# The Development of an Indoor Navigation Algorithm for an Autonomous Mobile Robot

A thesis
submitted in partial fulfilment
of the requirements for the Degree
of
Master of Science in Physics
and Electronic Engineering
at the
University of Waikato

by
Lucas John Sikking


THE UNIVERSITY OF
WAIKATO
*Te Whare Wānanga o Waikato*

2004

*For Mum, Dad and Richelle*

# ABSTRACT

A Mobile Autonomous Robotic Vehicle for Indoor Navigation (MARVIN) is under development at the University of Waikato. This mechatron's navigation system uses fixed beacons situated within the environment to help eliminate the accumulated error associated with odometry. The beacons in conjunction with sensor data will help to localise MARVIN so it will be able to autonomously navigate. The map the device localises itself with is a built-in map which has been constructed from measurements taken in the first floor of the University of Waikato's C Block corridors. These measurements have then been loaded into MATLAB so the software system can interface with the map. The sensory equipment on the device includes odometry and infrared proximity sensors.

# ACKNOWLEDGEMENTS

I like to gratefully acknowledge the following people for the help and time they have given me over the course of this project.

Thanks to Dr Dale Carnegie for his invaluable advice, direction and guidance during the write-up of this thesis. I would also like to thank the technicians, Scott Forbes and Bruce Rhodes for their assistance whenever it was needed. I would also like to thank my fellow graduate students, Chris Lee Johnson, Ashil Prakash, Andrew Payne, Adam Somerville and Craig Jensen for their help, guidance and support over the last four years.

Lastly I would like to thank my Mum, Dad, my sister Richelle, my girlfriend Greer and close friends for being there when I needed support and encouragement and understanding why I was always busy during the summer.

# CONTENTS

# LIST OF FIGURES

# 1. INTRODUCTION

This thesis forms the first attempt to get a Mobile Autonomous Robotic Vehicle for Indoor Navigation (MARVIN) to operate autonomously.

This chapter provides a background on the Mechatronics Group and on MARVIN, the flagship robot of the Group. Furthermore, the other projects currently being worked on in conjunction with the navigation system in order to bring MARVIN closer to the goal of being fully autonomous will be discussed. The objectives of the project are detailed and an overview of the structure of this thesis is presented.

## 1.1. Mechatronics Group

The Group was first formed in 1992. The first projects involved the construction of micromice. These mechatrons were designed around a wheel chair chassis configuration and powered from NiCd batteries. They used an integrated analogue/digital board, making the device condensed in size and very strong. Their intended application was to autonomously solve a competition 16 × 16 grid maze as quickly as possible. The first of these mechatrons designed by Gary Brightwell went on to win the New Zealand and Australian maze competitions.

Since the success of these early projects which were based on 8 bit microcontrollers, the more recent projects have evolved into using on-board PC motherboards with real world applications [Carnegie, 2002].

### 1.1.1. Submersible Remote Operating Vehicle (ROV)

The submersible ROV [King, 2002] was kindly donated to the Mechatronics Group by Nevil Fenwick of Oceaneering in Singapore. The intended application of the ROV

is for underwater exploration. This can encompass a range of activities including searching for shipwrecks, lost treasures and victims of oceaneering accidents.

The ROV is fitted with a camera and an array of lights, enabling it to take underwater video footage. It makes use of a compass along with pressure and tilt sensors to help with localisation while underwater. The device is comprised of four motors. Two provide forward propulsion and are located at the rear, while the two remaining motors provide lateral and vertical thrust. The device makes use of an AMD K62 500 MHz processor and can be seen below in Figure 1-1 [King, 2002].

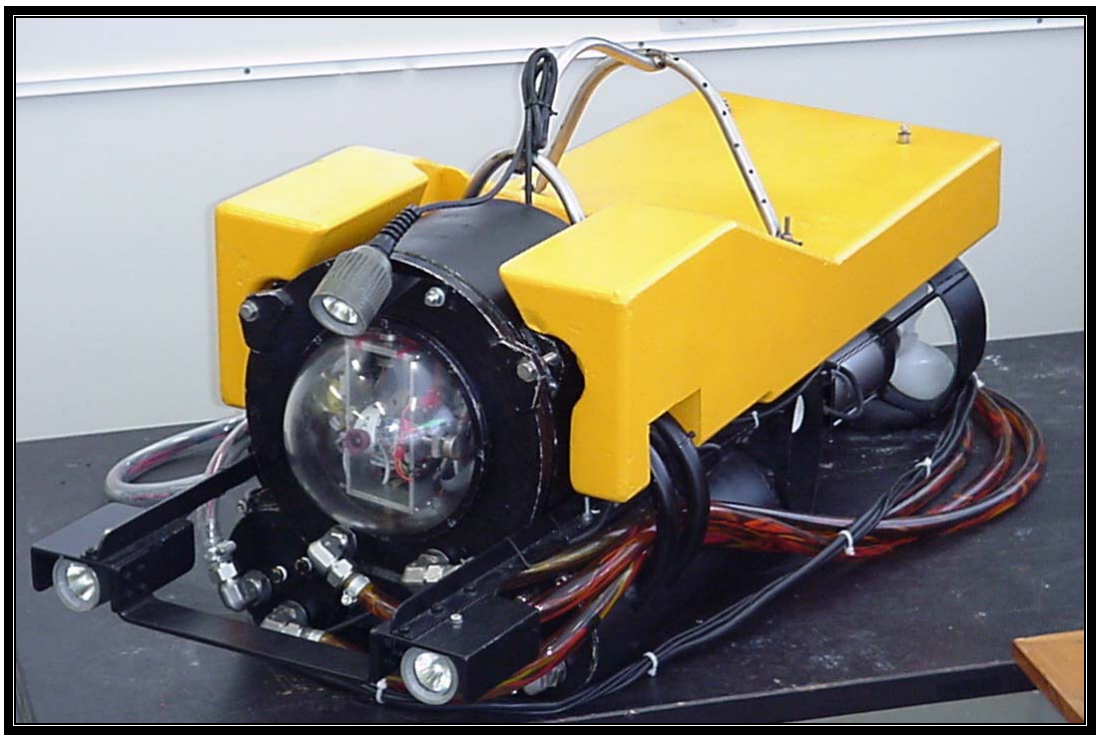


***Figure 1-1: Submersible ROV***

### 1.1.2. Multi-terrain Track Laying Robot

This device was designed and constructed by former master's student Jason Cordes. It was the first outdoor robot designed by the Mechatronics Group and makes use of a tracked drive method similar to that of military vehicles. By using a tracked drive system the device is very stable.

The applications of this mechatron are for outdoor use in irregular environments and terrains. It makes use of an AMD K6 500 MHz processor and can be seen below in Figure 1-2 [Cordes, 2002].

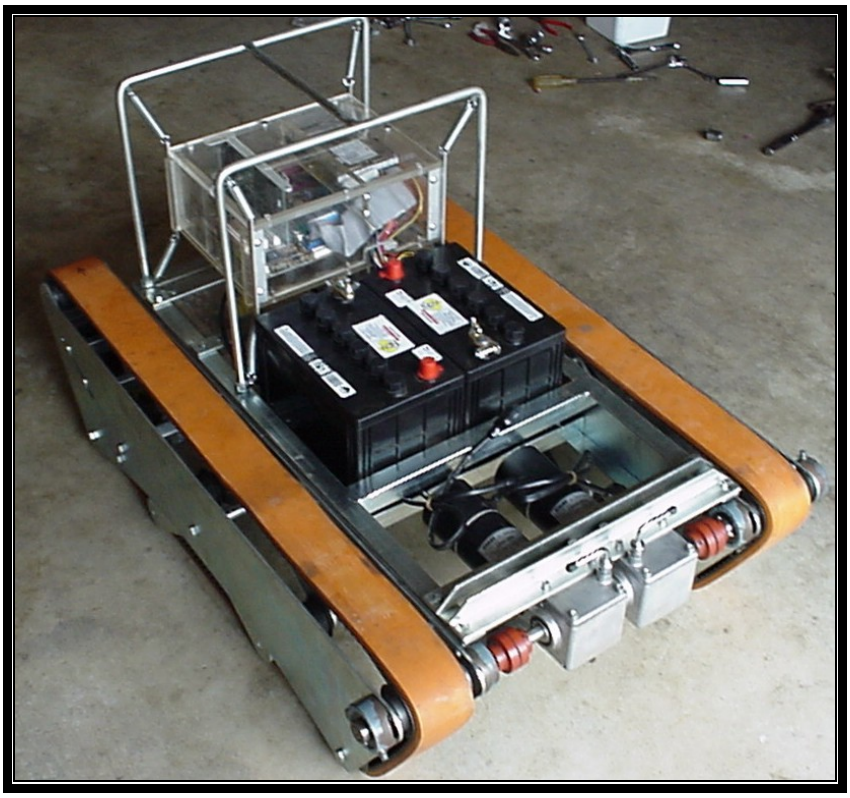


***Figure 1-2: Multi-terrain track laying robot***

### 1.1.3. Co-operative Robots

Two of these particular devices are currently under development by mechatronics student Andrew Payne. The concept behind co-operative task completion is that it can make some tasks easier to accomplish if more than one device is trying to complete it. A fleet of devices working in parallel on a task together also increases reliability. If one of these devices breaks down, there are still many others to continue working and won't lead to a stop in work or production.

The robots being designed make use of a tricycle arrangement for its simplicity and manoeuvrability. They will also be driven 'backwards' and steered from the rear, similar to that of a forklift, further increasing the manoeuvrability. Each robot is

fitted with an AMD 766 MHz processor and these devices can be seen below in Figure 1-3 [Payne & Carnegie, 2003].

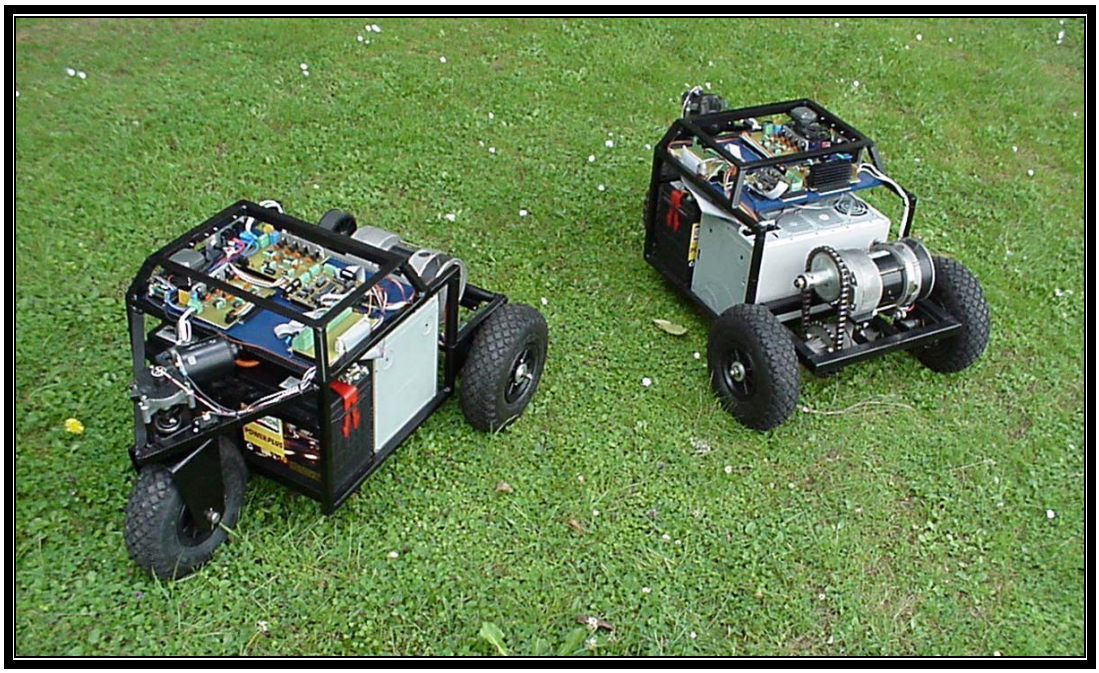


***Figure 1-3: Cooperative robots***

## 1.2. MARVIN

MARVIN is the mechatron that is the subject of this thesis. At present (without its case) it is 0.9 metres tall, with base dimensions of 0.52 × 0.575 metres and weighing 49.6 kg. The device can be seen in Figure 1-4 with its original cover and was built by previous masters student, Daniel Loughnane. The cover extends the base dimensions of the device to 0.5 × 0.8 metres and the height to 1.5 metres.


***Figure 1-4: MARVIN and the original cover***

MARVIN is powered by two motors in the wheel chair configuration which is also illustrated in Figure 1-4. This configuration involves the two drive wheels being located on the centre axis of the mechatron. Castors are used at the front and rear to provide stability. As each wheel is driven independently, MARVIN is able to turn through 360 degrees while staying in the same position by driving the wheels in opposite directions. Turning can also be implemented by driving one wheel faster than the other. This configuration also enables the wheels to support the majority of the device's weight.

The chassis of MARVIN has been designed to be very strong as it supports the frame and batteries. The base has been constructed from 3 mm thick square steel tubing of dimensions 25 mm × 25 mm, wielded together. The upper part of the chassis that is connected to the base is made from aluminium beams as it is light but still strong. These beams are connected to the base and to each other using bolts and rivets allowing them to be moved and altered easily.

The motors used to drive MARVIN's wheels were obtained from an electric wheel chair and are 24 V DC permanent magnet brush motors. The batteries used to power the motors and other circuitry on the device are Flooded Lead Acid (FLA) batteries. Two of these have been connected in series to provide 24 V. Each battery has a Cold Cranking Amperage (CCA) of 310 A and a Reserve Capacity (CA) of 55 minutes. The maximum current that can be drawn from this arrangement is 10 A with the batteries being able to be run at full power for 110 minutes before needing recharging.

The motors and batteries have been installed on the steel base of MARVIN to give the device a very low centre of gravity, further increasing stability [Loughnane, 2001].

## 1.3. Projects

At present there are three other masters' students working on projects involving MARVIN. The compilation of the projects forms the first attempt to get the device to move around its intended environment autonomously from instructions directed to it from particular people with security authorisation. The interaction of the projects can be seen below in Figure 1-5.

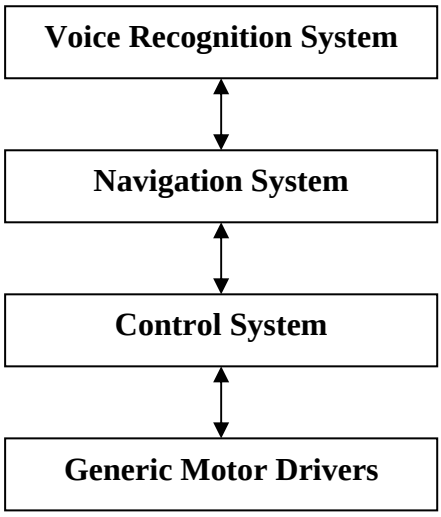



***Figure 1-5: Project interactions***

### 1.3.1. Humanisation

Ashil Prakash's project involves the humanisation of MARVIN. One of the aspects of the project is to develop a new casing to house the device. MARVIN's old casing can be seen in Figure 1-4. The purpose of the new case is to add human characteristics to the way the mechatron moves and even give the head certain facial expressions to help convey emotions and feelings. As one possible application of MARVIN is for security purposes, being able to change the device's expression to anger or that of a dominating figure will help deter any would be intruder. The intended shape of the new case can be seen in Figure 1-6.

***Figure 1-6: MARVIN's intended new casing***

A Cardax proximity security card system is also being implemented on the mechatron to help identify people that have access to the areas being patrolled by the device. This would enable MARVIN to react in a unique way to each authorised person, making the interactions with the device personal. Voice passwords and height checking will be used to give card holders protection against having their card stolen. Once a person has identified themselves, they then give voice commands to where they want to be taken and will be guided to their required destination [Prakash et al., 2003].

### 1.3.2. Navigation

This particular project is the subject of this thesis and will eventually be integrated with the voice commands provided from the above project. The commands will then be converted into co-ordinates by the navigation routine and an appropriate task plan will be calculated to carry out the indicated task.

As the task is being carried out, MARVIN will move through the environment without external help and autonomously navigate to the target point using sensor data passed back from the control system, ensuring the device stays on course, avoiding humans and walls as it navigates. The key aspect of this project is that MARVIN must know at all times where it is relative to its environment [Sikking & Carnegie, 2003].

### 1.3.3. Control

The object of this project is to carry out instructions sent to it by the navigation system. The control system is being designed by master's student Chris Lee-Johnson. It uses feedback from the sensors on MARVIN to regulate wheel speeds and provide position and orientation information.

The sensor information is interpreted and fused together with information regarding current position and orientation and then passed to the navigation system. Once passed to the navigation system it is used to help localise MARVIN within the environment. When an instruction is sent to the control system from the navigation algorithm, a velocity profile for the designated task is generated and adhered to. As the device moves, the control system interacts with LabVIEW which interfaces with the hardware, enabling the control system to control the speed of the wheels and receive the required sensor data [Lee-Johnson & Carnegie, 2003].

### 1.3.4. Generic Motor drivers

These motor drivers are being developed by master's student Craig Jensen. Their intended purpose is to be able to drive the motors on any of the mechatrons built by the Mechatronics Group and will eventually be used by MARVIN to drive the device's wheels.

The motor drivers use an H-Bridge configuration (Figure 1-7) with modified Pulse Width Modulation (PWM) switching techniques that enable back emf in the H-Bridge to be reduced. The two MOSFETs that are off in the H-Bridge when the motors are moving in a certain direction are used to help redirect stray switching currents and voltages. For example, if the mechatron is moving forward with switches S1 and S4 turned on, varying PWM signals of increasing duty cycle will be sent to the switches as the device accelerates. As S1 is turned on, due to the temporary voltage difference across the switch there is an in rush current peak from the high voltage rail. S2 can then be pulsed for an instant to allow this current to be fed back through the motor to the high side rail. The correct direction of current flow is provided from the natural inductance of the motor. These effects are at their most prolific during braking and accelerating.

