

# **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)

1st March 2012

Version: 0.5 (Draft)

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...  
I'd like to thank...

## **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

<b>1</b>	<b>Background &amp; Objectives</b>	<b>1</b>
<b>2</b>	<b>Development Process</b>	<b>3</b>
2.1	Introduction . . . . .	3
2.2	Modifications . . . . .	3
2.3	Version Control . . . . .	4
<b>3</b>	<b>Design</b>	<b>5</b>
3.1	Overall Architecture . . . . .	5
3.2	Justifications . . . . .	6
3.2.1	Materials . . . . .	6
3.2.2	Actuators . . . . .	6
3.2.3	Drive Type . . . . .	8
3.2.4	Sensors . . . . .	11
3.2.5	Control . . . . .	13
3.2.6	Power Source . . . . .	15
3.3	Feedback Interface . . . . .	16
<b>4</b>	<b>Implementation</b>	<b>17</b>
4.1	Prototype . . . . .	17
4.2	MK-I . . . . .	20
<b>5</b>	<b>Testing</b>	<b>23</b>
5.1	Overall Approach to Testing . . . . .	23
5.2	Automated Testing . . . . .	23
5.2.1	Unit Tests . . . . .	23
5.2.2	User Interface Testing . . . . .	23
5.2.3	Stress Testing . . . . .	23
5.2.4	Other types of testing . . . . .	23
5.3	Integration Testing . . . . .	23
5.4	User Testing . . . . .	23
<b>6</b>	<b>Evaluation</b>	<b>24</b>
<b>Appendices</b>		<b>25</b>
<b>A</b>	<b>Third-Party Code and Libraries</b>	<b>26</b>
<b>B</b>	<b>Code samples</b>	<b>27</b>
2.1	Random Number Generator . . . . .	27
<b>Annotated Bibliography</b>		<b>30</b>

## LIST OF FIGURES

2.1	Branch Merge Diagram . . . . .	4
3.1	Basic system diagram . . . . .	5
3.2	Initial design . . . . .	5
3.3	Servo Motor - robotshop.com . . . . .	7
3.4	DC Motor - sparkfun.com - CC BY-NC-SA 3.0 . . . . .	7
3.5	Stepper Motor - stepperonline.com . . . . .	8
3.6	Stepper Motor Internal- robotgear.com.au . . . . .	8
3.7	Omni Wheel - sparkfun.com . . . . .	9
3.8	Tracks - robotcombat.com . . . . .	10
3.9	3D One Legged Hopper - mit.edu . . . . .	10
3.10	NAO - aldebaran-robotics.com . . . . .	11
3.11	Light Dependant Resistor - robotics.org.za . . . . .	11
3.12	Camera Module - sparkfun.com - CC BY-NC-SA 3.0 . . . . .	12
3.13	Infrared Sensor - coolcomponents.co.uk . . . . .	12
3.14	Ultrasonic Sensor - coolcomponents.co.uk . . . . .	13
3.15	PIC - circuitstoday.com . . . . .	13
3.16	Arduino arduino.cc . . . . .	14
3.17	Netduino - netduino.com . . . . .	14
3.18	Atom Motherboard - intel.co.uk . . . . .	15
3.19	Raspberry Pi - raspberrypi.org . . . . .	15
3.20	Lithium Polymer Battery - robotshop.com . . . . .	16
3.21	Lead Acid Battery - kestrelectricalsupplies.co.uk . . . . .	16
4.1	Prototype mkI . . . . .	17
4.2	Transistor - zmescience.com . . . . .	18
4.3	H-Bridge - sparkfun.com . . . . .	18
4.4	Motor Driver - sparkfun.com . . . . .	19
4.5	Prototype Code Exert . . . . .	19
4.6	Collision Illustration . . . . .	20
4.7	Aluminium Sheet . . . . .	20
4.8	Aluminium Sheet Cut . . . . .	21
4.9	Chosen Stepper Motor . . . . .	21
4.10	Motor Mounts . . . . .	22
4.11	Baseplate Underside . . . . .	22

## **LIST OF TABLES**

## 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 works as intended.

Fast forward to college five years later and I am still fascinated by electronics. Still using these wonderous 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, having 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 number of libraries but the Arduino ones seems to have a much wider variety of what they support 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 LLC (2012) 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 manaeve 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

You need to describe briefly the life cycle model that you used. Do not force your project into the waterfall model if it is better described by prototyping or some other evolutionary model. You do not need to write about all of the different process models that you are aware of. Focus on the process model that you have used. It is possible that you needed to adapt an existing process model to suit your project; clearly identify what you used and how you adapted it for your needs.

