables control of speed, direction and how many steps the motor is to turn.

It would be wise to note that at this point in the project the power source is not on board the robot as it had not yet arrived from the supplier, and so I was using a desktop computer power supply connected to a wall outlet and using long leads to link the robot power rails to the 12 volt line on the power supply.

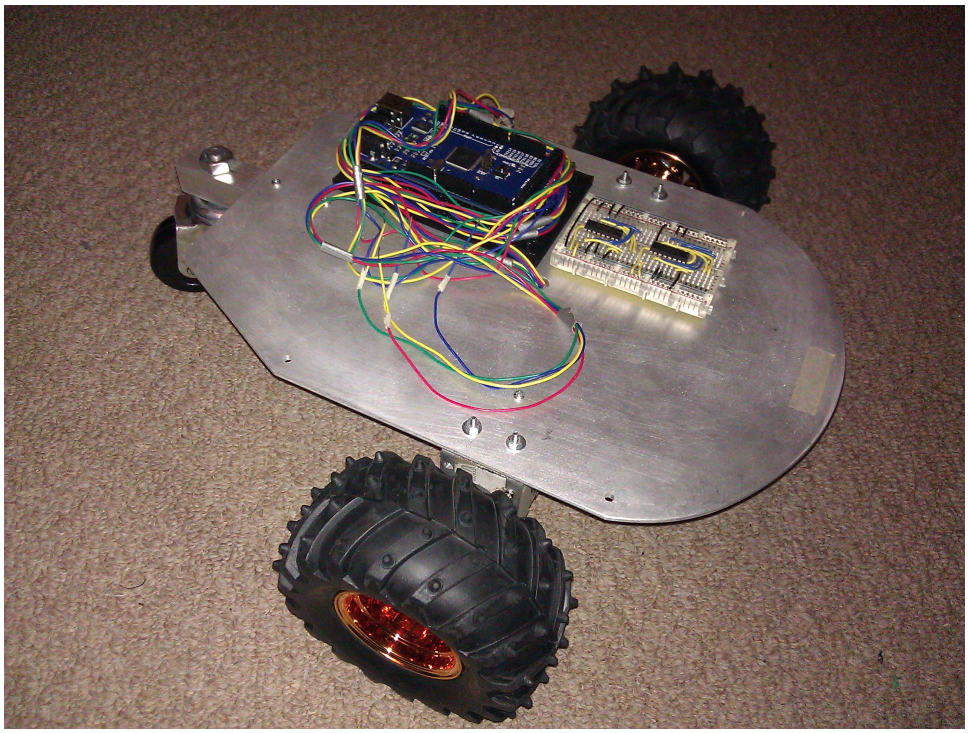

Figure 4.16: Completed MK-I

4.2.2 MK-I Evaluation

It successfully drives itself forward in a straight line. It does this very slowly and seems to struggle to do so with the motors seeming to jitter somewhat. However it does move a set distance of one meter very reliably. If the power supply has available and was fitted to the chassis I doubt the drive system would function as reliably as it is, or even move the robot at all.
The robot has a hard time turning reliably. It does turn a significant amount but at the start of each turn it vibrates. The vibrating is it struggling to shift it's weight but it does then start to turn. This will be a problem when any additional weight is added on top of the chassis and not all of the components are fitted at this point.

4.3 MK-II

The power supply had still not arrived at this point and I had planned on building on the previous version by addressing the issue of the current drive system, but without the correct components of the right size and weight there was little point in pursuing this until the parts were available. This iteration of the design will focus on the sensor system.
As the supplier I had placed my parts orders with had been having some unknown issues meeting their order commitments, at this point in the project I had still not received the bulk of the components that I had ordered. This means that I still only had a single infrared sensor available to use. There was no point in wasting time and waiting for these orders to be fulfilled and I moved straight onto using sonar as I had a pair of ultrasonic sensors.
These sensors are mounted by plugging their pins into a small prototyping breadboard each and connected to the microcontroller with jumper wires. Instead of having the sensors facing directly

forwards and due to the fact that only two were available I mounted them at an angle so each sensor can detect objects in a different region in front of the robot. Unfortunately this means that anything small enough that is directly in front of the robot will not be detected by the sensors.