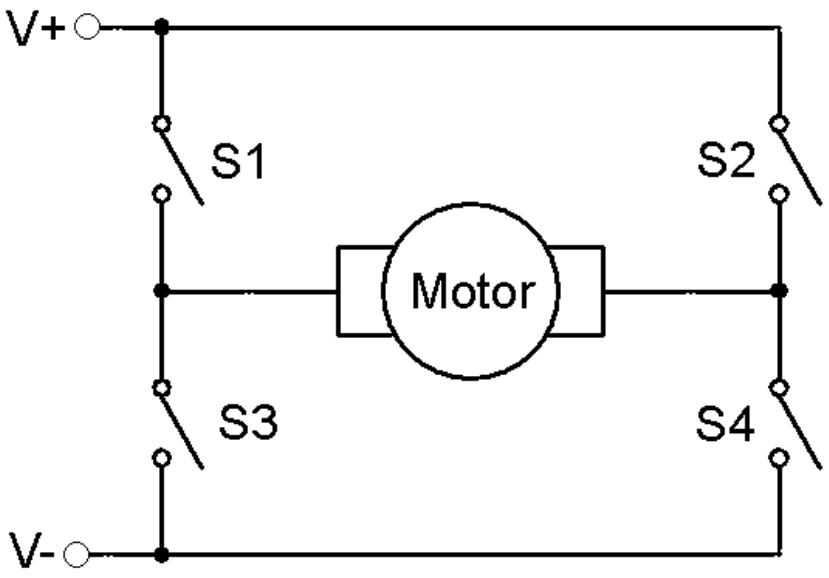



***Figure 1-7: H-Bridge***

The last part of the project is to write a software interface between the motor drivers and sensors that will interact with the control system. This will enable LabVIEW to be replaced as the interface will be written in MATLAB making it fully compatible with the navigation and control software which are both also written in MATLAB [Jenson et al., 2003].

## 1.4. Operating Environment

The environment for MARVIN is currently the corridors of the first floor of C Block at the University of Waikato. This restricted environment is to be used initially for testing and debugging the navigation system. It will be expanded eventually to encompass the entire science block and the different labs associated within each block.

The development of the navigation algorithm is aided by the current absence of passive and dynamic obstacles situated in the corridor, although these can be added at different stages of testing. A computer generated map of the main testing environment can be seen in Figure 1-8. It shows how the two corridors on the first floor of C Block intersect. Also shown are the indentations on the sides of the corridors, which make certain parts of the hallway wider.

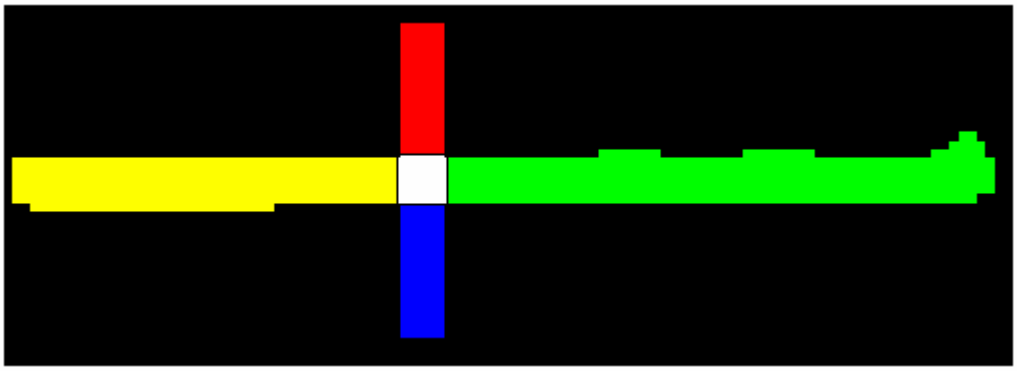


***Figure 1-8: MARVIN's main environment***

The environment depicted in Figure 1-8 has been broken up into five sections as indicated by the colours shown. Each section of the environment can be seen in the figures below. The photos were all taken from the white square in the middle.

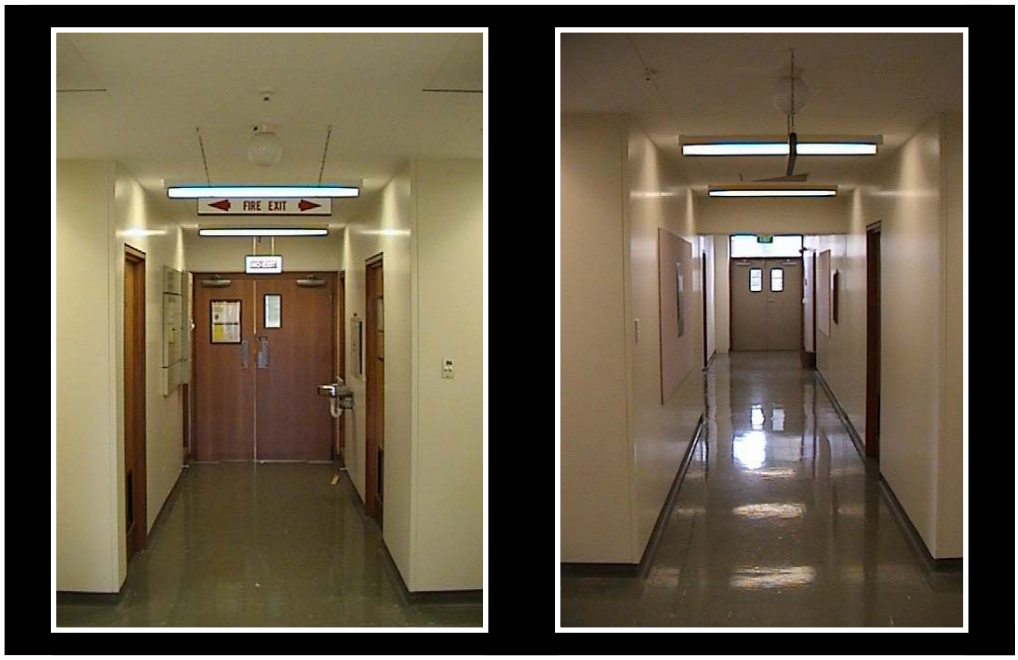



***Figure 1-9: Red and yellow sections***

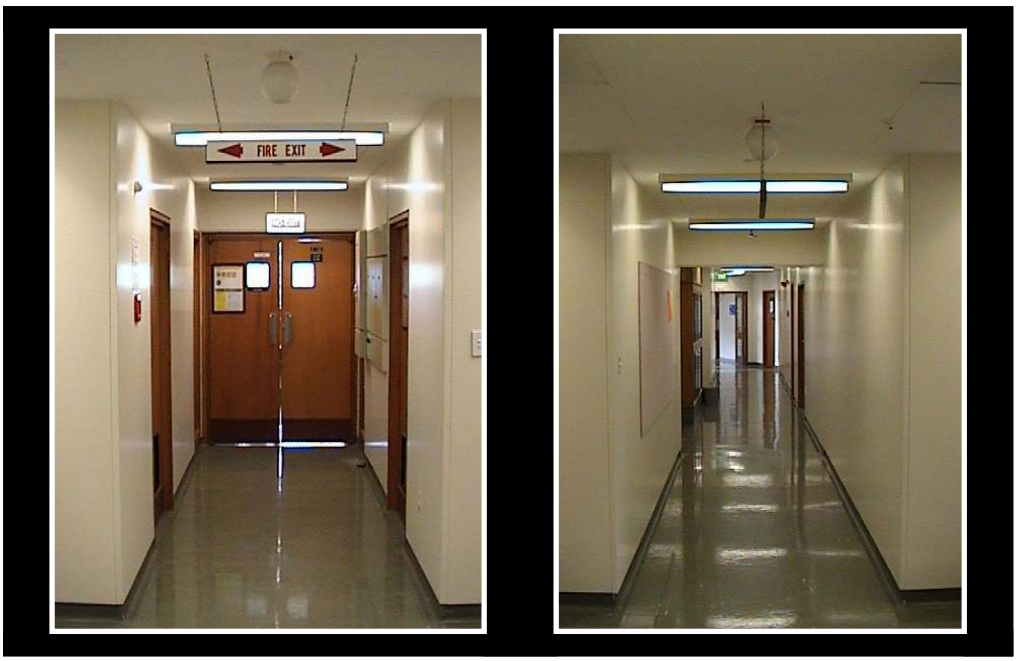



***Figure 1-10: Blue and green sections***

Further testing of the obstacle avoidance algorithm was performed in the Electronics Laboratory due to constraints in the width of MARVIN's main environment. This is discussed further in section 4.1. A map of this environment can be seen in Figure 1-11.

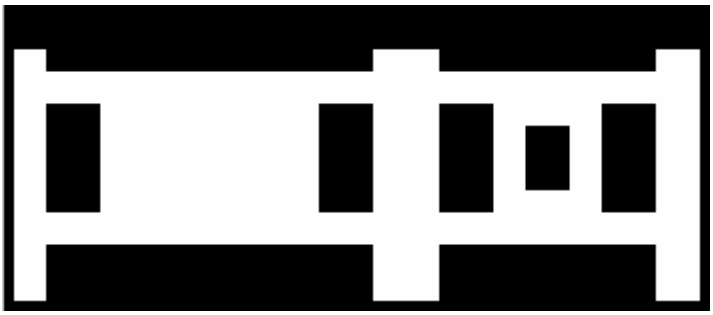


***Figure 1-11: Map of the Electronics Laboratory***

## 1.5. Project Objectives

The main objective for MARVIN is to navigate autonomously through the corridors of the first floor of C block at the University of Waikato. Being fully autonomous, the device must be able to conduct designated tasks independently without external help. During its journey it must be aware of its current position at all times.

In order for this to be achieved, several smaller objectives have to be obtained. The first of these is an interface between the voice recognition system and the navigation system. This will entail interpreting the voice string sent to the interface, identifying key words related to locations in MARVIN's environment. These key words will then be transferred into coordinates so the device can navigate from the current position to the target coordinates.

A technique for braking intended journeys up into small manageable tasks will also have to be devised. This will help simplify complex paths into straight lines and right angle turns and possibly integrated turns.

A form of feedback to the control system will be coded to help keep MARVIN in the centre of the corridors. As the navigation system has access to a map of the

environment and the corridors do not have uniform walls, variables will be sent to the control system. These will indicate the changing wall widths and other aspects of the environment that are not known to the control system.

Obstacles that appear in front of the device will have to be taken into account, with a new course being assigned to avoid these, and to get the mechatron back on its original course. This will also happen if for some reason MARVIN ends up off course.

A beacon system will be developed to aid in localisation and eliminating the accumulative error associated with odometry encoders. The compilation of these objectives being achieved will help achieve the overall goal of autonomously navigating MARVIN.

## 1.6. Report Structure

The report is presented in the following manner:

**Chapter Two**: Discusses and describes sensors used by mobile robots and also talks about other approaches used in the area of robotic navigation and localisation

**Chapter Three**: Gives an overview and description of the hardware and mechanical work done to the body of MARVIN.

**Chapter Four**: Explains the software used in developing the navigation algorithm and obstacle avoidance and how it operates.

**Chapter Five**: Displays and discusses the results of the project.

**Chapter Six**: Summarises the results, including discussing future work and final conclusions.

# 1.7. Development

The development process of the navigation project will take the following path:

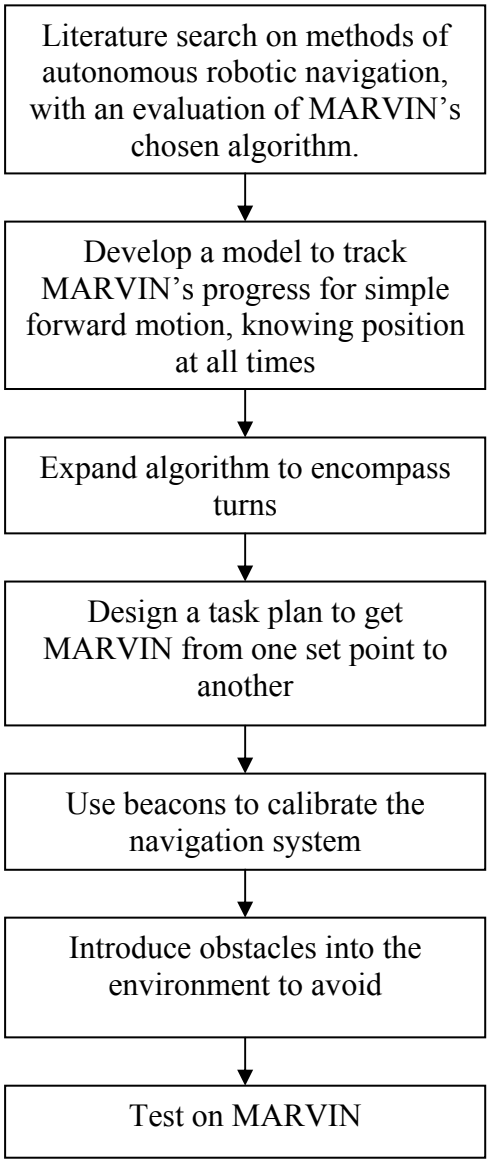



***Figure 1-12: Project flow chart***

# 2. NAVIGATION

## 2.1. Navigation Overview

The main goal of this project is for MARVIN to be able to navigate autonomously from a known start point, to a known goal point, being aware of its position at all times. This requires that MARVIN determine an appropriate path, and evaluate the control system commands necessary to implement this path.

To know its current position (i.e. determine the localisation issues) MARVIN is equipped with a variety of sensors. Shaft encoders are attached to the wheels of the robot. They accurately measure and in turn help to control the rotation of the wheels. Using the encoder readings and kinematics, the robot's position can be calculated. However, due to accumulating errors that are inherent in odometry measurement, this task becomes increasingly difficult. The result of these accumulating errors is that as the robot moves, the difference between the actual distance travelled and the distance reading obtained from the shaft encoders increases. This is mainly due to wheel slippage, but also from missed digital pulse counts and transmission slop [Victorino et al, 2000].

Because of this error, an additional sensor is normally required to compensate and even reset this error if possible. The sensors available, or those preferred by a particular research group, can determine the form of localisation method used.

There are two main techniques researchers have used to localise their robots in their particular environment. These are through:

- Landmark recognition
- Updating Beacons

This chapter will give an overview of the different sensors used by mobile robots as well as discuss the various methods of localisation and mapping techniques used in this area.

## 2.2. Sensing Overview

### 2.2.1. Visual Sensing

These particular sensors usually comprise two main parts. The first part consists of a receiver surface to receive the incoming light, with a lens above this surface to focus the light and improve the quality of the image. By improving the image quality, the computer processor will be able to achieve better pattern recognition. The last part of the system involves converting the light focussed on the receiver surface into a form that is compatible with a standard computer.

A Charged Coupled Device (CCD) camera is the common method used to receive and collect light from a particular point of interest. Used in conjunction with a form of digital conversion to enable the signal to be interpreted by a computer, the image can be processed.

Visual sensing is potentially the most important sensor used by mobile robots. Unfortunately this form of sensing generates an abundance of information from a single image when usually the area of interest is only a fraction of the picture, making it difficult to interpret certain visual features for positioning information.

An essential problem with using these particular sensors is an object's distance and size is hard to interrupt from each image taken of a particular part of the robot's environment. Image processing techniques are required to extract depth information from the images, which can prove to be very processor intensive.

There are a large number of different camera types available. The major considerations are whether the camera is analogue or digital, black and white or colour, and whether it has a low or a high frame rate with a good resolution.

Cameras that are suitable for robotic applications are generally required to be digital in order for the computer operating the device to be able to interpret the image and process it in the desired fashion.

Colour cameras are normally preferred over black and white as they enable more information to be obtained from a captured image. For some environments, landmarks may be able to be differentiated through colour and this can help in identifying other parts of the environment for localisation which black and white cameras can't provide.

Frame rate can also play a part in selecting a particular camera. A higher frame rate enables more information to be obtained from the environment per second which can help in identifying changes faster. As some image processing techniques such as Kalman filtering (this technique uses the statistical characteristics of received signals over time, giving greater accuracy than instantaneous measurements) use time information, so the more data received per second can help improve the accuracy of these techniques. A higher resolution enables a more detailed image to be obtained. A disadvantage of cameras with a high frame and with a high resolution is the price which tends to increase as these specifications increase.

A suitable camera for this application is a Pulnix TM-765I Interlaced, CCD Camera. This camera is an analogue, black and white camera with a frame rate of 50 frames per second. The main benefit of this camera is that it is already available to the Mechatronics Group as it is part of a previous masters project. Shaun Hurd developed a laser range finder system that made use of the camera and an Imagenation PX610A frame grabber card aswell allowing the data to be in a form able to be interpreted by the robot's computer.

As this camera is currently part of the laser range finding system and this particular sensor was designed for MARVIN it was deemed to be appropriate to use instead of further exploring the camera in a machine vision role.

A possible upgrade option is the Pulnix TMC-6700CL from OPSCI. It is a colour digital and analogue CCD camera with a high frame rate of 60 fps and costs $2545 US.

### 2.2.2. Sonar Sensing

Sonar sensors emit ultrasonic energy and are an example of reflective sensing. These sensors usually make use of an emitter and receiver pair. The emitter sends out a high frequency sound wave towards the area of interest. The wave is reflected back off the object to the receiver. The distance to the object is calculated using the Time of Flight (TOF) method assuming the speed of the sound wave is a known constant.

Sonar sensing is a cheap and affordable way of detecting the proximity of objects and is accurate over several metres. But there are several drawbacks to using these sensors. As the speed of sound can be affected by changes in atmospheric conditions, small errors can be introduced when calculating the distance of the object to the mechatron.

The accuracy of these sensors is also dependant on the span of the transmitted signal. The wider the span, the higher the chance of detecting an obstacle, but narrower spans allow the object's position with respect to the device to be more accurately known. False readings can also be generated through specular reflections. Additionally, once the emitted energy beam gets below a critical value, the reflected beam will not be within the acceptance angle of the receiver and therefore will not be detected. This can occur if the device is at an angle below the critical with respect to the walls of the environment or the particular object the device is trying to detect.