In most cases, the agreed objectives or requirements will be the result of a compromise between what would ideally have been produced and what was felt to be possible in the time available. A discussion of the process of arriving at the final list is usually appropriate.

You should briefly describe the design method you used and any support tools that you used. You should discuss your choice of implementation tools - programming language, compilers, database management system, program development environment, etc.

### 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. So, after designing the initial system, 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.

### 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 majorly 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.

There are other features that 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 independantly of the main project. You may make multiple branches at the same time and work on different things all independant 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 independantly from the trunk, even if you or somebody else makes another branch from the trunk, the changes made in your branch will not effect 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](#) 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.

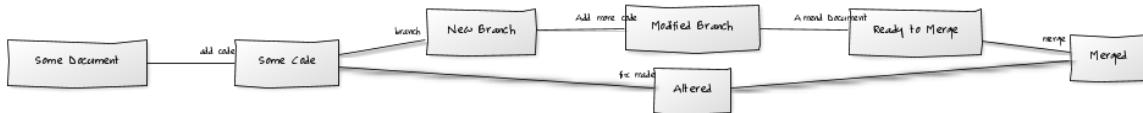


Figure 2.1: Branch Merge Diagram

# Chapter 3

## Design

### 3.1 Overall Architecture

The initial design for the robot is to produce a small wheeled vehicle with a platform for mounting the various systems. These systems should be a central control unit, motor control and the various sensors.

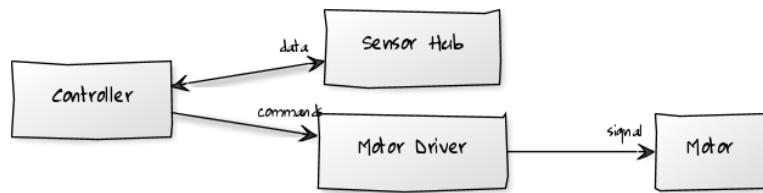


Figure 3.1: Basic system diagram

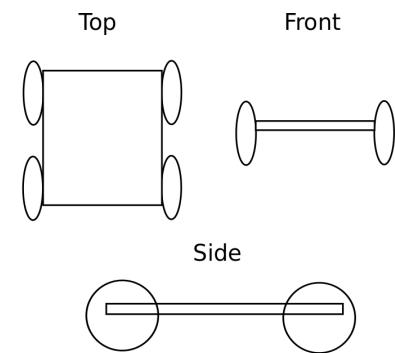


Figure 3.2: Initial design

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 Justifications

The various components that the project will need to come together into a finished product have many options.

### 3.2.1 Materials

I considered several materials for the robot chassis to be built of.

- 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

- Steel

A stronger material that can withstand a much heavier load, but is itself rather heavy compared to wood or plastic. This extra base weight before adding anything else will put more strain onto the motors 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 is 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.

- 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, they are typically not good for driving larger equipment. Also 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](http://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 device's 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. 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.



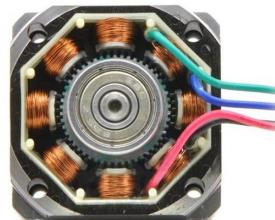
Figure 3.4: DC Motor - [sparkfun.com](http://sparkfun.com) - CC BY-NC-SA 3.0

- **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 a known amount without any additional measuring equipment, although it may be used to verify that it has in fact moved the amount expected



Figure 3.5: Stepper Motor - [stepperonline.com](http://stepperonline.com)



[www.pololu.com](http://www.pololu.com)

Figure 3.6: Stepper Motor Internal- [robotgear.com.au](http://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.

- **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 used in many configurations.

- **Two Wheels**

This is an unstable configuration which requires it to be in constant motion to maintain

an upright position. Due to the center of gravity being directly in the middle underneath the axle a weight is often positioned here to help keep it upright as well as a tilt sensor to help compensate. 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 is 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 risking 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 them to traverse in a lateral direction if the front and back pair are rotated in the opposite direction of each other.



Figure 3.7: Omni Wheel - sparkfun.com

Car type configuration is also four wheels but only either the front or the back pair are powered and either the front or 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 and having to lean and 'hop' to move around.

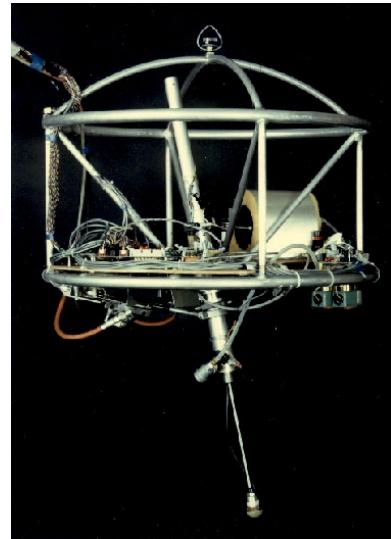


Figure 3.9: 3D One Legged Hopper - mit.edu

- **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 joints are added the complexity significantly increases.