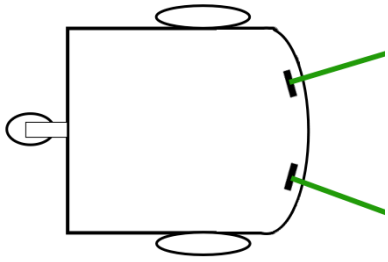

Figure 4.17: Sonar Configuration

This configuration would enable the robot to turn right or left depending on which sensor detected an object close to it. Now that the robot could collect sensor data it was time to build the feedback interface that was previously mentioned so that I could check what the robot thinks its sensors are telling it without having to follow the robot around with a laptop and a long cable.

4.3.1 Feedback Interface

The fact that I wanted this to be small, compact and portable meant that using a breadboard to link the components together would not conform to that criteria and as such I connected the components directly together with solder and wires. An Arduino Fio connected to the back of a 16 x 2 character liquid crystal display, with an xbee wireless module fitted into the modules slot on the Arduino unit as well as four buttons and a power switch glued to the base of the display for user interaction. A small lithium polymer battery if help in place between the wireless module and the socket on the Arduino. And the final component to make it all work is a small voltage step up unit to convert the Arduino's 3.3 volt output up to the 5 volts that the display needs to operate correctly.

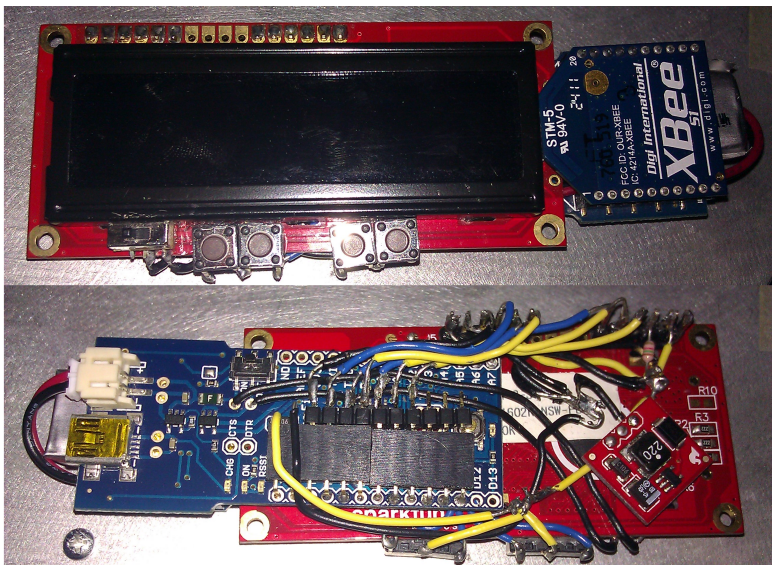

Figure 4.18: Feedback Interface

This device just receives information sent from the robot about the values the sonar sensors are providing. Each line of the display shows what distance each sensor thinks an object is away from itself. I had written a small menu system on the device so that the user can check how long it has been running for, what the current sensor readings are and to even be able to manually control the drive system of the robot. With the purpose of the device being that or portability I decided it needed to not be difficult to hold, as it was at this point because of the electronics being exposed. It was then sewn into a glove so that it could be used hands free. Having it inside a glove also protected the electronics from being interfered with.

Figure 4.19: Wrist Device

4.3.2 MK-II Evaluation

At this point the robot does move, but very slowly and with some difficulty, this is an issue that still needed solving. The robot can also detect if an object is in front of either sonar sensor.
The next step to improve on this iteration would be to enable the robot to turn to avoid an object that it has detected.

4.4 MK-III

This iteration was a short one due to the fact that the ordered parts had still not arrived from the supplier there was not much that could be done. The only changed made to this version was in the software. These changes were adding code that enabled the robot to change direction when approaching an obstacle. First was to decide on a threshold value for when to trigger the robots decision to turn. This was accomplished via trial and error, where I would set a default value and see if the robot actually avoided the obstruction. This default value was the distance from the center of the robot to the outer edge of the wheel, the logic being that this is the minimum amount of space the robot would need to be able to turn in.
This turned out to not be enough due to how the robot struggles to move normally and when turning it has a harder time moving when turning. The robot does not turn enough and drives at an angle into the obstacle it is trying to avoid.