The SRF04 Ultrasonic Sonar Sensor from Robotics World is a suitable low range sonar sensor for possible applications of wall and object detection. It is a very small

package with dimensions of 4.5 cm in width and 1.6 cm in height. It has a minimum range of 3 cm with a maximum range of 3 m and is sensitive enough to be able to detect a stick 3 cm in diameter at distances greater than 2 m.

Another sensor that can perform the required sensing is from Acroname Products, part number R15-SONAR2 and is a package comprising a SensComp 7000 Series Transducer and a SensComp 6500 Sonar Ranging Module. The package has a range from 15 cm up to almost 10.6 m with an absolute accuracy of ± 1%, making it suitable for longer range obstacle detection. It comes with interface cables, application notes and specifications for $45 US.

As stated in the previous section, the custom made laser range finder designed for MARVIN will initially be used. If this proves inadequate and the option of using sonar is revisited, the sensor package from Acroname Products would be an adequate replacement and it has a very similar range to that of the custom made laser range finder that is discussed below.

### 2.2.3. Laser Range Finder

Laser Range Finders use two main approaches to calculate the distance to a particular object. The first uses TOF as discussed earlier in the sonar section, except now the speed is equal to the speed of light rather than sound. The second approach employs triangulation.

The triangulation method projects a beam of laser light onto the surface of the target. The target then diffusely reflects the light, with some of the reflected light reaching the detector. The location of this light is dependent on the distance between the sensor and target (Figure 2-1). As the distance between the emitter and receiver are known along with the angle between them, the distance to the particular target can be calculated using trigonometry, in particular the Law of Sines. This law states “the ratios of the sides of the sines of the opposite angles are equal” [Anton, 1988].

From Figure 2-1 the following can be seen:

$$\frac{L}{\sin\alpha} = \frac{D}{\sin\theta} \qquad \textit{Equation 2-1}$$

where:

L = separation between light source and detector.
D = distance to the object of interest.

Due to the fact sin α = sin(180 – α) = sin(β + θ), Equation 2-1 can be rewritten with D as the subject, given by:

$$D = \frac{L\sin\theta}{\sin(\beta+\theta)} \qquad \textit{Equation 2-2}$$

Therefore by measuring the distance between the light source and detector, and also measuring the angles β and θ, the distance to the object can be calculated.

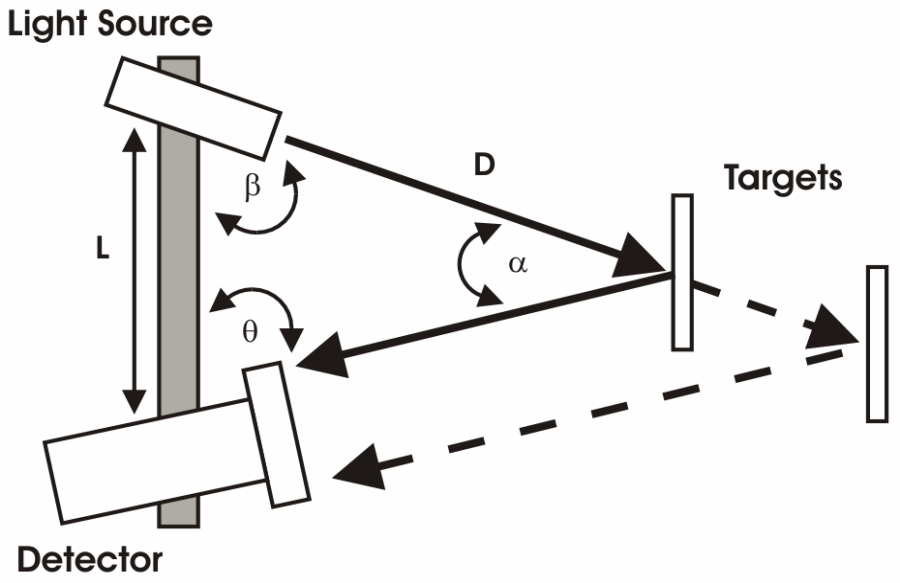



***Figure 2-1: Triangulation diagram***

The different kinds of detectors that can be used include CCD arrays, Position Sensitive Detectors (PSD) or diode arrays. These particular detectors allow the

received light to be detected at high sampling frequencies and with high spatial resolution.

Shaun Hurd's laser range finder uses the triangulation method and is suitable for use on mobile robots. It uses a class IIIa 633 nm laser as the emitter and a Pulnix TM-765I interlacing CCD camera as the detector as shown in Figure 2-2 below. It has a minimum range of 0.5 m with a maximum range of up to 10 metres and is discussed in more detail in section 3.2.3.4 [Hurd, 2001].

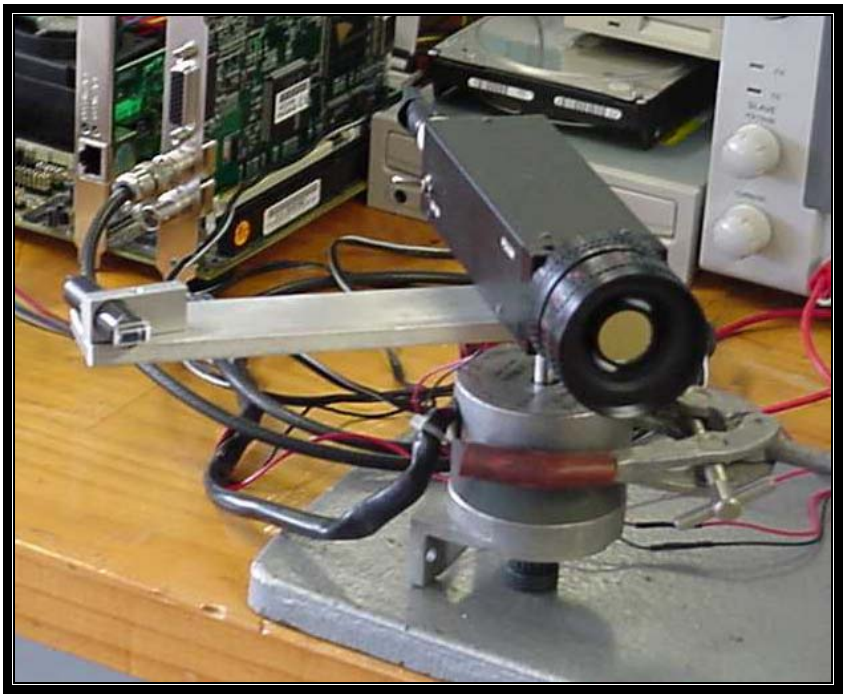


***Figure 2-2: Laser Range Finder designed by Shaun Hurd***

## 2.2.4. Infrared Sensing

These sensors use two main techniques to find the distance from a particular point to an area of interest: Triangulation (as discussed in section 2.2.3) and intensity.

The intensity technique involves an emitter sending out a beam of infrared light that is reflected off the object and received by the emitter. The distance of the object is determined by the intensity of the received infrared light which is usually not a linear relationship and is different for each sensor. Importantly, the intensity of the received

light can be affected by different surface textures and colours which is why the triangulation method is often preferred.

A sensor that can be used is the Sharp GP2D02 infrared ranger from Acroname Products. This detector uses a 2-wire serial interface and only takes readings when requested, leading to less power consumption. It has a range of 10 to 80 cm and uses triangulation to obtain the distance to the object of interest. The detector generates a digital output and costs $19 US.

Another infrared sensor that also uses triangulation to obtain distance information is the Sharp GP2Y0A02YK infrared distance sensor also from Acroname Products and can be seen below in Figure 2-3. It continuously outputs an analogue voltage corresponding to the distance measured and requires no external control circuitry for $15 US. It has a minimum sensing distance of 20 cm and a maximum sensing distance of 1.5 m.

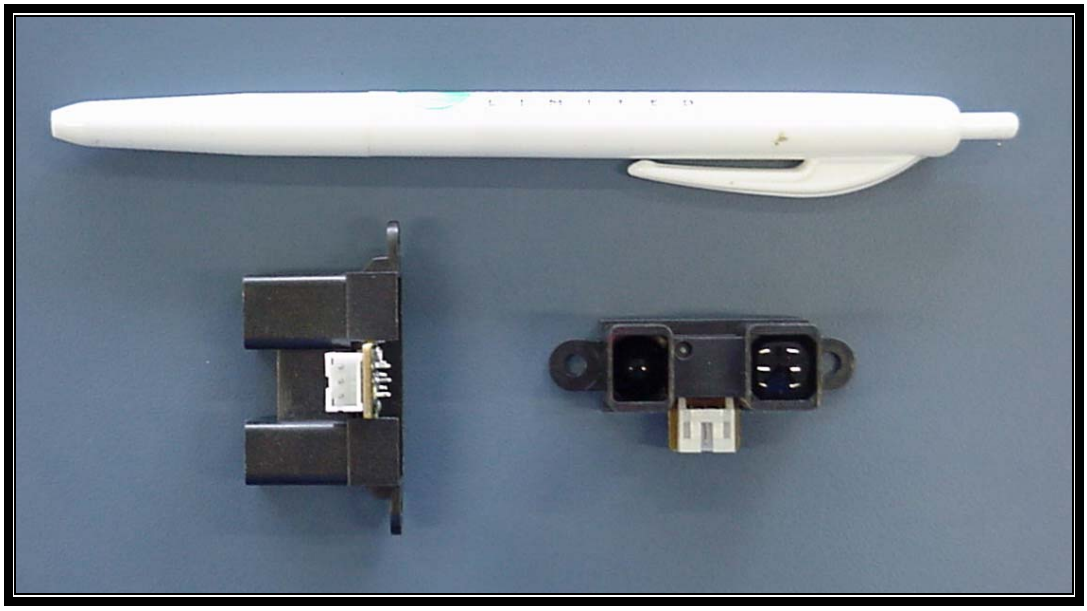


***Figure 2-3: Sharp GP2Y0A02YK infrared distance sensor***

The Sharp GP2Y0A02YK infrared distance sensor (discussed further in sections 3.2.3.3 and 3.3.2.2) was selected to be used on MARVIN as it possesses several desirable aspects. As MARVIN's environment includes a corridor of width 1.98 m at its widest point, and MARVIN is 0.583 m wide, the maximum range of this sensor

will never be exceeded. Furthermore, the sensor continuously outputs distance data enabling the distance from MARVIN to the corridor walls to be known at all times. Being analogue, the sensor is easier to interface with the DAQ card than a digital version of the sensor due to the latter's complexity of setting up a serial communication protocol.

### 2.2.5. Dead Reckoning

Before the end of the 15th century, sailors navigated by what was then called "deduced" (or "dead") reckoning. This method of navigation was used by Columbus and many other sailors of his era. It involved the sailor finding the position of the vessel by measuring the course and distance he has sailed from a known start point.

A high percentage of mobile robots are based on dead reckoning which use odometry to implement this form of navigation. The principle operation of odometry encoders is to monitor the rotation of each wheel to determine the device's speed, heading and distance travelled. The rotation of each wheel is generally measured using a form of code wheel, which is a circular disc with a fixed amount of holes around its perimeter and an optical encoder as shown in Figure 2-4 below. By monitoring the number of counts of the code wheel, the distance travelled by each wheel can be obtained. These sensors are usually very similar with the only significant difference between them being the number of slots on the code wheel.

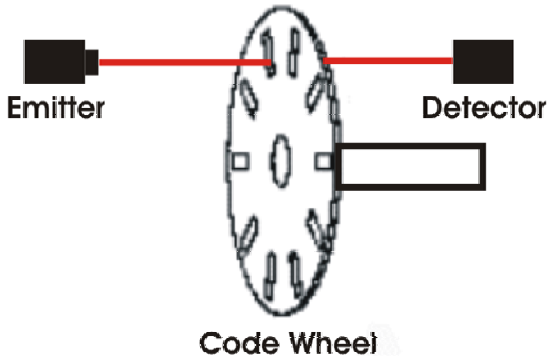



***Figure 2-4: Optical encoder diagram***

This technique is prone to accumulative errors which will continually compound due to such events such as wheel slippage and missed counts. For this reason, dead reckoning systems are often used in conjunction with other sensors.

The Hewlett Packard HEDS5500-A11 Package is a suitable two channel optical encoder which generates 500 pulses per revolution. It is comprised of a Hewlett Packard HEDS-9100 encoder module and Hewlett Packard HEDS-5120 code wheel for $78 US. The package can be seen below in Figure 2-5.

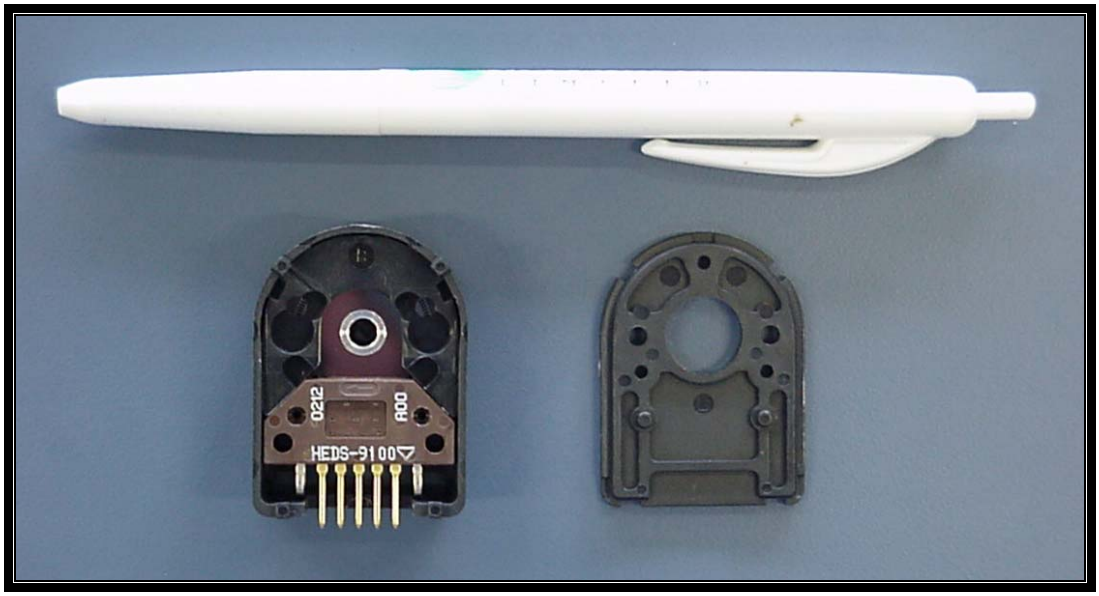



***Figure 2-5: Hewlett Packard HEDS5500-A11 optical encoder***

This sensor was used on MARVIN as the original encoders were almost identical. The only difference is that this encoder package comes with a housing that encloses the code wheel and optical encoder, protecting the sensor; these are discussed further in sections 3.2.3.2 and 3.3.2.3. The earlier version came in two parts with the code wheel being mounted to the shaft of the motor and the optical encoder mounted to the motor housing.

## 2.2.6. Global Positioning System

The Global Positioning System (GPS) is maintained and operated by the US Department of Defense and was first initiated in 1973. The system uses 24 satellites

that are positioned in six orbital planes, 20,200 km above the Earth's surface and circle the earth twice a day.

The satellites return a unique digital code to the receiver that contains the position of the satellite as well as timing information. Using the TOF method, the known signal propagation times can be used to find the line of sight distances to the various satellites that are currently being detected by the receiver in their known positions. A minimum of four satellites have to be detected by the receiver to give a position estimate, with the more satellites detected, the more accurate the position estimate. The position is calculated through a trilateration technique based on the TOF information.

GPS has several limitations. The antenna on the receiver needs to be able to have a direct line of sight with the satellites, objects such as trees, buildings and human bodies can block the signal from a satellite. This makes GPS suitable for outdoor use only, limiting its applications on mobile robots as the vast majority are designed for indoor use.

The TR Control Solutions DS-GPM GPS module from Acroname Products is a suitable sensor for this particular application. It provides the data in easy to access registers via I2C and RS232 communications. It costs $299 US.

Another suitable GPS for an outdoor mobile robot is the Motorola Oncore M12 GPS Receiver. The benefits of this receiver are that it has low power consumption, has battery backup and is very small with dimensions of 40 x 60 x 10 mm. The receiver can be purchased from Avnet Kopp (Pty) Limited for $200 US.

As MARVIN's environment is indoors the option of using GPS is not feasible. The Motorola Oncore M12 GPS Receiver discussed above is used by the co-operative robots being designed by fellow masters student Andrew Payne.

## 2.3. Mapping Techniques

The two main techniques used to implement maps on autonomous robots are through built-in maps and self generated maps. Each approach has its own benefits and problems.

Built-in maps take the approach of storing a blueprint of the robot's environment. This method is computationally easier to implement than a self generated map but has the essential problem of not necessarily being constantly reliable if the environment is dynamic. Dynamic environments contain objects that do not necessarily remain in the same place, such as chairs, tables and humans.

Self generated maps are created by the robot as it is navigating around an enclosed environment. The robot is able to relate its current position with its own perception of its environment. The advantage of this is that a robot can be placed in a room it has never been in before, and is able to navigate around inside it. However, it can be extremely difficult to accurately create such a map as noted by the closed loop problem. This problem relates to knowing if you are in a room you have been in before if you enter via a different entrance.

The two main forms of maps used are metric maps and topological maps (also referred to as relational maps). Metric maps are based on an absolute reference frame and numerical estimates of where objects are in the environment. Topological maps explicitly represent connectivity information, usually in graph form [Dudek & Jenkin, 2000].

## 2.4. Landmark Recognition

There are several factors that characterise a landmark position estimation system. Important aspects to consider include are the landmarks passive or active? Active landmarks typically emit unique signals whereas passive landmarks are generally natural to the environment, for example doors. Also, what is the sensing modality?