- **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 no get stripped under the weight of whatever payload the robot



Figure 3.10: NAO - [aldebaran-robotics.com](http://aldebaran-robotics.com)

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.

### 3.2.4 Sensors

- **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, kind of like a physical bump skirt which activates when something touches it.

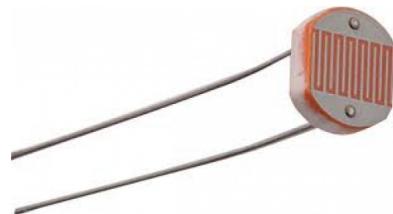


Figure 3.11: Light Dependant Resistor - [robotics.org.za](http://robotics.org.za)

- **Camera**

A camera could be used to detect objects in front of it using various image processing techniques. This method is good because it can potentially map a relatively large area in a single image. On the other hand it requires more processing to do, 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. I could use a more powerful processor to overcome this but it adds complexity, cost and power consumption.



Figure 3.12: Camera Module - [sparkfun.com](http://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 un-needed and unwanted and as such if it arrives at the receiver the readings will be inaccurate from those expected.



Figure 3.13: Infrared Sensor - [coolcomponents.co.uk](http://coolcomponents.co.uk)

- **Sonar**

Again 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.

A combination of both sonar and infra red 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



Figure 3.14: Ultrasonic Sensor - [coolcomponents.co.uk](http://coolcomponents.co.uk)

be used to compensate for sonar bouncing around the environment. Hopefully this will reduce the number of false readings produced.

### 3.2.5 Control

The robot will need a controller, that connects the software to all the hardware.

- **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.

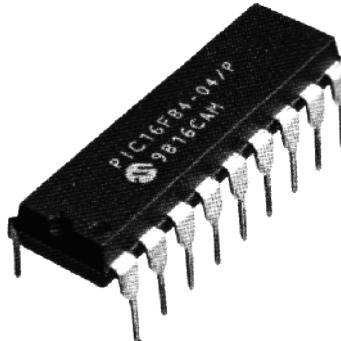


Figure 3.15: PIC - [circuitstoday.com](http://circuitstoday.com)

- **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.

- **Netduino**

This is also an open source electronics prototyping platform but instead of being based on

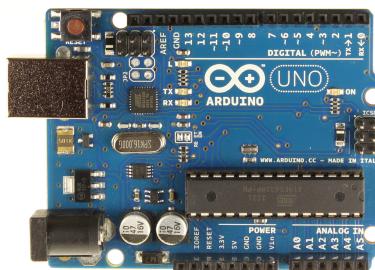


Figure 3.16: Arduino arduino.cc

C and C++ it is based on the .Net Micro Framework which is Microsoft's version of an embedded framework.

These boards 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 compared to the consumption of a microcontroller. 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.

The cost of such boards is also very high as they are far more complex pieces of electronics.

- **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. 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 3.18: Atom Motherboard - [intel.co.uk](http://intel.co.uk)



Figure 3.19: Raspberry Pi - [raspberrypi.org](http://raspberrypi.org)

### 3.2.6 Power Source

As this robot is intended to move around freely, 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. The battery will have to be several cells or a single high output cell due to the size of motors intended.

I will need as many Amp hours as possible for longer runtimes. 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).

- **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.

- **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 needed a high power motor for the chassis already, a higher weight battery is not too much of an issue and provides the benefit of the higher power output.



Figure 3.20: Lithium Polymer Battery - [robotshop.com](http://robotshop.com)



Figure 3.21: Lead Acid Battery - [kestrelectricalsupplies.co.uk](http://kestrelectricalsupplies.co.uk)

### 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. There cannot be any cables trailing from the robot to a laptop so a wireless solution would be good.

All that is needed is a small microcontroller, a wireless module and a display. 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.

A small LCD (liquid crystal display) can be connected to the Arduino to display information it receives.

# Chapter 4

# Implementation

## 4.1 Prototype

It is a good idea to first build a basic 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 conform 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 get my hands on 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 an afternoon.