I just increased the threshold to make the robot try to turn when it is much further away from the detected obstacle.

4.4.1 MK-III Evaluation

The MK-III iteration has highlighted the fact that the project could not move much further forwards until the rest of the ordered parts have arrived.
There is still the issue with the drive system not being strong enough to move the weight of the entire robot.
The other issue with the current iteration is that the sensor threshold had been calibrated to accommodate for the lack of power in the drive system and as such it tried to avoid objects too far away. This can make the robot miss gaps that are big enough to easily fit through.

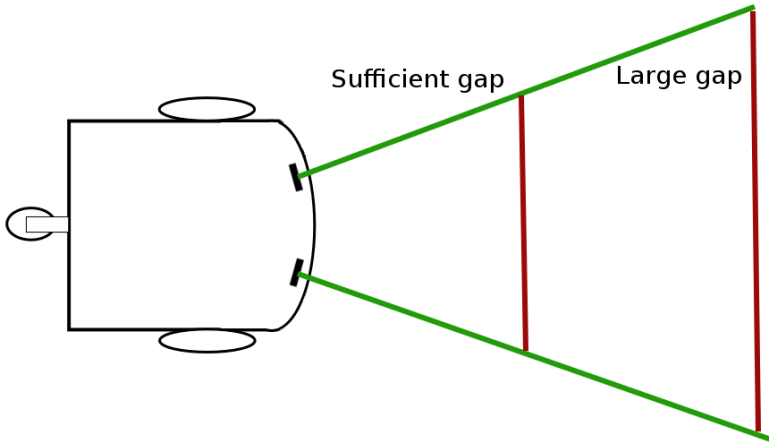

Figure 4.20: Sonar Range Gap

When the drive system is strong enough to make tighter turns the sonar range threshold can be re-calibrated to a much shorter distance and allow the robot to move inside much tighter spaces.

4.5 MK-IV

The supplier had fulfilled their orders by this time and I now had the parts that I needed. After mounting the power supply onto the chassis in the form of a lead acid battery and some adhesive tape, I tested if the current drive system could still move with the extra weight.
As it struggled to move when the power source was not on-board it was unlikely to cope any better when a heavy lead acid battery is added to the payload. The robot no longer moves under it's own power apart from vibrating violently as the motors try to turn the wheels.
The stepper motors I previously bought for the project did not have enough torque to move the weight of the full payload. When this first came apparent I ordered a pair of stepper motors with a much higher holding torque rating of 125 oz.inch.
These new motors are much bigger than the old ones which required me to make new motor mounts. I made these mounts out of the excess aluminium strips I had left after making the first pair of motor mounts and designed them to use the same mounting holes as the first pair of mounts.