(e.g., vision, sonar, etc.) What are the geometric properties of the landmark? (e.g., circular, square, etc.) Lastly, how easy is it to detect, identify, or measure a landmark? [Dudek & Jenkin, 2000].

The problem of landmark recognition relates to orientation estimation of a landmark with respect to a fixed sensor. One of the keys for success in this area is the choice of landmarks to be used. Good landmarks are those that remain reliable to an autonomous robot over time. If a change in the robot's position or orientation occurs, the robot should still be able to identify the landmark. Sections 2.4.1 and 2.4.2 will discuss the two main landmark recognition techniques.

### 2.4.1. Visual Landmark Recognition

Vision sensors enable a large amount of information to be obtained from the environment. There is a limit to the information that can be acquired from a single image due to the narrow field of view offered by normal camera systems. Even considering this limit, the amount of information obtained is often too large as what is wanted is generally only a small fraction of the total picture. One of the keys in this area is to isolate the relevant information and discard the rest.

[Thompson et al., 2000] designed a successful visual landmark recognition system, using convex mirror shapes to enable greater fields of view. The Panoramic mirrors provide a full 360 degrees of horizontal visual field and over 140 degrees in the vertical field. This is illustrated in Figure 2-6 below.

***Figure 2-6: Panoramic image***

By using landmarks in localisation, only parts of their robot's environment need to be stored. Landmarks are required that would be "reliable" to a changing environment, especially in regard to changing perspective and light levels, in order for their robot to still able to detect them. This can be seen below in Figure 2-7, where the landmarks can still be identified by the robot with different levels of illumination over time as indicated by the coloured boxes.

***Figure 2-7: The panoramic view with changing illumination***

As they use a panoramic scene that encloses possible landmarks in all directions, a Turn Back and Look (TBL) approach is used. This tests the reliability of the landmarks after movements in multiple directions.

Possible landmarks are tracked as the robot moves along a portion of their environment. At approximately 400 points along the path, the landmark's reliability measures are evaluated. Each landmark's final reliability value corresponds to the average of these 400 measures. Landmarks with the highest average reliability from each sector are used to represent that particular place.

Localisation then involves matching a set of landmarks, with their location, to the current visual picture. The robot is now assumed to be in the area related to the set of landmarks with the best relationship to the particular picture.

### 2.4.2. Sonar Landmark Recognition

Sonar sensors (as discussed earlier in section 2.2.2) are a common choice due to their weight, low price and long detection range.

When using sonar sensors, a popular approach is to organise the information into evidence grids. These are used to convert wide angle range measurements into detailed spatial maps. The grids are used due to the fact that sonar has an arc of uncertainty, as specular reflections can give misleading values, and so multiple readings are required. A benefit of the grids is that they can be quickly updated. To minimize the occurrence of problems, most techniques search for important objects in local environments, but this technique is mainly limited to static environments.

[Yamauchi & Beer, 1996] confront dynamic environments by storing local grids of 64 × 64 cells, which are rotated and translated to find the best match with each grid learnt.

[Bandera et al., 2000] designed a sonar based localisation technique that was useful for both dynamic and static environments. Their system stores a short feature vector, which is obtained from sonar readings via a simple transform, rather than using local grids.

Their method had the following advantages:

- The feature vectors can be created at any position, even in unstructured environments.
- They do not depend on the robot's orientation.

The biggest problem with landmark recognition using sonar is that they are susceptible to specular reflections. This problem is avoided as the robot relies on

gathering readings through as many movements as is required, rather than a single sonar reading.

Evidence grids mainly supply the robot with information about distances between itself and the nearest obstacles for each particular beam direction in polar coordinates. Bandera et al. store this information into a one-dimensional function, called a depth map.

The problem of dependence on the robot's orientation is solved by calculating a circular correlation to find the similarity of two depth maps. An important advantage of depth maps is that as they are one dimensional, it is simple to compare them and they have less data than an average evidence grid.

Finally Bandera et al. calculate their landmarks by projecting the Fourier Transform of the depth map onto a bidimensional base of its vectorial subspace. This is illustrated in Figure 2-8 which shows the sonar readings obtained by the robot in a), followed by a 64 point depth function acquired from a random location inside a messy room in b). As depth maps are still quite large, it can be shown that the Discrete Fourier Transform (DFT) of an average depth map produces a high amount of zero components as shown in c). By projecting the Fourier Transform on the depth map as discussed at the start of the paragraph, landmarks can be calculated as shown in d).

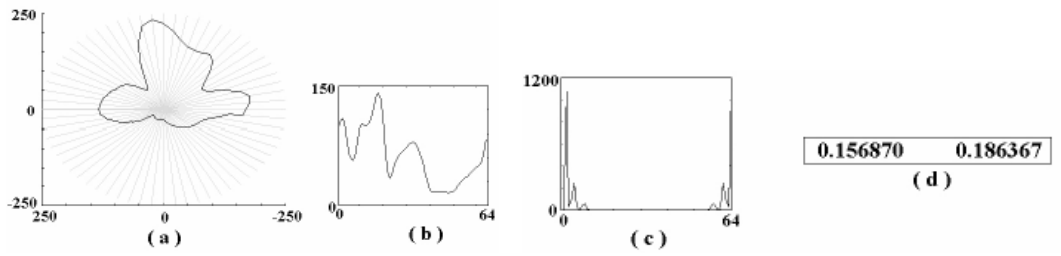



***Figure 2-8: a) Sonar readings; b) Depth map; c) Discrete Fourier Transform of an average depth map; d) Landmark.***

# 2.5. Updating Beacons

Updating beacons are another approach to help localise robots. They can be thought of as artificial landmarks due to the fact they can take any form and can be distributed wherever the designer likes. Researchers in the field of robotics have developed several robust systems using beacons of different forms.

## 2.5.1. Light Beacons

[Launay et al., 2002] developed quite a robust system that involved lights on the roof that acted as beacons. Initially a map is acquired by guiding the robot in corridors. This map is completely based on odometry. Since there are errors involved with odometry as discussed earlier, this map is not topologically correct but can still be used by the robot to localise itself while navigating.

The pose of the robot is represented with (X,Y,Θ) Cartesian coordinates, the third variable being an angle to help indicate the robot's pose. The coordinate system used is illustrated in Figure 2-9 below:

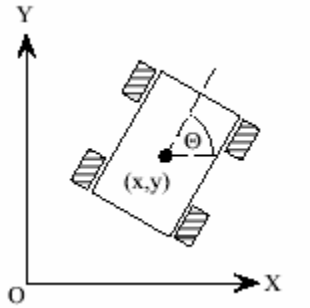



***Figure 2-9: Illustration of the coordinate system [Lee et al, 2000]***

Launay et al.'s robot uses a camera aimed at the roof. It is directed under each light, adding beacon information to the map whenever a new light appears above the robot. They detect a light by finding a suitable histogram-based threshold for each image. If the number of pixels brighter than the threshold is bigger than a given value, it is

classed as a light, which is illustrated along with their camera arrangement in Figure 2-10.

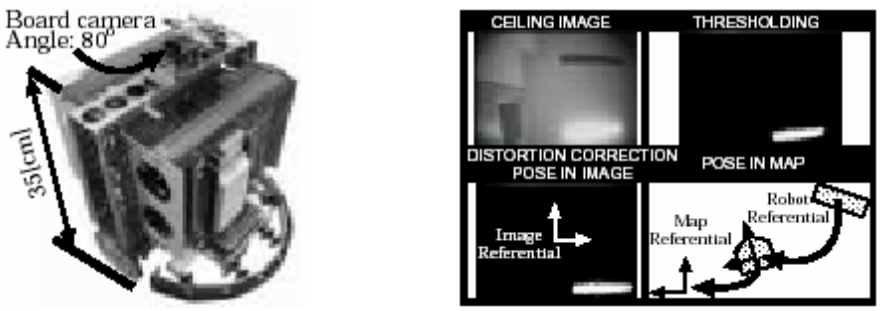



***Figure 2-10: The camera system and image processing steps***

## 2.5.2. Magnetic Field Beacons

[Prigge & How, 2000] have developed a system that uses magnetic fields to localize their robot. The advantage of this method is that it supplies pose and absolute position without line-of-sight constraints.

The system uses beacons that are located in fixed positions throughout the environment. These beacons generate magnetic fields at all times, but due to a special pseudorandom code, the current through each beacon changes polarity. The pseudorandom code is a special sequence of -1's and 1's, with each beacon being assigned its own code that repeats when its end occurs.

A mobile sensor unit is used to sample the local magnetic field. Gold codes (refer Appendix A.1) are used of length 1023 chips. The code sequences recycle and start at the same time, enabling the beacons to be synchronized. Due to the orthogonality of the Gold codes, each individual beacon's field strength can be estimated through a correlation.

All of the beacons produce fields continuously and a Code Division Multiple Access (CDMA) method is used to differentiate between the fields produced by particular beacons. Using the calculated field strengths from several beacons in conjunction with their positions allows the mobile sensor to ascertain pose and position.

The CDMA technique utilises the advantages of current magnetic field positioning systems, permitting many beacons to be operating simultaneously while preserving the benefits of sampling the magnetic field as it nears a steady state value. It also supplies a natural way to significantly increase the number of beacons in the system.

## 2.6. Other Localisation Techniques

Besides the two main localisation techniques of landmark recognition and localising beacons, researchers have investigated other means to localize their robots. [Suksakulchai et al., 2000] developed a localisation method using an electronic compass.

Electronic compasses are used to detect the heading of mobile robots. There is a problem in using these inside buildings, as ferro-magnetic structures such as bookshelves and electromagnetic sources such as power lines make a dependable heading value impossible.

After inspecting error readings of an electronic compass caused by these structures they found that heading errors for each specific position had an exclusive signature. As the magnetic interference along the hallway (which was caused by these particular objects), had their own characteristics, they were able to take advantage of this and use them to help localise their robot, HelpMate. The HelpMate robot and its compass location can be seen in Figure 2-11.

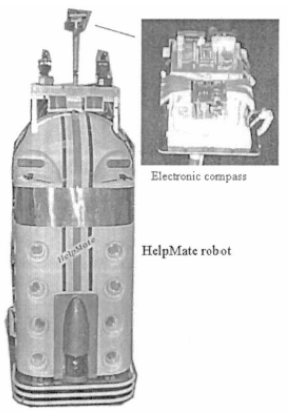



***Figure 2-11: Electronic compass situated on top of the HelpMate robot***

The compass shown in Figure 2-11 is mounted above the HelpMate robot so as to minimise any interference from the robot itself.

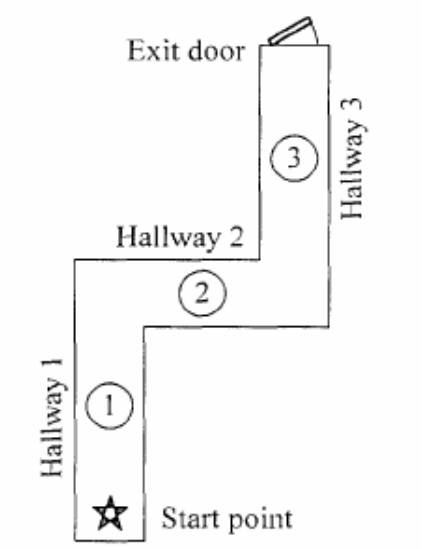



***Figure 2-12: Robots testing environment***

As can be seen in Figure 2-12, the robot's environment is separated into three parts for each side of the hallway. The robot travels down the centre of the hallway taking compass readings every 200 mm. As its approximate speed is 40 mm/s, a reading is taken about every 5 seconds. A least-squares approximation is used for matching the signatures.

A beacon system will be used on MARVIN to reset the accumulated odometry error. It will be similar to Launay et al. in the fact that lights are used but they will be located on the side walls of the environment and will be a modulated infrared light source. This will be discussed in more depth in the following chapter.

## 2.7. Summary

This project forms the first attempt to get MARVIN to navigate autonomously. The decision was made to use a built-in map in order to develop an initial navigation system. This could possibly be altered in future projects as discussed in section 6.2.1.

As the Mechatronics Group has access to a laser range finder developed by Shaun Hurd certain sensing options weren't furthered pursued. These options included image processing [Thompson et al., 2000] which are expensive and complex and also sonar, which requires significant signal conditioning. The option of using GPS wasn't explored due to MARVIN's intended environment being indoors. The infrared position sensors from Hewlett Packard discussed in section 2.2.4 were ordered and implemented on MARVIN to help with wall distance and heading information.

A beacon system similar to Launay et al. as discussed in the previous section will be implemented. The coordinate system will again be similar to Launay et al. as discussed in section 2.5.1 where MARVIN's position and pose will be indicated in the form (X,Y,Θ) as illustrated earlier in Figure 2-9.

At the moment MARVIN makes use of

- Tactile sensors
- Optical encoders
- Infrared distance sensors
- Infrared beacon receivers

The laser range finder will be installed and made use of in following projects once the initial navigation system has been implemented (refer to section 6.2.3).

# 3. HARDWARE & MECHANICAL LAYOUT

## 3.1. Overview

Before the navigation algorithm could be fully tested on MARVIN certain hardware and mechanical issues had to be addressed. In its initial state the device was non operational and required a mini overhaul to get it up to a standard where testing in the initial environment could occur. At the onset of the project the mechatron had two odometers fitted, one on each shaft of the motor for distance information, with no other sensors installed.

The initial PC being used was not fully enclosed in a case or similar housing, leaving the mother board and associated circuitry exposed. It rested on to two horizontal aluminium bars, with two screws used to keep it fixed in place. The PC was powered by an Uninterrupted Power Supply (UPS) that was connected to the car batteries at the base of the mechatron. The UPS was quite large and bulky and was mounted to the frame with Duct tape. Its purpose was to convert the 24 V DC from the batteries to 240 V AC which would be used to power the PC.

There were no motor drivers interfaced with the device, but there was a housing to accommodate these, with a fan included for cooling. The original drivers had been used on another project. A connector block was cellotaped to the top of the frame. The purpose of this was to connect signals to the Data Acquisition (DAQ) Card on the PC, enabling various sensor data to be read into the computer. The batteries at the base of MARVIN were electrically isolated from the frame and stray bits of conductive material by a piece of cardboard.

A diagram of MARVIN in its original state can be seen in Figure 3-1.

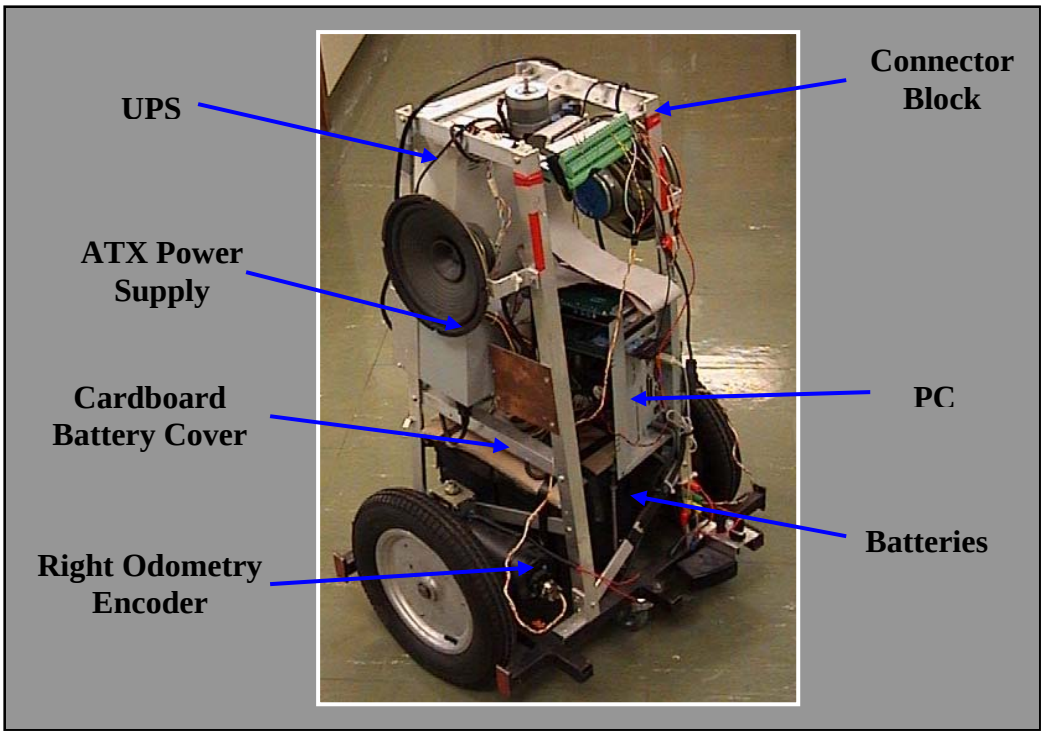



***Figure 3-1: MARVIN originally***

## 3.2. Hardware

The first aspects of MARVIN that were altered and added to were in the hardware area. Motor drivers needed to be developed along with a microcontroller board to control the motors. In addition to these, the various sensors had to be ordered and interfaced with the DAQ card.

### 3.2.1. Motor Driver and Microcontroller Board Development

The motor drivers and microprocessor board constructed and used by MARVIN were designed by fellow masters student Andrew Payne. Their initial application was to drive the wheels on two co-operative robots also designed by Andrew (see section 1.1.3). Due to delays in the prototype development of the generic motor drivers being constructed by masters student Craig Jensen, Andrew's drivers were employed on MARVIN.

The motor driver circuit uses a full H-Bridge configuration (Appendix B.1) which enables the direction of the mechatron to be controlled depending on which diagonal pair of transistors in the H-Bridge are turned on. As each wheel of the device is controlled independently, two motor drivers are used. The current to each wheel is controlled by varying the duty cycle of an input PWM signal. The 24 V the circuit requires is provided from the batteries at the base of MARVIN. The motordriver PCB can be seen below in Figure 3-2.