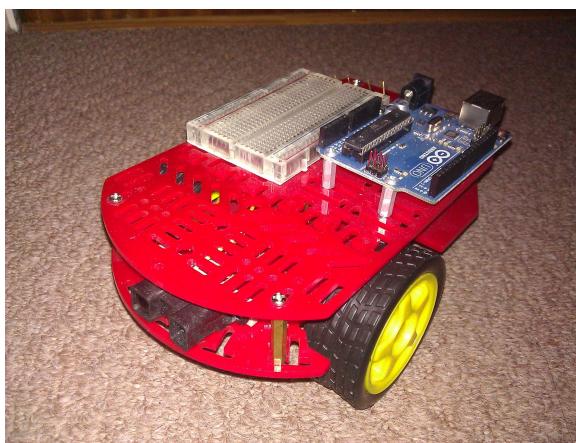


Figure 4.1: Prototype mkI

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 of the system and just read the value coming back on the signal pin. The difference between zero up to the amount of power being given to the sensor, in this case the datasheet specified 5 volts and that is what I supplied it 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. 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 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.

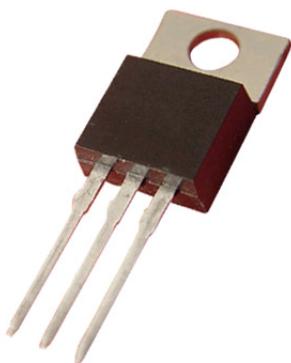


Figure 4.2: Transistor - zmescience.com

This could be solved using a transistor (a semi-conductor 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. The h-bridge chip also has its issues, as it generates heat

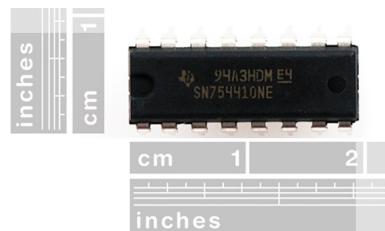


Figure 4.3: H-Bridge - sparkfun.com

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 of 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. I decided to build a simple motor

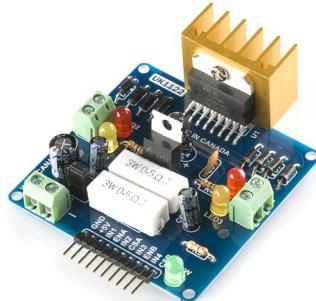


Figure 4.4: Motor Driver - sparkfun.com

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: This seems to work quite well, it does drive forwards and it does turn

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

Figure 4.5: Prototype Code Excerpt

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. 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.

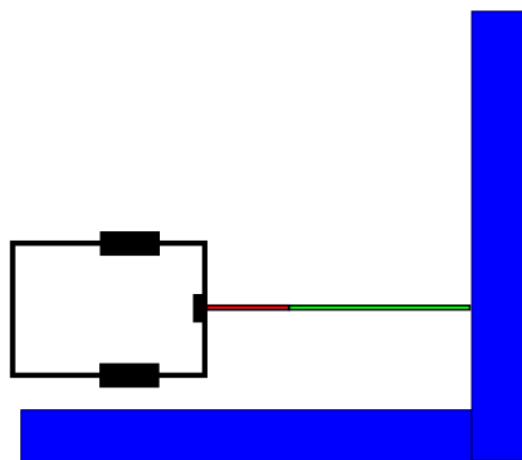


Figure 4.6: Collision Illustration

## 4.2 MK-I

The next step is to build a version based on the design and using the things I have learnt from the prototype to improve it.

First thing to do is to build a sturdy chassis for all the systems to be mounted onto. I bought several sheets of aluminium to be cut into a similar shape as the small red plastic prototype. I chose this



Figure 4.7: Aluminium Sheet

shape mainly because of how much I liked working with said prototype but also because that a curved front seemed logical for mounting an array of sensors to.

Next is 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.

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



Figure 4.8: Aluminium Sheet Cut



Figure 4.9: Chosen Stepper Motor

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.

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 4.10: Motor Mounts



Figure 4.11: Baseplate Underside

# **Chapter 5**

## **Testing**

Detailed descriptions of every test case are definitely not what is required here. What is important is to show that you adopted a sensible strategy that was, in principle, capable of testing the system adequately even if you did not have the time to test the system fully.

Have you tested your system on 'real users'? For example, if your system is supposed to solve a problem for a business, then it would be appropriate to present your approach to involve the users in the testing process and to record the results that you obtained. Depending on the level of detail, it is likely that you would put any detailed results in an appendix.

### **5.1 Overall Approach to Testing**

### **5.2 Automated Testing**

#### **5.2.1 Unit Tests**

#### **5.2.2 User Interface Testing**

#### **5.2.3 Stress Testing**

#### **5.2.4 Other types of testing**

### **5.3 Integration Testing**

### **5.4 User Testing**

# Chapter 6

## Evaluation

Examiners expect to find in your dissertation a section addressing such questions as:

- Were the requirements correctly identified?
- Were the design decisions correct?
- Could a more suitable set of tools have been chosen?
- How well did the software meet the needs of those who were expecting to use it?
- How well were any other project aims achieved?
- If you were starting again, what would you do differently?