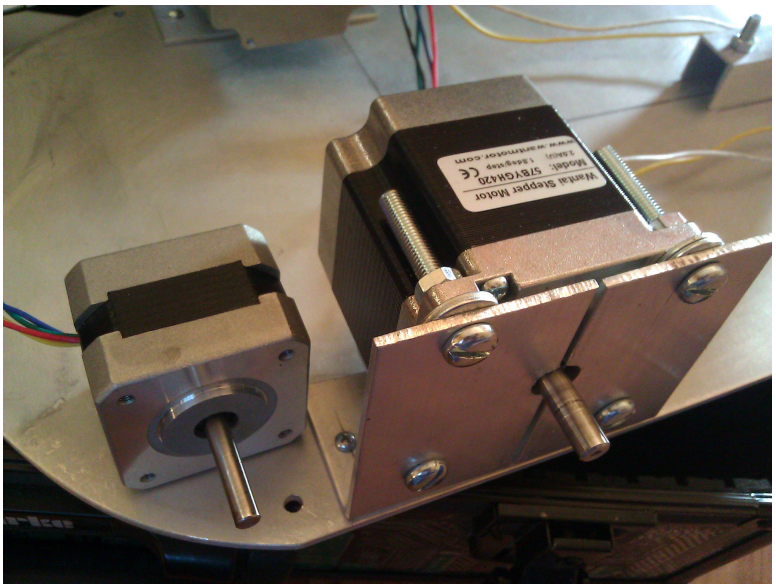

Figure 4.21: New Motor Next to Old Motor

These new motors can move the robot much much greater ease than the previous ones, even when trying to turn. It can now turn in a much smaller space resulting in me being able to reduce the threshold on the sonar sensor setting.
Another change made in this iteration was removing the code that used the stepper motor library and instead I wrote my own control code which just sends signal pulses to each stepper motor wire manually. All this involves is powering each coil in order to turn the motor. The advantage to performing this manually instead of using the library is that the library does not support braking. If all of the coils are powered at the same time the magnets actively hold the motor shaft in place, this can help when starting to move as the first part of the motion is already done by holding the shaft in position instead of having to pull it into position at the start of the motion cycle of code.

4.5.1 MK-IV Evaluation

The MK-IV now moves with greater easy and can now turn faster so that it does not drive straight into walls. It is now configured to avoid objects when it is closer to them.
The new issue that has turned up with this iteration of the design is again with the drive system. This time the h-bridge motor drivers overheat, this is due to the additional load that the new bigger motors draw from the system. One of them actually burnt out even with a heat sink attached.

4.6 MK-V

The MK-V was focussed on resolving the latest drive system issue. I tried adding some current limiting resistors which did reduce the temperate by a small amount but the motor drivers still ran too hot.
I had then decided to implement a pre-built motor driver into the system, one which is not too expensive. I found a suitable unit with standard spacing holes in the board which pins can be soldered through in order to fit into the breadboard that is already attached to the robot chassis baseplate for the previous motor driver. This unit is a Pololu motor driver that operates between 8 and 35 volts which is well within the voltage range of the robot's system. The new motor driver

can also supply 1.2 amps of current to each coil of the motor without the need for cooling, but with proper cooling it can supply up to 2 amps of current which is much more than the h-bridge drivers that had a maximum supply of 1 amp and overheated before that.

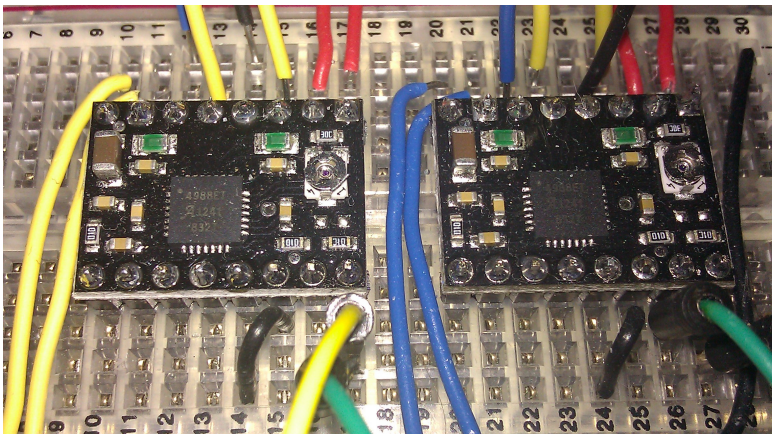

Figure 4.22: Replacement Motor Driver

With the old driver system I had to manually control the signals to each of the connections of each stepper motor, the new drivers only require a direction signal and a step pulse per motor. The motor turns a step each time the step pin on the motor driver receives a signal pulse. This system is less complex to control but has the downside of not being able to brake. By this I mean that with the previous system I was able to supply current manually to all of the motor connection coils which would hold the motor in it's current position, but with the new drivers it manages the current to the motor coils itself and because of this I cannot specify if I want the motors holding their position rather than just not turning.

4.6.1 MK-V Evaluation

The MK-V robot now drives with greater reliability than previous iterations due to the combination of the bigger stepper motors and the new motor drive boards.
The downsides to this iteration is that it can no longer actively brake and it still only uses two sonar sensors to detect obstacles, this means that it still suffers from the cons of using sonar to detect range.