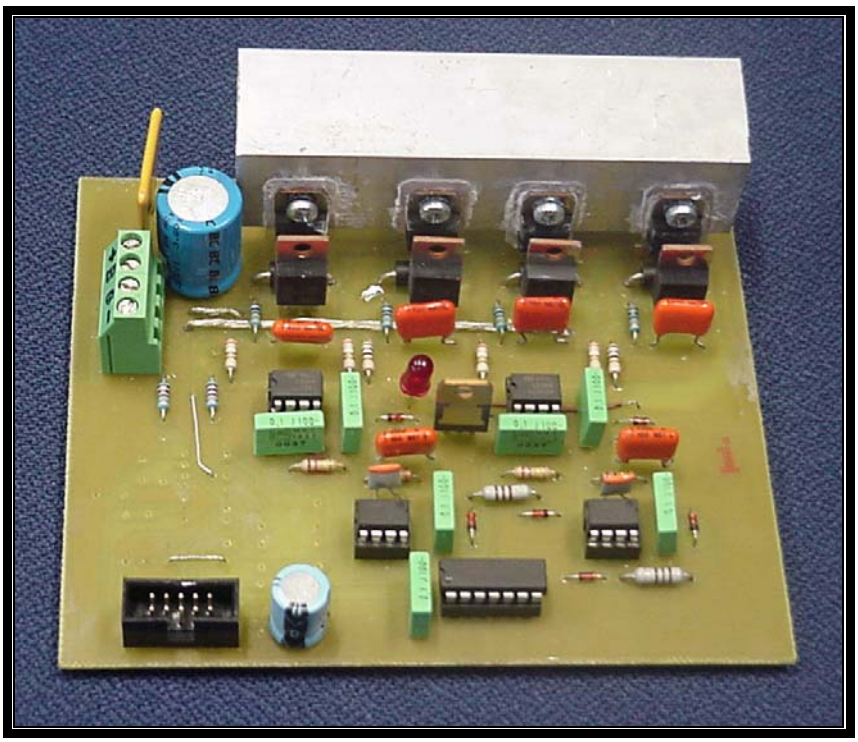


***Figure 3-2: Motor driver PCB***

The microcontroller circuit (Appendix B.2) uses a Philips P89C51RC2HBP microcontroller chip that is programmed to generate an appropriate PWM signal that is sent to the motor drivers to power the motors. The microcontroller interacts with the DAQ card through a 12-bit parallel interface. The interface comprises a 4-bit handshaking connection and an 8 bit data connection. Each instruction sent to the microcontroller to drive the motors is delivered as two bytes. The first byte represents the particular motor and which direction it is to be driven and the second byte represents the PWM value. The microcontroller interprets these instructions and sends the appropriate signal to the motor drivers. The PCB for the motor controller can be seen in Figure 3-3.

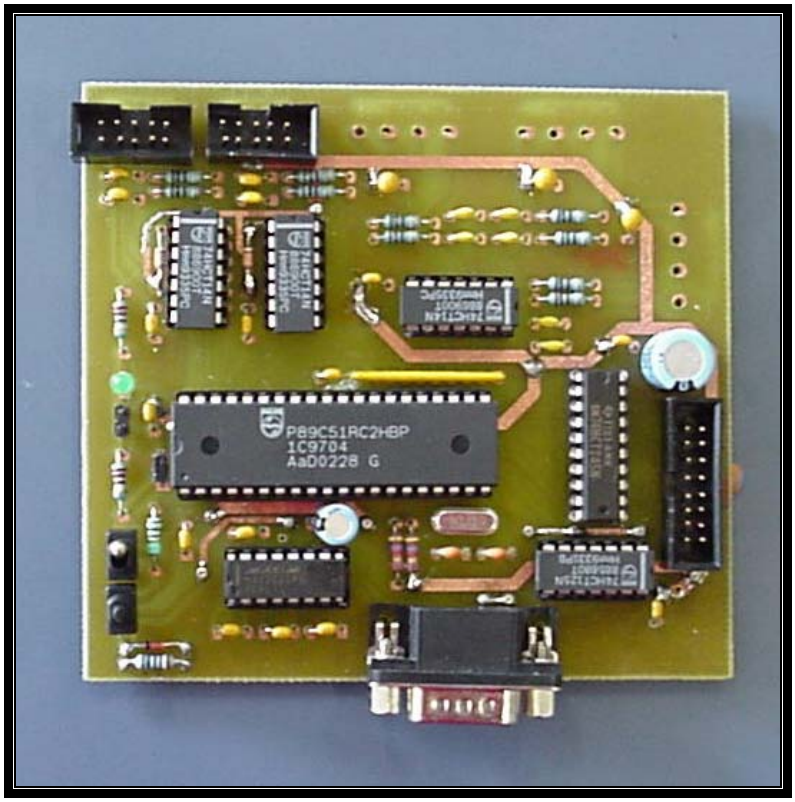



***Figure 3-3: Micro controller PCB***

Both circuit boards were tested on a 24 V DC motor, enabling small bugs in the circuit construction to be ironed out before tests were conducted on the actual motors of MARVIN under load conditions.

## 3.2.2. Upgrades

### 3.2.2.1. PC

The original PC used by MARIN can be seen earlier in Figure 3-1. It had Windows 2000 installed together with LabVIEW 5.1 and MATLAB 6.1. The basic specifications of the PC were as follows:

- CPU: Celeron 466 MHz
- RAM: 128 MB
- Motherboard chipset: AOpen MX3ZA
- Hard Disk: 6 GB

- Expansion Slots: 3 x PCI
  1 x ISA
- I/O Ports: 2 x USB 1.0
  2 x Serial DB9
  1 x Parallel DB25

The housing for the PC as illustrated in Figure 3-1 was a standard PC casing with parts of it trimmed and cut away to enable it to fit inside MARVIN's chassis. This form of housing was not satisfactory as it left a significant proportion of the associated circuitry exposed.

As each control cycle of the navigation and control software needs to occur as fast as possible and with future tasks required of the PC likely to be affected by its current speed, it was thought a faster CPU would be preferable.

The new PC ordered was a Small Form-Factor (SSF) computer that was housed in a Shuttle xPC case. The size of the casing enabled it to fit with relative ease inside MARVIN's chassis, eliminating the need to make a custom mounting.

The only significant problem with the new computer is the fact it has only one PCI slot which is being used by the DAQ card. A problem that will arise in the future will be the installation of the laser range finder developed by Shaun Hurd [Hurd, 2001]. As this sensor uses a PCI video capture card there will be no room for it so another form of card may have to be used or possibly even another PC upgrade.

The new PC (Figure 3-4) has Windows XP installed together with MATLAB 6.1 and LabVIEW 5.1. The specifications of the PC are now as follows:

- CPU: Athlon 2000+
- RAM: 512 MB
- Motherboard chipset: nVidia nForce 2 / MCP-T
- Hard Disk: 10 GB
- Expansion Slots: 1 x AGP

| | |
|---|---|
| | 1 x PCI |
| • I/O Ports: | 4 x USB 2.0 |
| | 2 x IEEE 1394 |
| | 1 x Serial DB9 |

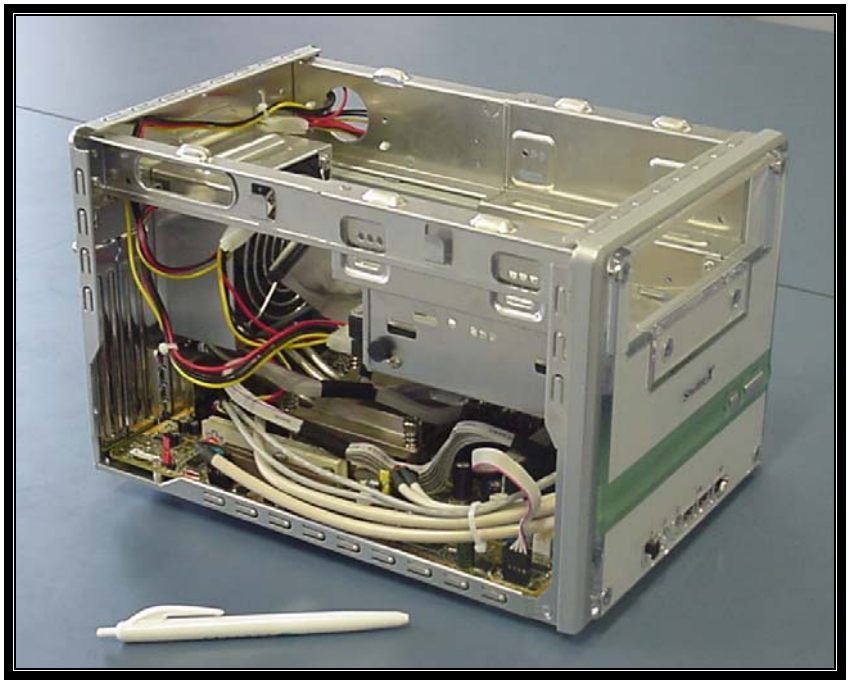


***Figure 3-4: MARVIN's new PC***

### 3.2.2.2. DAQ Card

The original National Instruments Lab-PC+ DAQ card (Figure 3-5) made use of the ISA slot on the original PC, enabling the sensors and PC to communicate with each other. The basic specifications of the DAQ card were as follows:

- Digital I/O
  - Number of channels: 24
- Analogue Inputs
  - Number of channels: 8
  - Resolution: 12 bit
  - Maximum sampling rate: 83 kS/s
  - Type: Successive approximation

- Analogue Outputs
  - Number of channels: 2
  - Resolution: 12 bit
- Counters/Timers
  - Number of channels: 2
  - Resolution: 26 bit
  - Maximum frequency 8 MHz
  - Type: Down

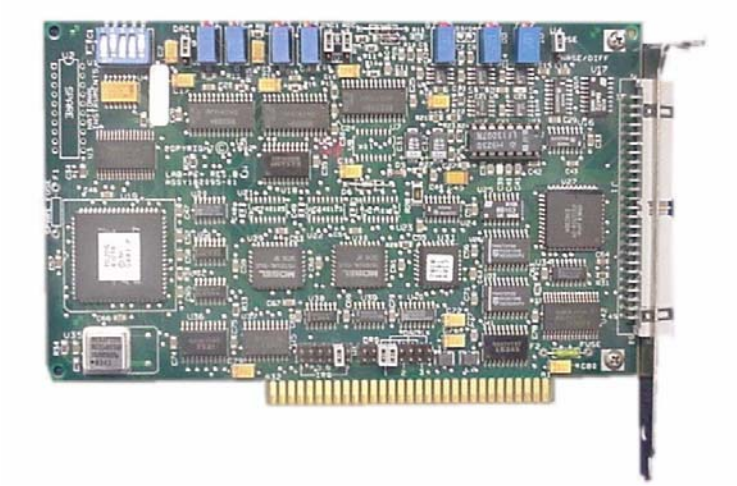


***Figure 3-5: National Instruments Lab-PC+ DAQ card***

This particular card was more than sufficient for the requirements of the project but due to the fact modern motherboards no longer support ISA cards, the card was now obsolete. The replacement 6025E card was sourced from National Instruments (Figure 3-6).

New DAQ card specifications:

- Digital I/O
  - Number of channels: 32
- Analogue Inputs
  - Number of channels: 16
  - Resolution: 12 bit
  - Maximum sampling rate: 200 kS/s

Type: Successive approximation

- Analogue Outputs
  - Number of channels: 2
  - Resolution: 12 bit
- Counters/Timers
  - Number of channels: 2
  - Resolution: 24 bit
  - Maximum frequency 20 MHz
  - Type: Up/Down

***Figure 3-6: National Instruments 6025E DAQ card***

### 3.2.2.3. Power Supply

MARVIN uses two 12 V car batteries connected in series, providing 24 V to power the various electronic equipment attached to the device. The power supply was connected to an UPS that was used to generate a 240 V mains equivalent signal. This signal was then connected to the ATX supply, which provided the required voltage levels for the operation of the PC. As this method of powering the PC involved a conversion between a low DC voltage to a high AC voltage and back again, it proved to be inefficient and included a step that could be eliminated.

A 24 V ATX power supply (Figure 3-7) was ordered to replace the previous standard ATX power supply. The new ACE-828C version from ICP Electronics doesn't require a mains equivalent signal, instead it can be run from the 24 V supplied by the

batteries. This enabled the bulky and heavy UPS to be removed, freeing up more space on MARVIN's chassis.

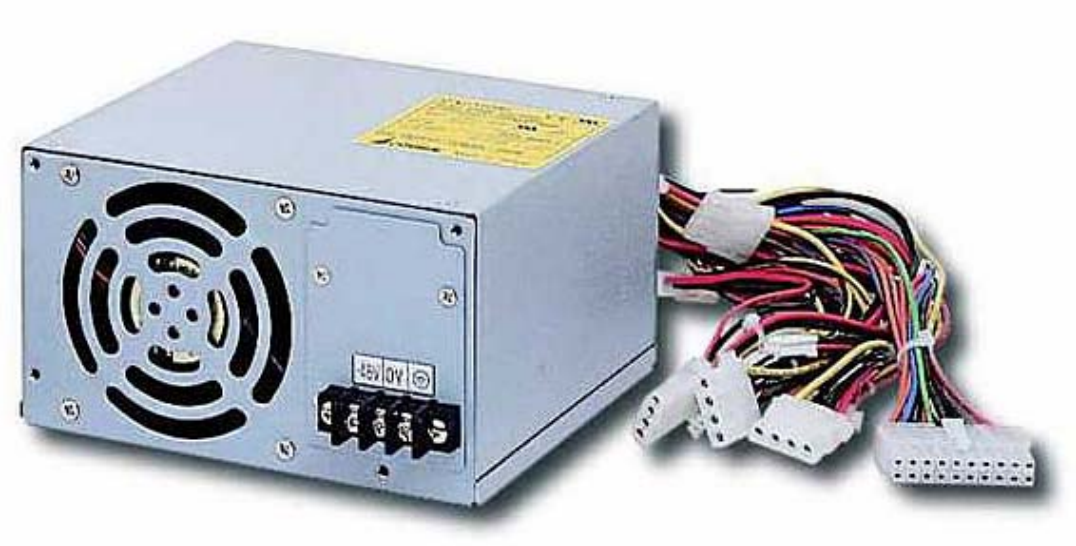


***Figure 3-7: ACE-828C ATX power supply from ICP Electronics***

### 3.2.2.4. Wireless LAN

The software on MARVIN is remotely operated from a notebook PC. The notebook and the mechatron's PC communicate over a wireless LAN connection using the Windows Terminal Service / Remote Desktop Connection tool present on Windows XP.

Due to MARVIN's motherboard having only one PCI slot, an internal wireless LAN card could not be used. This problem was overcome by using a USB module, the ZyAIR B-220 (Figure 3-8) which has the following specifications:

| | | |
|---|---|---|
| Media Acess Protocol: | IEEE 802.11b | |
| Data Rate: | 11 Mbps / 5.5 Mbps / 2 Mbps / 1 Mbps | |
| Coverage Area: | Indoor: | 50 m @ 11 Mbps |
| | | 80 m @ 5.5 Mbps or lower |
| | Outdoor: | 150 m @ 11 Mbps |
| | | 300 m @ 5.5 Mbps or lower |
| Frequency: | 24 ~ 2.835 GHz (Industrial Scientific Medical Band) | |

Output Power: 17 dBm (typical)

Receiver Sensitivity: -82 dBm @ 11 Mbps

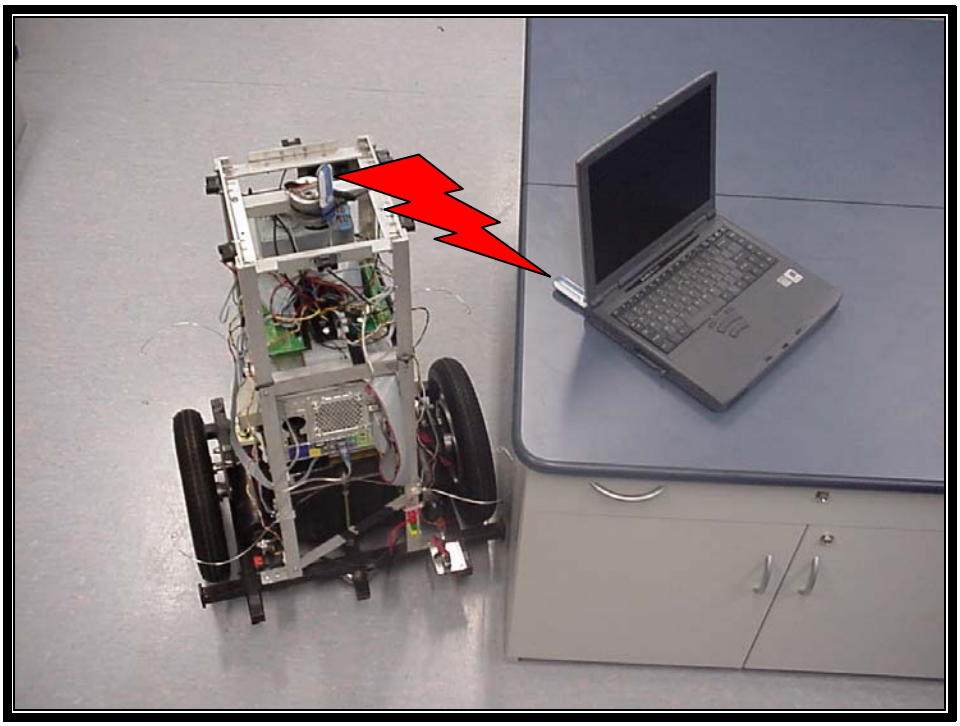


***Figure 3-8: ZyAIR wireless LAN***

## 3.2.3. Sensors

The sensors are vitally important to the movements of MARVIN. They help give the software an indication of how far the device has travelled and where it is travelling to, enabling the mechatron to avoid obstacles and helping with localisation. MARVIN makes use of an array of sensors for different distance and localisation applications ranging from simple tactile sensors to infrared proximity sensors.