Such material is regarded as an important part of the dissertation; it should demonstrate that you are capable not only of carrying out a piece of work but also of thinking critically about how you did it and how you might have done it better. This is seen as an important part of an honours degree.

There will be good things and room for improvement with any project. As you write this section, identify and discuss the parts of the work that went well and also consider ways in which the work could be improved.

The critical evaluation can sometimes be the weakest aspect of most project dissertations. We will discuss this in a future lecture and there are some additional points raised on the project website.

# **Appendices**

## **Appendix A**

# **Third-Party Code and Libraries**

If you have made use of any third party code or software libraries, i.e. any code that you have not designed and written yourself, then you must include this appendix.

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.

## Appendix B

# Code samples

### 2.1 Random Number Generator

The Bayes Durham Shuffle ensures that the psuedo random numbers used in the simulation are further shuffled, ensuring minimal correlation between subsequent random outputs Press *et al.* (1992).

```
#define IM1 2147483563
#define IM2 2147483399
#define AM (1.0/IM1)
#define IMM1 (IM1-1)
#define IA1 40014
#define IA2 40692
#define IQ1 53668
#define IQ2 52774
#define IR1 12211
#define IR2 3791
#define NTAB 32
#define NDIV (1+IMM1/NTAB)
#define EPS 1.2e-7
#define RNMX (1.0 - EPS)

double ran2(long *idum)
{
    /*-----
     * Minimum Standard Random Number Generator      */
     * Taken from Numerical recipies in C          */
     * Based on Park and Miller with Bays Durham Shuffle */
     * Coupled Schrage methods for extra periodicity   */
     * Always call with negative number to initialise   */
     *-----*/
    int j;
    long k;
    static long idum2=123456789;
    static long iy=0;
```

```

static long iv[NTAB];
double temp;

if (*idum <=0)
{
    if (-(*idum) < 1)
    {
        *idum = 1;
    }else
    {
        *idum = -(*idum);
    }
    idum2=(*idum);
    for (j=NTAB+7;j>=0;j--)
    {
        k = (*idum)/IQ1;
        *idum = IA1 *(*idum-k*IQ1) - IR1*k;
        if (*idum < 0)
        {
            *idum += IM1;
        }
        if (j < NTAB)
        {
            iv[j] = *idum;
        }
    }
    iy = iv[0];
}
k = (*idum)/IQ1;
*idum = IA1*(*idum-k*IQ1) - IR1*k;
if (*idum < 0)
{
    *idum += IM1;
}
k = (idum2)/IQ2;
idum2 = IA2*(idum2-k*IQ2) - IR2*k;
if (idum2 < 0)
{
    idum2 += IM2;
}
j = iy/NDIV;
iy=iv[j] - idum2;
iv[j] = *idum;
if (iy < 1)
{
    iy += IMM1;
}
if ((temp=AM*iy) > RNMX)

```

```
{  
    return RNMX;  
} else  
{  
    return temp;  
}  
}
```

# Annotated Bibliography

Cockburn, Dr. Alistair. 2008. Using Both Incremental and Iterative Development. *Pages 27–30 of: Stsc crosstalk (usaf software technology support center)*. CrossTalk.

Detailed explanation and usefulness of the iterative development model

Dee, H. M., & Hogg, D. C. 2009. Navigational strategies in behaviour modelling. *Artificial intelligence*, **173**(2), 329–342.

This is my annotation. I should add in a description here.

Duckworth, Sylvia. 2007. *A picture of a kitten at Hellifield Peel*. <http://www.geograph.org.uk/photo/640959>. Copyright Sylvia Duckworth and licensed for reuse under a Creative Commons Attribution-Share Alike 2.0 Generic Licence. Accessed August 2011.

This is my annotation. I should add in a description here.

LLC, Adept Mobilerobots. 2012. *Adept mobilerobots*. <http://www.mobilerobots.com/ResearchRobots.aspx>. Accessed October 2012.

Information various models of research robots and additional modules for them

Press, W.H., et al. 1992. *Numerical recipes in C*. Cambridge University Press Cambridge.

This is my annotation. I can add in comments that are in **bold** and *italics and then other content*.

Various. 2011 (Aug.). *Fail blog*. <http://www.failblog.org/>. Accessed August 2011.

This is my annotation. I should add in a description here.