4.7 MK-VI

The MK-VI iteration focused on adding infrared sensors to accompany the sonar sensors. As previously mentioned in the design section, it was always intended that sonar and infrared would be used together to decrease the number of false readings produced by the sensor system. The infrared sensors suffer from ambient infrared radiation giving them false readings. The sonar sensors suffer from echo where the sounds bounces more than once before coming back to the receiver and thus gives a distance reading indicating the object is at a greater distance from the sensor that it actually is in reality. As the two different types of sensors suffer from different issues the objective is that if one sensor gives a reading that is too far different than the other sensor it is paired with then the robot will discard that reading and request a new set of readings from the sensor pair before making a decision if it should move or not.

The sensors were just placed on the front of the robot chassis to begin with, this is for testing purposes. Sonar sensors are already plugged into small breadboards and the infrared sensors are placed underneath the sonars against the side of the small breadboards.

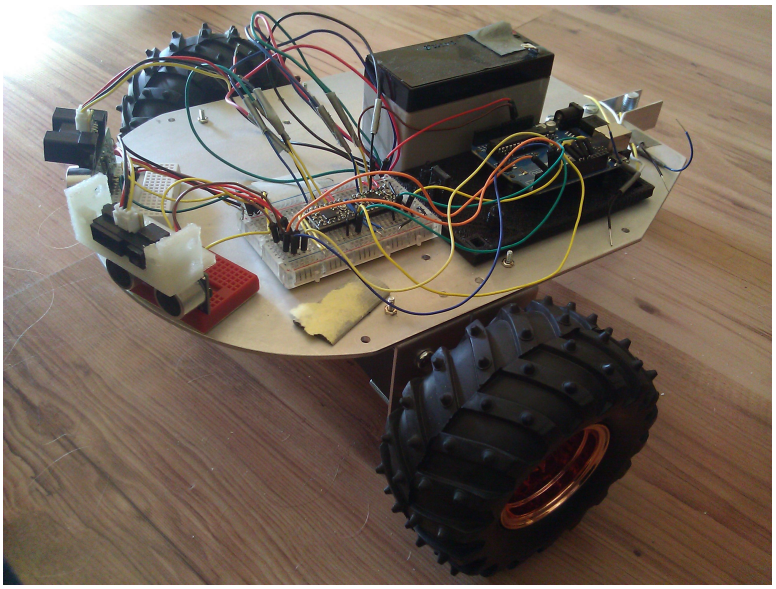

Figure 4.23: MK-VI Robot

After placing the sensors on the robot I then had to write the code that gathers the sensor data, compares each pair to check if there are any major differences in their readings. If there are any differences above a threshold value then discard the readings and take a new set. Once an acceptable set of readings have been taken the robot will use the already existing decision code to determine which direction for the drive system to move the robot in. To test this new code I put the robot up on blocks so that the wheels did not move the robot, then I put an object at different distances from each sensor in each pair. This worked very well as the system kept requesting new sensor readings until I moved the objects to roughly the same distance away.
The 3D printer was used again to make the sensor mounts which are then attached to the robot chassis and house a sensor in each.

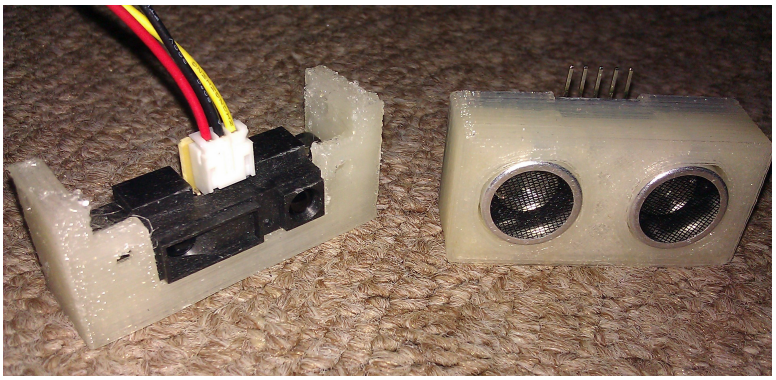

Figure 4.24: Printed Sensor Mounts

Chapter 5

Testing

5.1 Overall Approach to Testing

Testing any complex hardware of software system is a critical process because if there are any major points of failure the whole system could cease to operate. Time does not always permit this but at least some general testing of all systems should be carried out to highlight any obvious problems.