### 3.2.3.1. Tactile Sensors

These basic sensors have been designed around a simple micro switch. The concept behind the sensor is that once MARVIN hits an obstacle, the switch will be closed by a wire feeler. The signal from the switch is sent to the DAQ card where it is polled by the control system to see if the signal goes high, indicating a collision. This signal

will then trigger a software routine that will stop MARVIN immediately. The signal obtained from the DAQ card is either high or low and comes from the simple circuit below (Figure 3-9):

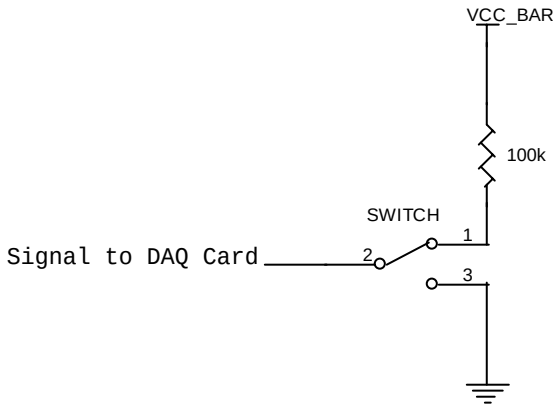



***Figure 3-9: Tactile sensor circuit***

### 3.2.3.2. Odometry Encoders

These optical encoders measure and help control the rotation of the wheels. They are used to give an indication of how far MARVIN has travelled, with each code wheel providing 500 counts per revolution. The optical encoder used is a Hewlett Packard HEDS-5500 (Figure 2-5, section 2.2.5) which contains a Hewlett Packard HEDS-9100 encoder module and Hewlett Packard HEDS-5120 code wheel.

A full rotation of MARVIN's wheels generates 25780 pulses. After factoring in the circumference of the wheels which was calculated to be approximately 1.0367 m, the amount of pulses per metre can be found to be 24867. After experimental tests in the device's initial environment, this value was found to be slightly off which was primarily attributed to different pressures in the tyres that alter the wheel circumferences. It was recalculated as being 28062 pulses per metre (discussed further in section 5.1). The outputs from the encoders are connected to the counter pins on the DAQ card.

### 3.2.3.3. Infrared Proximity Sensors

The main use of these sensors is for obstacle avoidance and wall detection. They have a minimum sensing distance of 0.15 metres and a maximum sensing distance of up to 1.5 metres as illustrated in Figure 3-10.

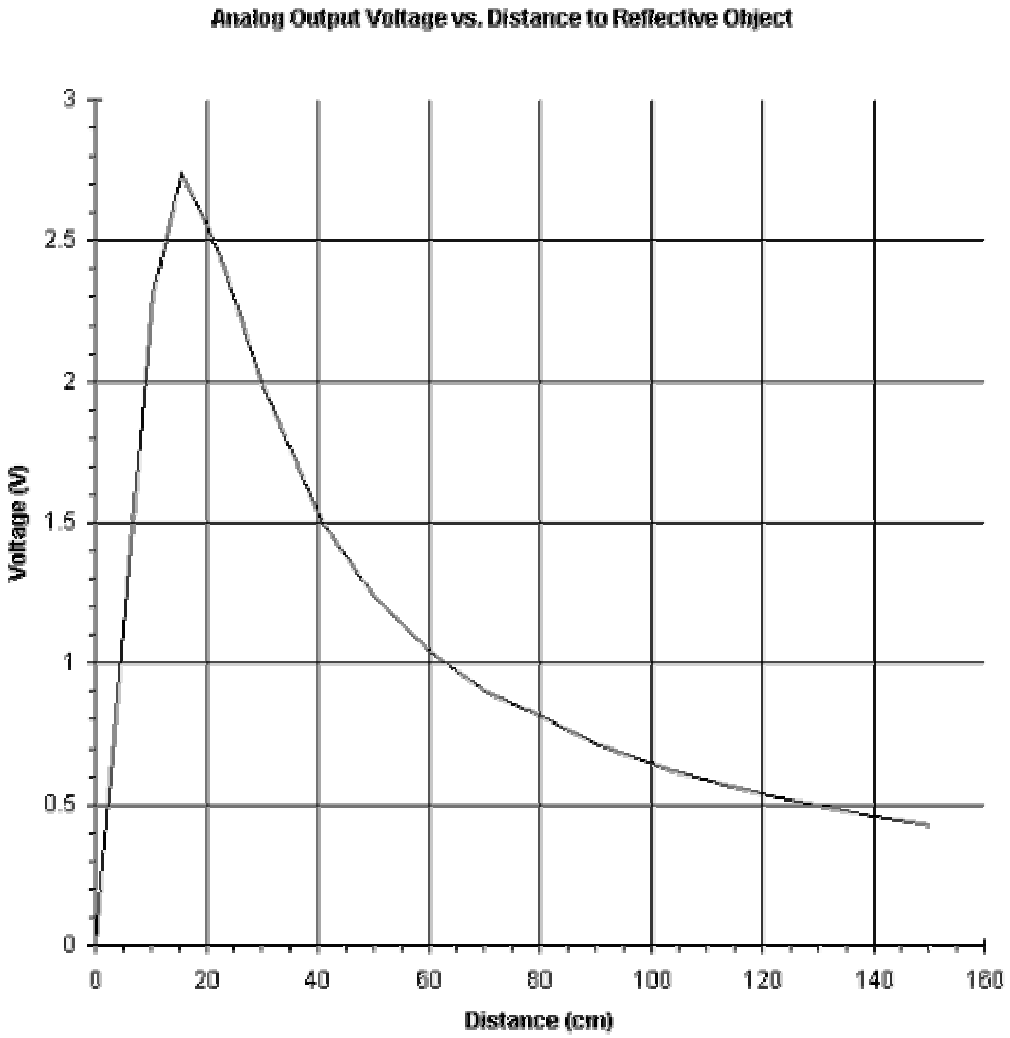



***Figure 3-10: Infrared proximity sensor voltage vs distance graph***

The sensor used (Figure 2-3, section 2.2.4) is a Sharp GP2Y0A02YK Long Distance Measuring Sensor. It uses triangulation to calculate the distance to the object of interest and so the distance received is independent of the reflectivity of the object and also the colour and texture of the object.

To ensure the 5 V supply to these sensors is as stable as possible a 7805 linear regulator has been used in conjunction with a 12 V line to produce a stable 5 V supply

with minimal noise. The outputs of these sensors are connected to the analogue inputs of the DAQ card.

#### 3.2.3.4. Laser Range Finder

This sensor (Figure 2-2, section 2.2.3) was developed by a previous master's student Shaun Hurd [Hurd, 2001]. It scans through 360 degrees by rotating the platform it is mounted on using a stepper motor. The stepper motor enables the ranger's orientation relative to the robot to be accurately controlled. The laser range finder can measure ranges from 0.5 metres up to 9.75 metres, independent of the object's colour or texture as it uses the triangulation method to obtain readings.

The range finder uses a laser beam, shone through a cylindrical lens generating a vertical beam that is captured by a CCD camera and analysed to establish the specific ranging data with an accuracy of ± 476 mm at 9.75 m and ± 50 mm at distances less than 3.2 m.

#### 3.2.3.5. Beacons

This sensor was developed to help eliminate the accumulative error associated with odometry encoders [Borenstein & Feng, 1996]. MARVIN passes the sensor, for that instant in time its exact position will be known and any odometry error can be reset.

The sensor system used is an infrared light barrier from Jaycar Electronics [www.jaycar.co.nz]. The sensor system is comprised of two Kemo B062E infrared receivers (Appendix B.3) and a slightly modified Kemo B062S emitter (Appendix B.4). The schematic of the original emitter can be seen in Figure 3-11.

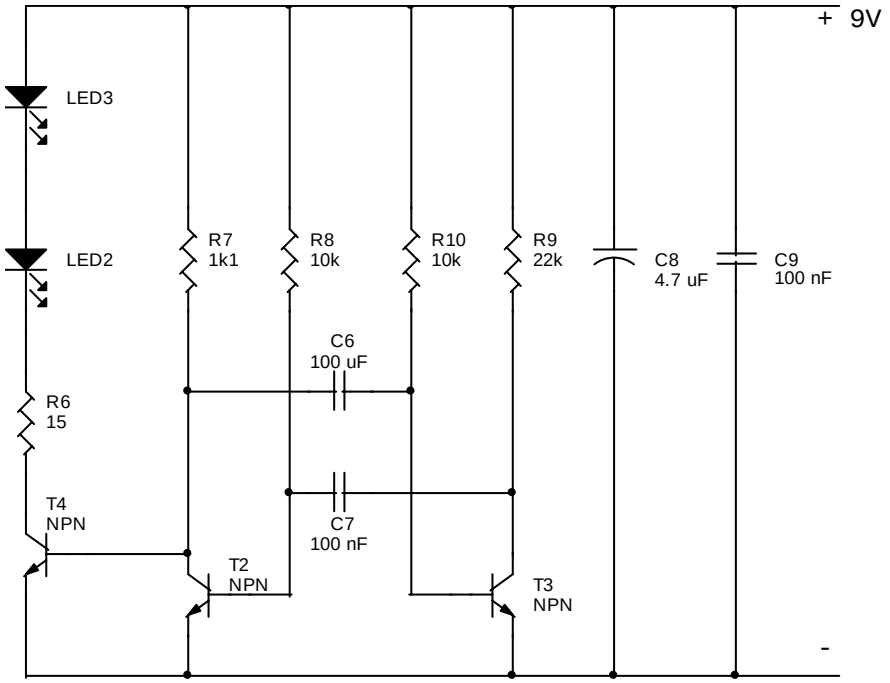



***Figure 3-11: Original infrared emitter schematic***

The system has a range of 18 metres with the emitter LEDs modulated at 14 kHz and the receiver photo diode fitted with an infrared filter to help block out ambient light. After initial testing, the system was found to be quite sensitive and needed to be altered. LED3 was removed from the schematic to limit the amount of IR light being emitted. Also R6 was changed from a 3 W, 15 ohm resister to a 2 W, 42 ohm resister to take this into account and permit less current to flow through LED2, further decreasing the sensitivity.

Once constructed, the emitter and receiver were housed in separate boxes (Figure 3-12). A switch was added to the emitter enabling it to be turned on and off along with a 9 V battery to provide the required power. The receiver has four wires coming from it, the yellow and black grouped together are for the 12 V supply and the orange and black are for the signal to the DAQ card. The signal is connected to a simple circuit similar to that used by the tactile sensors in Figure 3-9 to generate an appropriate signal for the digital ports on the DAQ card.

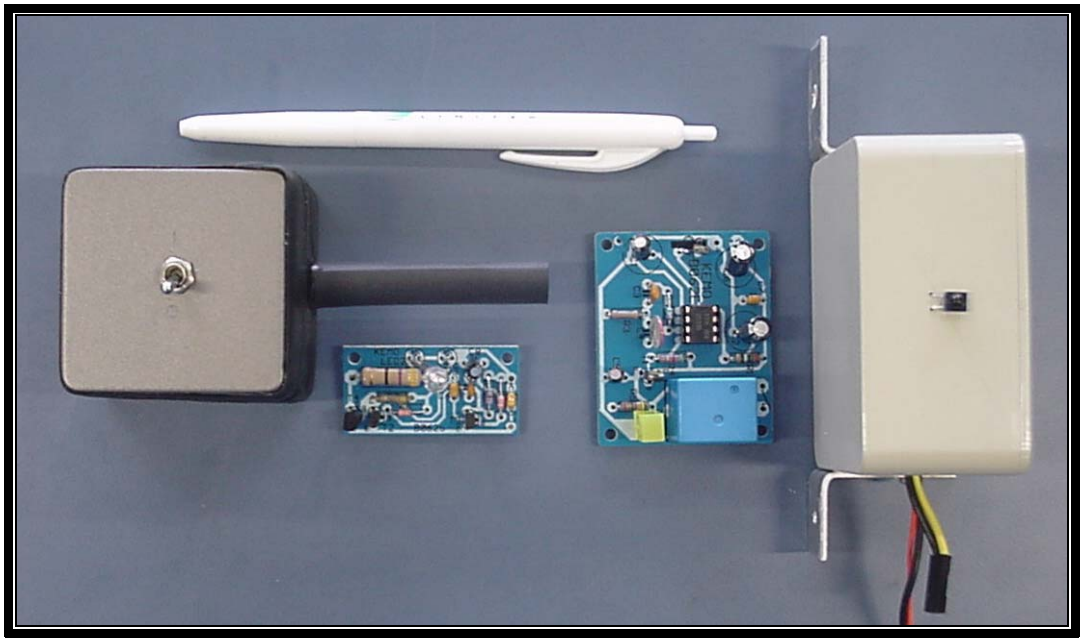


***Figure 3-12: Infrared light barrier emitter and receiver housed PCB's***

## 3.3. Mechanical Layout

In order to get MARVIN operational, certain mechanical aspects had to be slightly altered along with brackets and fittings to cater for the new PC and the array of sensors.

### 3.3.1. Overhaul of MARVIN

One of the first alterations made to MARVIN was to inflate the tyres which were half full of air. However this caused an increase in the device's height which meant the supporting casters needed to be lowered in order to rebalance the mechatron. This was achieved using washers to pack the casters, lowering them by the desired amount.

A Perspex cover was made to isolate the battery terminals from MARVIN's frame and stray conductive materials as shown in Figure 3-13. It also served the purpose of acting as a platform to mount the microcontroller and motor driver circuits as it was in an ideal location close to the motors. The circuits were fixed to the cover using plastic PCB mounts that went over the edges of the circuits and were stuck to the board using double sided tape, enabling the PCBs to be held firmly in place.

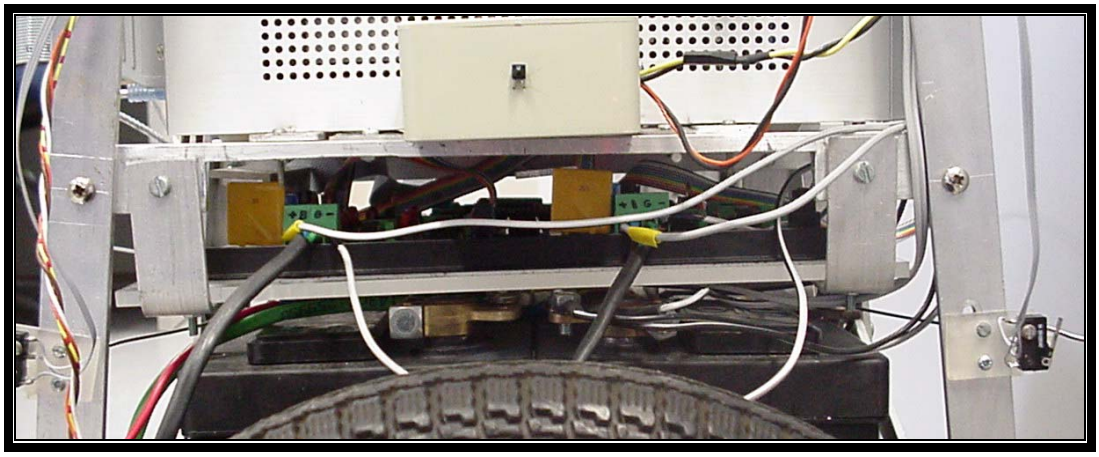


***Figure 3-13: Battery cover and motor driver platform***

Just above the motor drivers, a platform for the Shuttle xPC case was constructed. Two horizontal aluminium beams were attached to the two existing beams, enabling the PC casing to rest securely on top. Another aluminium beam went across the face of the case. It was attached to MARVIN's chassis and to the face of the PC, fixing it in place. This left the PC mounted roughly in the centre of the device's body, just above the batteries enabling it to be easily accessible to other parts of the robot. The position also enabled MARVIN's centre of gravity to remain low, helping with stability.

Another Perspex platform was constructed just above the PC as shown in Figure 3-14. This platform was used to mount the two DAQ card connector blocks and some sensor circuitry used to convert the sensor data into a form that could be interpreted by the DAQ card. A portion of the platform was cut to enable part of the 24 V ATX power supply to be fitted up the side of MARVIN's frame. The power supply was fixed to two horizontal beams that were in turn connected to the frame holding the power supply firmly in place.

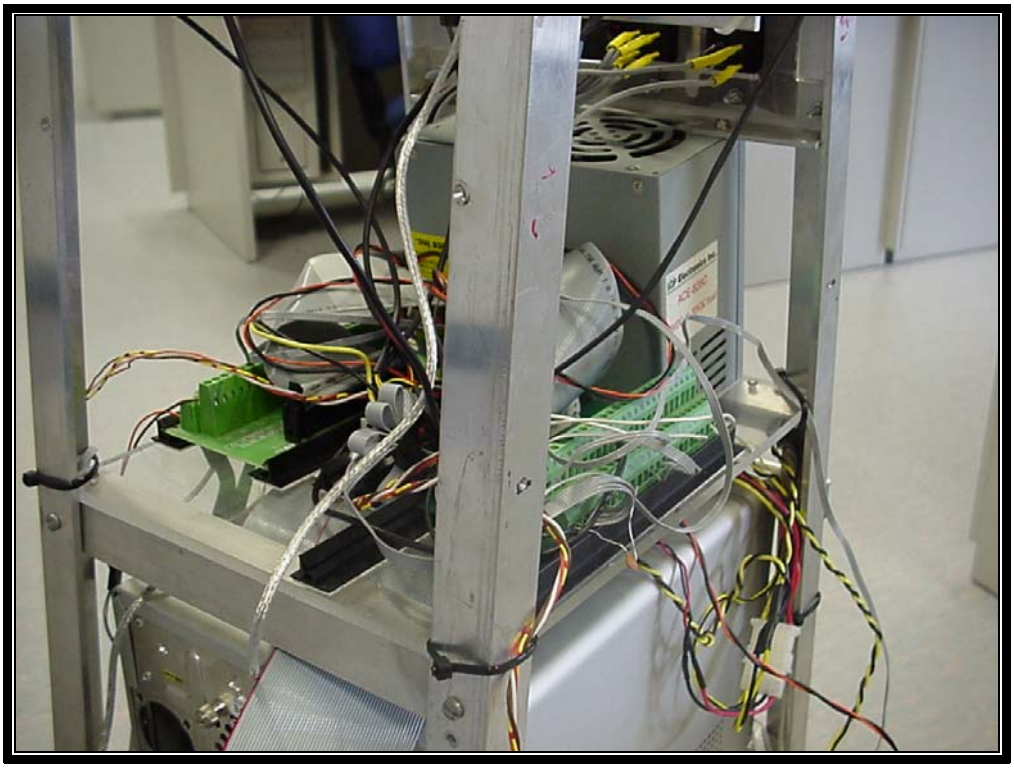


***Figure 3-14: DAQ card connector block platform***

Lastly an aluminium plate was attached just above the 24 V ATX power supply. It contained two switches that are used to turn the power to the motors and PC on and off as illustrated in Figure 3-15.