5.2 Automated Testing

Automated testing is a very good process to be using as the system being developed is tested and the results of these tests whether they be positive or if they highlight problems with the system all without the developer having to do anything after the initial setup of this testing process.
Due to the embedded nature of this project, automated testing is very limited. As there is no simulator for an Arduino microcontroller and all the various components that can be used in conjunction with one, no automated testing of this can be achieved without a far more complex hardware system which somehow can test other hardware configurations. The only part of the system that can be automatically tested with any degree of reliability is to parse the software code and check is it is syntactically valid.

5.3 System Tests

These are written to test each individual part of the project. This starts with testing if the code validates. A compiler is used to validate this and a compiler is another program that checks if code will actually work, not necessarily for its intended purpose but just check that it does not have a major flaw that will stop the finished program from running at all and converts it into either a binary capable of being run as an executable program or into another language. This compiled code is what will run on the hardware itself rather than through an interpreter which is another program that interprets what the code is trying to do and runs that.
The next stage of testing is to check that the software for each individual hardware interaction works as expected. This includes checking that when the code tells a motor to turn in a specific direction that it actually turns as it is told to. A motor is laid out on a workbench, hooked up to the microcontroller and told to turn. If it does as expected then the test will be considered as passed, otherwise it will be considered as failed and debugging will be required before running the tests again.

Similar tests will be run on the sensors with then placed at set distance intervals from objects and tested to see if the recorded distance measured by the sensor is the same as the real world measurement specifies.

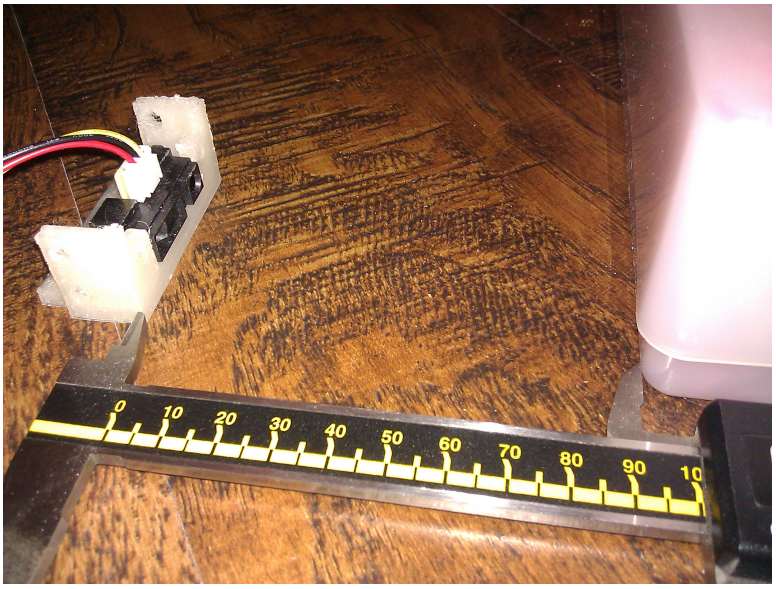

Figure 5.1: Infrared Distance Test

5.4 User Interface Testing

The only user interface on the robot is a button to apply or cut off power to the system and an activity readout over a serial communication line. This communication can either be over a USB (universal serial bus) cable or over some form of wireless serial such as bluetooth or the xbee module I am using.
The only tests I ran on this were turning the robot on and off and see if it powered on and off, also tested the wireless feedback interface by running the robot and reading the display to see if it displayed the sensor readings that the robot was receiving from the sensor pairs.

All of the tests performed on the robot were manual in nature and required a physical setup of the environment, this made repeating the conditions of any errors a difficult task as the environment will have to be exactly the same as when the error happened.

Chapter 6

Evaluation

The final product met what was outlined in the abstract. This being that a robot was built that could move around an environment under it's own power, detect and avoid approaching obstacles.

It does not perform this task as reliably as it was expected to from the initial description, the main issue being with the drive system. I had under estimated how much torque the motors would require to move the total weight of the finished robot, this was not identified correctly in the initial design. This issue was addressed later on in the project implementation but was not fully corrected, the robot does move but it only takes a moderate incline like a ramp to make the robot struggle to move again. Choosing stepper motors alone was a bad design decision and in hindsight I should have used a gear system to gain the extra torque that it needed to handle movement up an incline.

The sensor pair worked well to validate each other in the event of interference or an unfavourable environment. With the robot only being able to see in two very narrow straight lines out in front of it, I would have liked to add more sensor pairs to get a wider view of the environment. A wider view would allow for better calculating which direction had the least obstacles in and to move accordingly.
Extra sensor pairs would be arranged in a circle around the robot to form a halo. This would still only give narrow points of view of the environment. I could possibly mount a sensor pair on a servo controller pan and tilt mount, this could then point the sensors at any angle from the robot and take readings. This would work but it would also be very slow and impractical. A camera would also be a good solution I would consider for generating a more accurate map of the robot's surrounding environment to better work out where to go. The camera solution will require far more processing than the current sensor solution but has a much higher resolution.

I think that the tools I used to complete this project were appropriate, these being the Arduino development environment, version control system used to develop the software and the various hardware tools used to create the physical robot. A different version control system could have been used to maintain the code such as subversion or mercurial but this project did not actually require any complex features of these systems apart from merging. All of the merges that took place were simple and very easy to do, but that may have been due to how good Git is at performing this function.
As stated previously the project aims were met but not to as high of a standard as I had hoped for. The robot has to stop moving to take sensor readings as the microcontroller is not capable of multitasking, this is an issue as it puts more strain on the motors with this stop and start behaviour.

If I were to start this project again from scratch I would seperate the main tasks such as motor control, sensor readings and decision making each to their own processor. By this I mean that each subsystem would have it's own microcontroller and a central unit would tell them what to do. This would mean that the control unit can tell the robot to move forwards unti told otherwise and the sensor subsystem could take all the readings it wants without disrupting the movements of the robot. I would also use a sensor system with higher resolution of the environment and which is more reliable such as a camera or a pair of cameras to be able to detect depth. A laser scanner would also be a good sensor to have but they are very expensive and require even more processing than the camera.
The robot is very heavy due to the amount of aluminium used in the construction of the chassis and the mounts, this could be greatly reduced by using the 3D printer to fabricate a chassis. This chassis would be different from one made out of cut sheets of plastic because it can be made hollow with an internal structure such as a honeycomb weave to keep the strength near to that of a solid sheet. This method would reduce the cost and be made to whichever specifications are required.

Overall the project met it's basic aim but has much room for improvment and expansion by building upon all of the mentioned system alterations.

Appendices

Appendix A

Third-Party Code and Libraries

As has been said in lectures, it is acceptable and likely that you will make use of third-party code and software libraries. The key requirement is that we understand what is your original work and what work is based on that of other people.

Therefore, you need to clearly state what you have used and where the original material can be found. Also, if you have made any changes to the original versions, you must explain what you have changed.

1.0.1 Arduino

The main piece of third party software that I used is the Arduino integrated development environment. This is a software package used to write programs, compile them and deploy them onto the Arduino microcontroller series.
This software includes a collection of libraries to be used with the Arduino hardware to aid in the development of programs for it. The only one of these libraries I use is for serial communication of the sensor data base to a human readable terminal.

1.0.2 Ino

This is a command line toolkit for working with the Arduino hardware. This software is distributed via Github and is released under the MIT licence.
This software is incomplete and required a small amount of scripting to get it working with my project. It was only used as a command line way of validating my code when I was working on the project without full access to the computer I was using and as such used a virtual private server I rent which is hosted at a remote location. With this server I used to work on being in a remote location there was no screen to access, this made using the official Arduino software difficult to use and thus using this command line alternative was useful in this regard.
Ino does require python 2.6+ and the Arduino software to be installed as well but does not require the Arduino IDE to be launched.