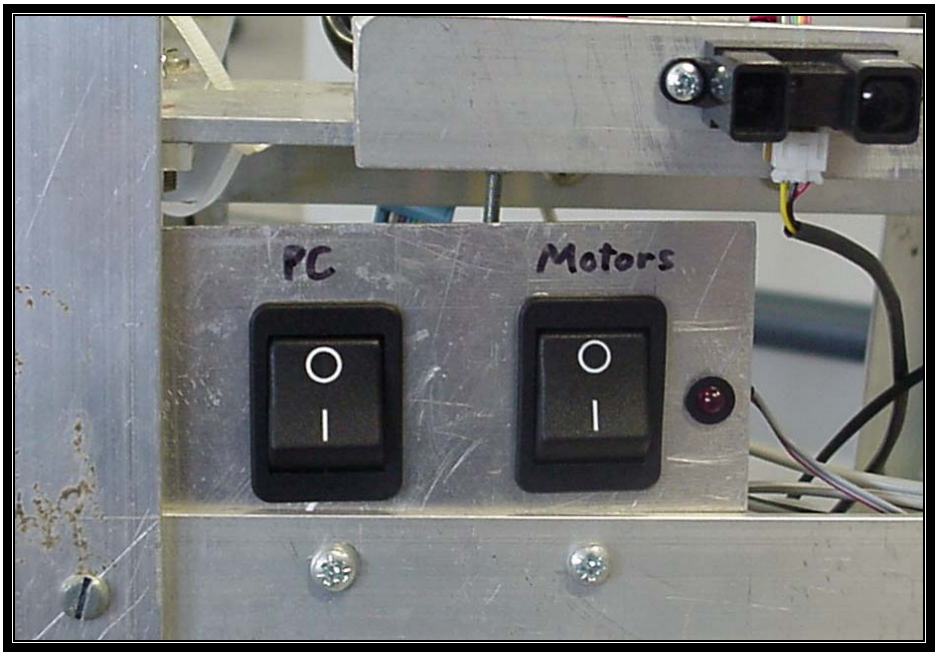



***Figure 3-15: Motor and PC switch panel***

## 3.3.2. Sensor Installation

### 3.3.2.1. Tactile Sensors

Four of these sensors were installed on the mechatron, with each switch positioned on each corner, front and back. Wires were installed on these switches to act as whiskers to detect when a wall or obstacle was encountered. Small Perspex guides were mounted just below the switches for the whiskers to rest on, helping eliminate any possible misalignment problems.

### 3.3.2.2. Infrared Proximity Sensors

These sensors were installed on the top of MARVIN's frame. Due to the frame tapering as it got higher, (away from the wheels) the 15 cm deadzone of the sensors was not a problem. In total, six of these sensors were installed in a configuration shown in Figure 3-16.

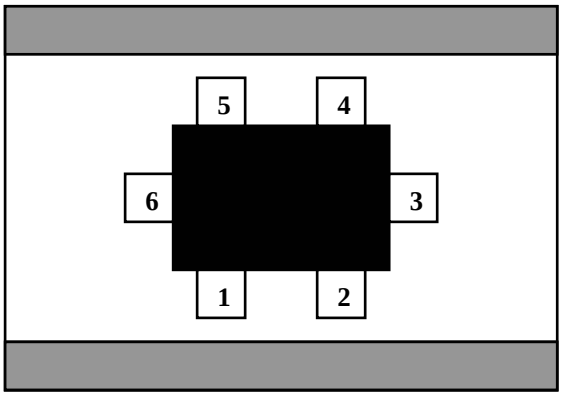



***Figure 3-16: Infrared Proximity Sensors configuration***

With two proximity sensors on each side of the device, heading information could be calculated in a corridor environment, along with distances to the walls. The side sensors were installed on slight angles (1°) away from each other to help prevent crosstalk. The intensity of the sensor vs off-axis angle was measured and is displayed in Figure 3-17. This indicated that the effect of sensor crosstalk should be minimal with this 1° offset.

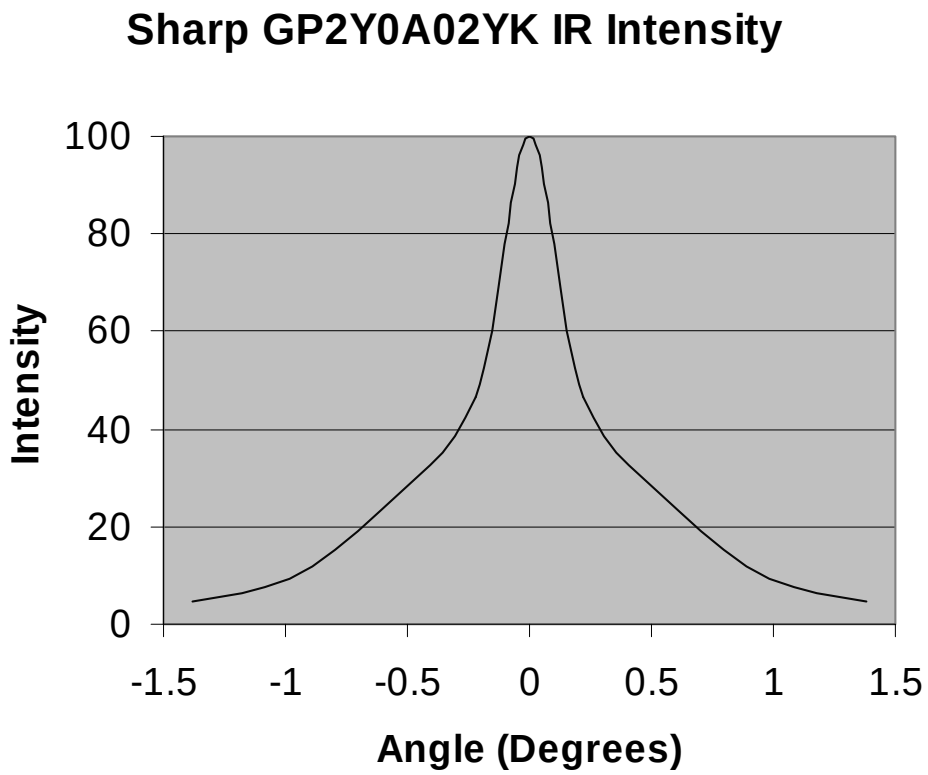



***Figure 3-17: Sharp GP2Y0A02YK IR intensity graph***

#### 3.3.2.3. Odometry Encoders

Both motors of MARVIN already had encoders fixed to them. These encoders comprised a code wheel that was fixed to the shaft of the motor and an optical encoder that was mounted to the motor housing. The encoder sat just below the code wheel enabling it to spin around generating the appropriate signals. As these encoders had no housing, stray bits of metal often affected the signal and they were prone to misalignment if bumped.

The encoders discussed earlier in this chapter replaced these. They were exactly the same as the previous kind except they had a plastic housing to hold the code wheel and optical encoder. Care had to be taken when mounting these encoders as they had to be accurately aligned to avoid scraping between the code wheel and encoder.

### 3.3.2.4. Beacons

As the beacons use infrared light as do the proximity sensors, the receivers were fitted in positions that were relatively far from these sensors to eliminate the possibility of crosstalk. The proximity sensors are mounted approximately 87 cm from the ground with the beacon receivers mounted approximately 40 cm from the ground.

The receivers are mounted on the same level as the PC. At this level, nothing from the structure of MARVIN could interfere with the infrared signal and there was an acceptable distance to the proximity sensors. The emitters are mounted 40 cm off the ground so they are in line with the receivers on MARVIN. The final layout of the device can be seen below in Figure 3-18.

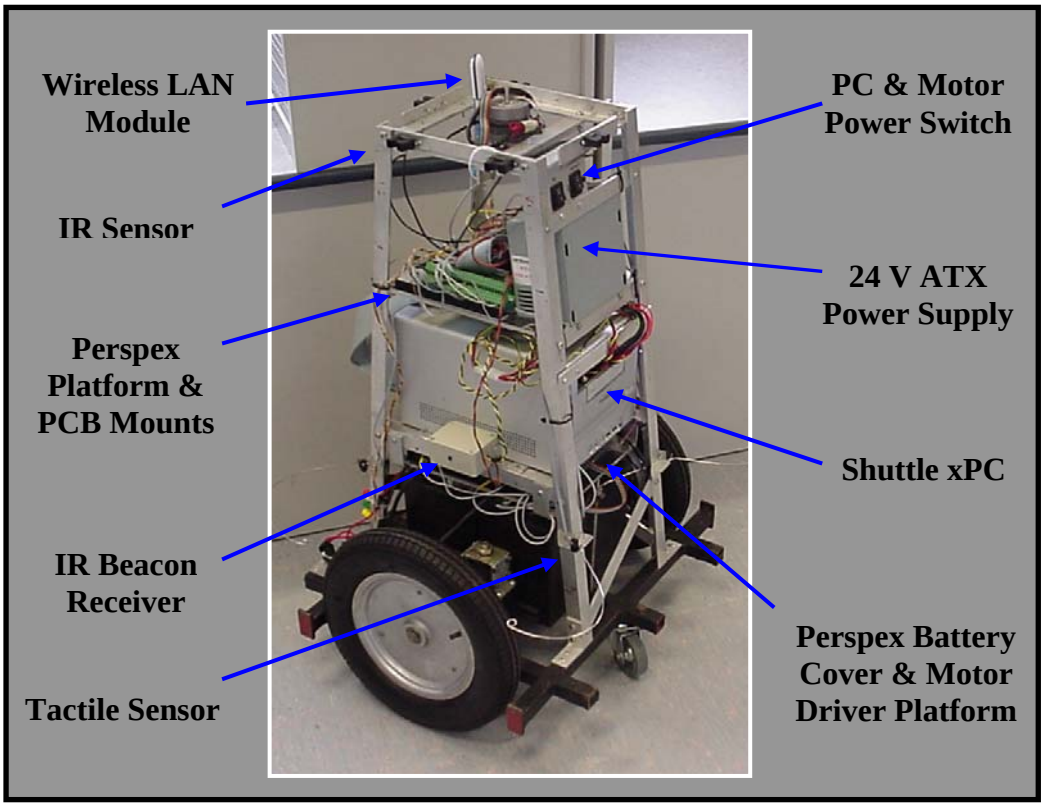



***Figure 3-18: Modified MARVIN***

# 4. SOFTWARE

## 4.1. Software Overview

As MARVIN utilises a complete PC, software packages like MATLAB and LabVIEW are able to be used. The use of a full PC motherboard gives further advantages of code portability and greater processing speeds. The navigation system software is written in MATLAB. This is a high-performance language for technical computing. The software package integrates computation, programming and visualisation in an easy to use environment, conveying results to solutions in a familiar mathematical notation. The name MATLAB comes from matrix laboratory, due to the software package's base coming from matrix manipulation.

The navigation software for this project forms the first attempt at getting MARVIN to conduct fully autonomous tasks. To be able to conduct the instructed tasks autonomously, the device must be able to move from a known start point, along a planned path to a known destination. During the journey the current location of the mechatron must be known at all times.

Three other masters projects were undertaken in parallel with this navigation project in order to achieve MARVIN's goal of full autonomy. These other projects include the humanisation of MARVIN which involves voice recognition and person identification, the development of a control algorithm and the implementation of generic motor drivers. The intended layout of the projects is illustrated in Figure 4-1.

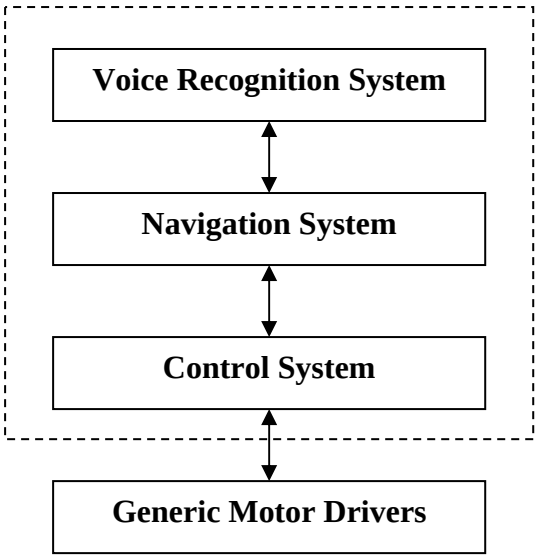



***Figure 4-1: Software system***

The humanisation of MARVIN including the voice recognition system is being developed by Ashil Prakash. A basic interface between the navigation system and voice recognition system has been developed, with only limited testing to date due to delays in the humanisation development (discussed further in section 6.2.2). The navigation system therefore uses a temporary front-end to generate the desired co-ordinates for MARVIN to travel which will be discussed later in this chapter.

The control system designed by Chris Lee-Johnson receives distance and other commands sent to it by the navigation algorithm. The instructions are interpreted and the appropriate velocity profile is generated as they are acted out by MARVIN. This will be touched on in more detail later in this chapter.

The last project which is indirectly related to MARVIN is the construction of generic motor drivers for the five large scale devices of the Mechatronics Group, by Craig Jenson. Due to delays in prototype development, motor drivers developed by fellow Mechatronics Group member and masters student, Andrew Payne are used, as discussed in section 3.2. Mircocontroller code was written by Chris and Andrew to enable instructions from the control system to be interpreted, generating the appropriate PWM signal to drive the motors.

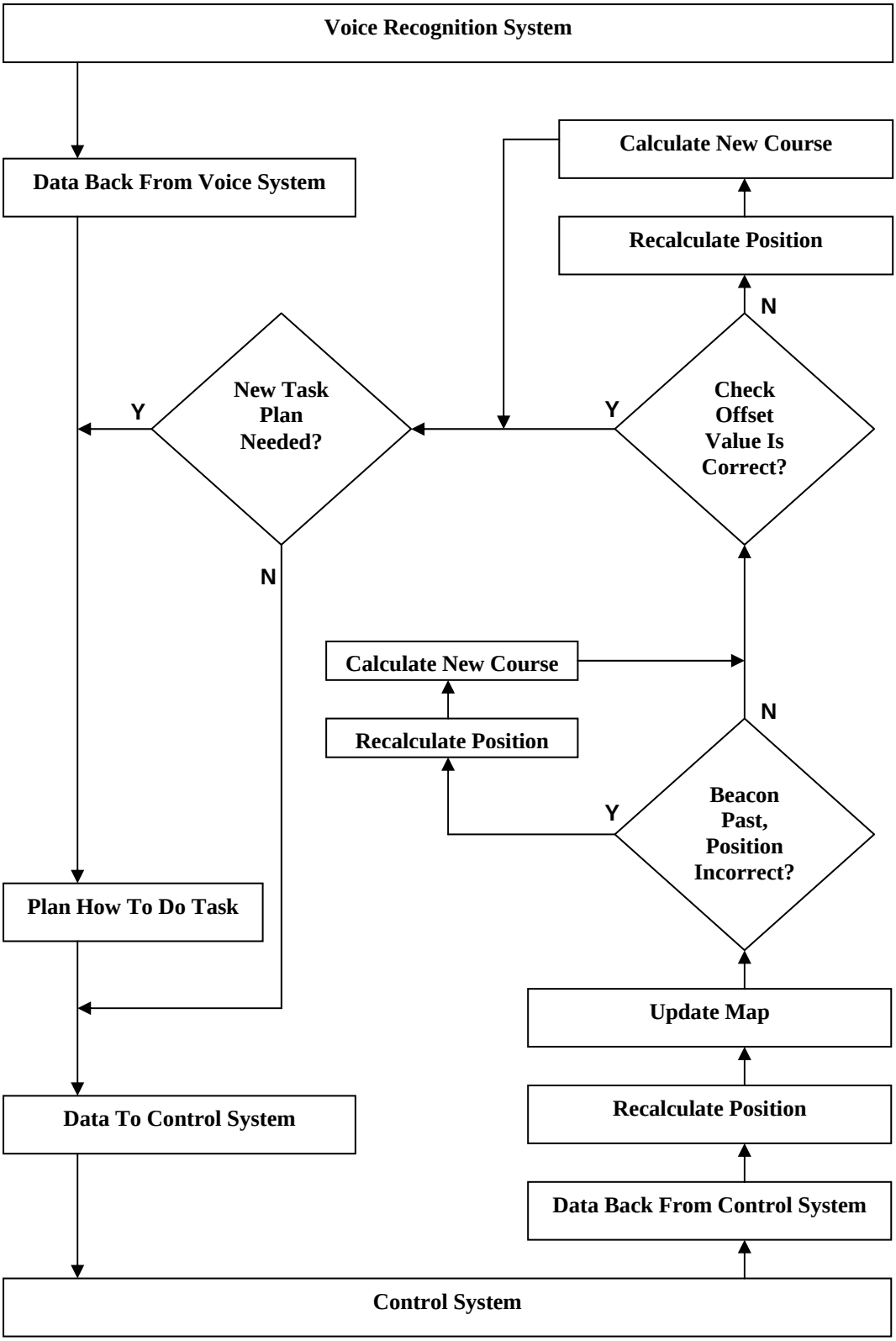



***Figure 4-2: Navigation system block diagram***

A detailed expansion of the dotted area in Figure 4-1 can be seen in Figure 4-2, showing the structure of the navigation system and its operation, including the intended interaction with the voice recognition and control system. Another navigation system was written for obstacle avoidance. Due to size constraints with respect to the width of the corridor walls of MARVIN's initial test environment (there was insufficient clearance for MARVIN to reliably negotiate past an obstacle), this algorithm was written independently of the main navigation routines and was tested instead in the University of Waikato's Electronics Laboratory. An overview of this algorithm can be seen in Figure 4-3 showing the operation of the code and how it interacts.