Appendix B

Code samples

2.1 Motor control

Motors moving forward using stepper library and h-bridge.

```
//full rotation divided by degrees turned per step
const int fullTurn = 360/1.8;
//Make a motor object using the stepper library
//on the correct pins
Stepper motorL(fullTurn, 40,41,42,43);
Stepper motorR(fullTurn, 50,51,52,53);
void moveForward(int distance){
  for (int i = 0; i < distance; i++){
    // step one step:
    motorL.step(-1);
    motorR.step(-1);
    delay(defaultDelay);
  }
}
```

Motors moving forward using replacement Pololu stepper drivers.

```
//full rotation divided by degrees turned per step
const int fullTurn = 360/1.8;
void moveForward(int distance){
  digitalWrite(leftDirection, LOW);//fwd left motor direction
  digitalWrite(rightDirection, HIGH);//fwd right motor direction
  for (int j = 0; j < distance; j++){
    digitalWrite(leftStep, HIGH);//Step left wheel forwards
    digitalWrite(rightStep, HIGH);//Step right wheel forwards
    delay(defaultDelay);//wait
    digitalWrite(leftStep, LOW);//Stop left wheel
    digitalWrite(rightStep, LOW);//Stop right wheel
    delay(defaultDelay);//wait
  }
}
```

2.2 Sonar

Trigger and echo control of the sonar sensors.

```
/*
Example of sonar sending out a trigger pulse and
measuring how long it takes for the echo to come back
*/
int getSonarReading(int trigger, int echo){
  int duration, distance;
  digitalWrite(trigger, HIGH);
  delayMicroseconds(soundDelay);
  digitalWrite(trigger, LOW);
  duration = pulseIn(echo, HIGH);
  distance = (duration/2) / 29.1;
  return(distance);
}
```

Annotated Bibliography

[1] D. A. Cockburn, "Using Both Incremental and Iterative Development," in *STSC CrossTalk (USAF Software Technology Support Center)*. CrossTalk, 2008, pp. 27–30.

Detailed explanation and usefullness of the iterative development model

[2] B. Dynamics, "Bostom dynamics," http://www.bostondynamics.com, 2012, accessed December 2012.

Information about Boston Dynamics cutting edge experimental robots that have various different movement behaviours.

[3] R. Faludi, *Building Wireless Sensor Networks*. O'Reilly Media, 2010.

A guide to using the Zigbee mesh networking protocol, xbee uses this protocol which I use for wireless communications between the robot and a user feedback device

[4] G. Feliksdal, "Felix printers," http://www.felixprinters.com, accessed July 2012.

Source for the 3D printer used to make parts in this project.

[5] E. Gamma, R. Helm, R. Johnson, and J. Vlissides, *Design Patterns*. Addison-Wesley, 2010.

Design patterns book I used to get an idea how to structure my program code.

[6] I. Github, "Github," http://www.github.com, 2013, accessed April 2013.

Git version control host website I used to store my code repositories on and view statistic on the code that I submitted.

[7] A. Honka, "3d printing in f1," http://3dprintingindustry.com/2013/01/22/3d-printing-in-f1-3dpi-talks-to-robert-fernley-from-the-sahara-force-india-team/, Jan. 2013, accessed February 2013.

Interview with Formula One team's deputy team principal Robert Fernley talking about using 3D printing in formula one car development.

[8] A. M. LLC, "Adept mobilerobots," http://www.mobilerobots.com/ResearchRobots.aspx, 2012, accessed October 2012.

Information various models of research robots and additional modules for them.

[9] S. Nathan, "Printing parts," http://www.technologyreview.com/demo/425133/printing-parts/, Aug. 2011, accessed March 2013.

MIT review of 3D printing parts for commercial aircraft.

[10] C. Petzold, *CODE - The Hidden Language of Computer Hardware and Software*. Microsoft Press, 2000.

Interesting book of various concepts which I used to go over bit level calculations and logic.