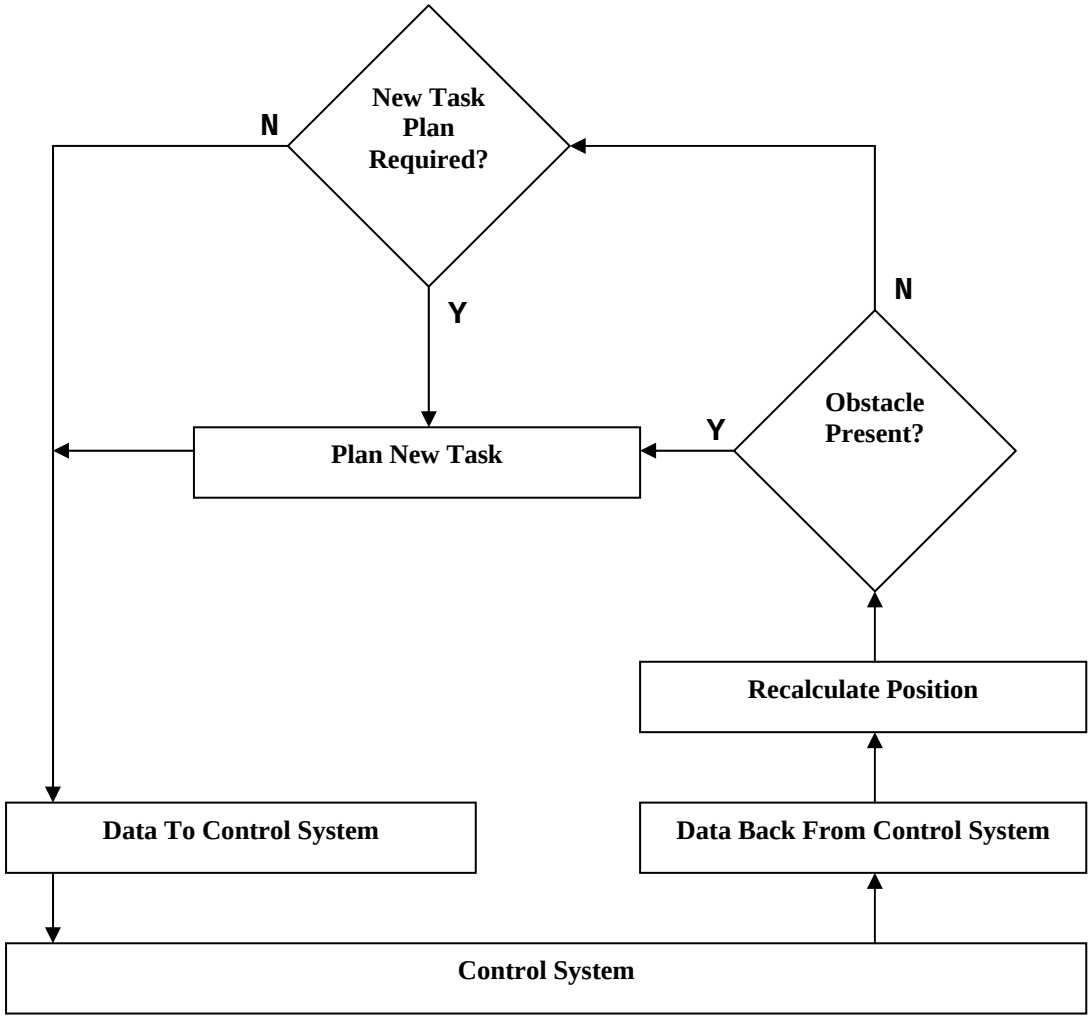



***Figure 4-3: Open space navigation algorithm block diagram***

## 4.2. Built-In Map

MARVIN's navigation system is based on a built-in map which is constructed from recorded measurements of the first floor of C Block's corridors. The measured environment is 12 × 33.6 metres. As MARVIN's base dimensions are 0.575 × 0.52 metres, the initial size of the grids were 0.6 × 0.6 metres giving a resolution of 20 × 56 grids. This enabled MARVIN to be represented by a minimum of one grid space. However, this lead to a poor depiction of the environment as the resolution was insufficient.

It was decided to alter the gird size to 0.3 × 0.3 metres which corresponds to a resolution of 40 × 112 grids. Now MARVIN is represented by a minimum of four grid spaces. This map was converted into a two dimensional array in MATLAB of the same size with a "1" representing a wall and a "0" used to represent an unoccupied grid.

A segment of the map that was loaded into MATLAB can be seen in Figure 4-4. The numbers on the side of the grid indicate the co-ordinates of each grid. The 0's inside the dark boundary lines (as described earlier) indicate that no wall or obstacle is present, whereas the brief scattering of 1's indicate a wall. Lastly the 2 that can be seen represents the position of a beacon. Figure 4-5 shows the environment after it has been loaded into MATLAB, with black representing areas not accessible to MARVIN, white representing areas that are able to be explored and the grey squares representing the positions of the beacons.

The map of the Electronics Lab was created in exactly the same way as the map of the corridors of C Block. The resolution of this map is 28 × 65 grids and was illustrated earlier in Figure 1-11 of section 1.4.

| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 | 13 | 14 | 15 | 16 | 17 | 18 | 19 | 20 | 21 | 22 | 23 | 24 | 25 | 26 | 27 | 28 | 29 | 30 | 31 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 2 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 3 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 4 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 5 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 6 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 7 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 8 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 9 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 10 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 11 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 12 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 13 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 14 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 15 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 16 | | | | | | | | | | | | | | | | | | | | | | | | | | | | | | |
| 17 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |
| 18 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 19 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 20 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 21 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 22 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 23 | 1 | 1 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 0 | 1 |
| 24 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 1 | 2 | 1 | 1 | 1 |

***Figure 4-4: Segment of MARVIN's map before being implemented in MATLAB***

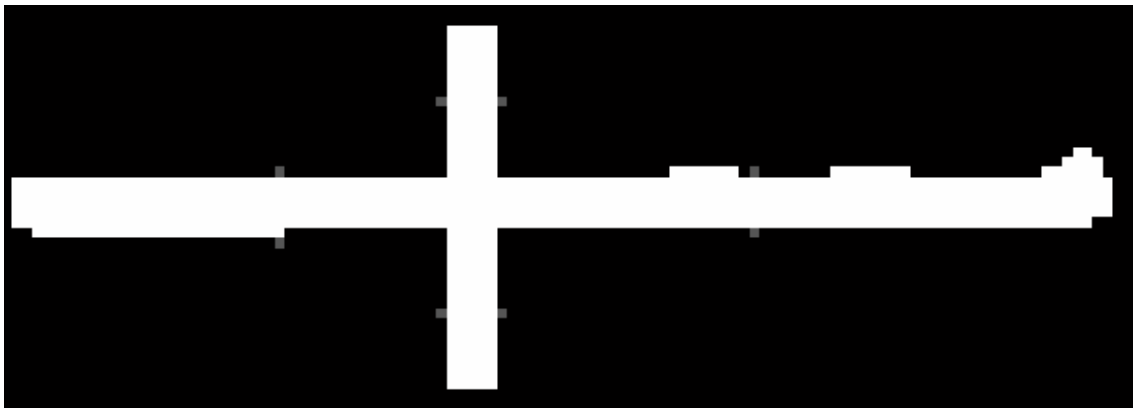


***Figure 4-5: MARVIN's map after being implemented in MATLAB***

# 4.3. Navigation Software

## 4.3.1. Front-end

The skeleton interface written between the voice recognition system and the navigation system uses string manipulation functions to detect key words. The specific key words passed to the interface function (called `Voice_Data()`) relate to destinations throughout the corridors of C Block. The string passed to the function contains two key words which are altered and refreshed upon a new instruction.

At present there are six main destinations. These are three offices, the graduate student's room and the electronics and physics laboratories. A function called `findstr()` is used in conjunction with the received string to identify and isolate the intended destination. The `findstr()` function returns a number that relates to the position of the key word in the string, so the target destination can easily be distinguished.

Once the string has been analysed, a series of checks are preformed to determine which of the six main destinations has been selected by the person talking to MARVIN. When this has been isolated and identified, the target coordinates are assigned relating to this destination. When these two programs are combined they will need to be multi-threaded, enabling both pieces of code to run concurrently (discussed further in section 6.2.2).

The front-end of the navigation system currently used to generate the coordinates for MARVIN to travel is a modified version of a graphical user interface (GUI) written by Chris Lee Johnson. By using a GUI as a front-end when testing the navigation algorithm, discrepancies between MARVIN's position and the position shown by the GUI can help with debugging and fine tuning.

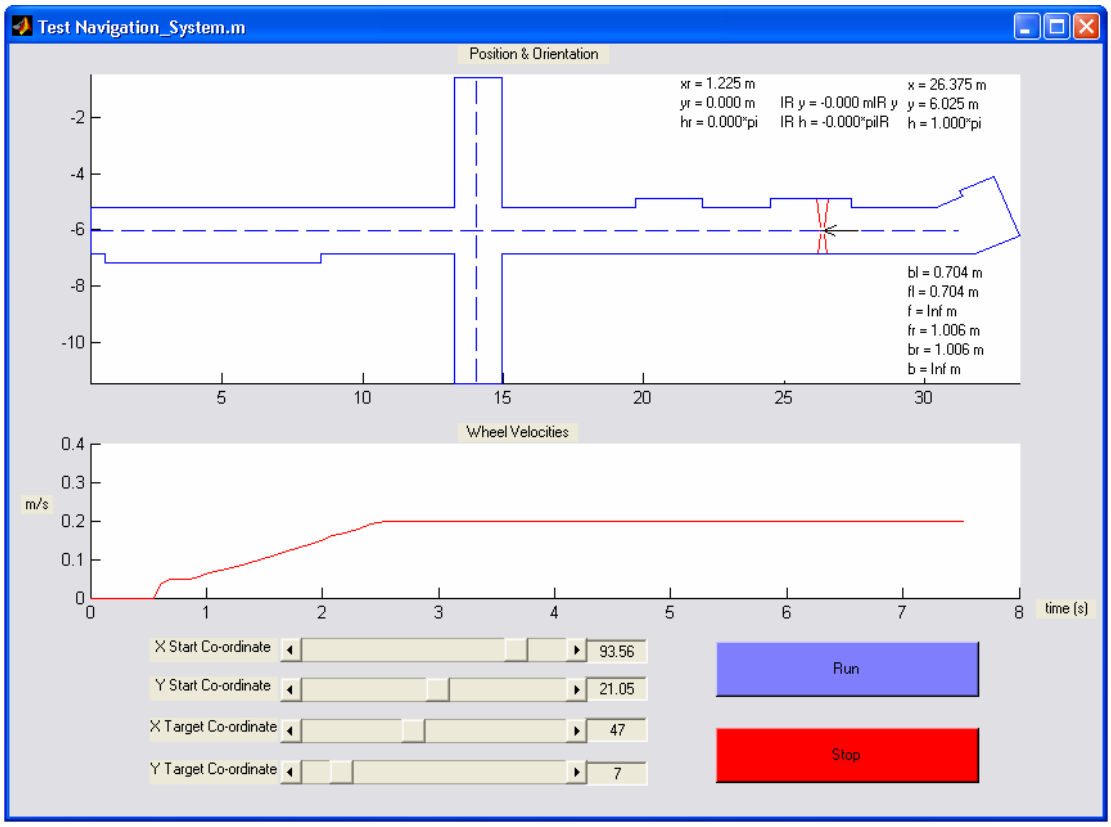



***Figure 4-6: Navigation GUI***

As can be seen from Figure 4-6, the GUI displays two graphs. The first represents the environment and where the sensor data believes MARVIN to be, with the second showing the velocity profile of the mechatron.

The first graph also displays key sensor information and data that is important to gauge how the device's journey is going. This is all displayed on the right hand side of the graph. The information contains the following:

- Relative distance in the *x* direction
- Relative distance in the *y* direction
- Relative heading
- Infrared offset distance
- Infrared heading
- Absolute distance in the *x* direction
- Absolute distance in the *y* direction
- Absolute heading

- Six infrared distances from each IR sensor

The sliders on the GUI enable the start and target coordinates to be entered into the navigation system. Due to the nature of incrementing the values on the slider, they are not always integer values which the coordinates are required to be. This problem is resolved by using the `floor()` function to round the floating point value down to the closest integer.

Once the coordinates are entered, the blue "run" button starts the code running and the "red" stop button acts as an emergency stop, preventing MARVIN doing any damage to the environment or its own structure and hardware. The navigation system is then called in a `while` loop until the start coordinates (which alter as the device moves) are equal to the target coordinates, indicating the assigned journey has been completed.

The data from the sliders and push buttons is read into the GUI code through callbacks. These are functions that are called by MATLAB when an event occurs. Once the coordinates have been entered and the run button is pressed, certain variables are initialised along with the two graphs. After the initialisation, a `while` loop is entered where the navigation system is called along with the updating of the graphs. An `if` statement checks to see if the "stop" button has been pressed which alters a command sent to the navigation system, stopping MARVIN instantly. The conditions for the `while` loop to be entered are that the "run" button must have been pressed along with the start and target coordinates not being equal. This is similar to the condition in the skeleton voice recognition interface, except in this case only the start and target coordinates have to be equal for the `while` loop to finish.

### 4.3.2. Navigation System Function

This function forms the heart of the software system and is responsible for navigating MARVIN around the environment. Persistent variables are used to store and remember data for the next cycle through the function, enabling positional and other

information to be always known. They are equivalent to static variables in C. There are two main difficulties in using these variables:

- They cannot be declared and initialised at the same time
- They cannot be returned from a function

These two problems were avoided in different ways. An initialise flag is passed to the system the first time the function is called. The various persistent variables are then initialised inside an `if` statement that checks the value of the flag. This enables the persistent variables to be initialised whenever this flag goes high, which generally occurs during the first cycle of the software system. Additionally, during this initialisation sequence the map is created and MARVIN's initial heading is calculated. The heading calculation assumes the device is facing towards its intended target location. This will be altered when a compass is fitted to the mechatron enabling this condition to be relaxed. The second problem is solved by equating the persistent variable to be returned as another variable.

Being the heart of the navigation algorithm, the sub-functions that are required to help the device autonomously navigate are called from this routine. During a cycle of this main function, not all the sub routines are called as some rely on flags to be set high, enabling them to be entered only when required, decreasing the program cycle time.

### 4.3.3. Control System Interaction

As the control system and navigation system are interdependent, the variables passed back and forth between each program are vital to their success and operation. The data returned to the control system is mainly sensor related and are shown in Figure 4-7.

- **new_time** – A variable used for display purposes in the GUI code.
- **Abs_x_Metres** – The absolute distance in metres travelled in the x direction.
- **Abs_y_Metres** – The absolute distance in metres travelled in the y direction.
- **Abs_Heading** – The absolute heading of MARVIN.
- **Rel_x_Metres** – The relative distance in metres travelled in the relative x direction.
- **Rel_y_Metres** – The relative distance in metres travelled in the relative y direction.
- **Rel_Heading** – The relative heading of MARVIN.
- **tgt_x** – Where the device should be in metres in the x direction, for display purposes.
- **tgt_y** – Where the device should be in metres in the y direction, for display purposes.
- **tgt_heading** – Heading MARVIN should be facing, for display purposes.
- **w_vel** – A 2 element array containing the wheel velocities.
- **ir_y** – IR sensor offset distance from the middle of the corridor in metres in the relative y direction.
- **ir_Heading** – MARVINs heading derived from the IR sensors.
- **ir_obj_dist** – A six element array containing each IR sensors current value in metres.
- **contact_switch** – A four element array containing 0's and 1's indicating if the switches have been pressed.
- **Beacon** – A two element array containing 0's and 1's indicating if a beacon has been passed.

***Figure 4-7: Variables returned from the control system***

Not all the values passed are used by the navigation system for localisation. All the variables that are returned are also returned by the navigation system to the GUI so they can be displayed, helping with debugging as discussed in the previous section. Furthermore, the values of all the returned variables are stored in a text file, enabling a log of all the data generated during a journey through the environment to be recorded. The logged data can be used at a later date to re-create MARVIN's expedition through the corridors of C Block.

As the `Abs_x_Metres` and `Abs_y_Metres` give the absolute distances the device has moved through, these are used to recalculate the coordinates of MAVIN each cycle through the navigation system. This occurs inside the function `Data_Update()`. The code to calculate the coordinates can be seen on the following page:

```
Start_x = ceil((Abs_x_Metres + x_Origin_Metres) / 0.3);
Start_y = ceil((Abs_y_Metres + y_Origin_Metres) / 0.3);
```

where `Abs_x_Metres` and `Abs_y_Metres` are explained earlier in Figure 4-7 and `x_Origin_Metres` and `y_Origin_Metres` are distance values in metres from the origin of the map, to where the device initially started. By dividing the sum of these values by 0.3 metres, the grid MARVIN is currently in for that particular axis will be known. The `ceil()` function rounds the result upwards to the nearest whole number.

The variables passed to the control system are shown in Figure 4-8.

- **New_Instruction_Flag** – Indicates if a new instruction has been sent.
- **Distance** – Distance in metres for MARVIN to travel.
- **Angle** – Angle for MARVIN to turn through.
- **Offset_Angle** – Difference between actual angle turned through and expected angle turned though.
- **IR_Weighting** – A 2 element array giving weighting values for the left and right IR sensors.
- **Corridor_Offset** – Distance from MARVIN to the middle of the corridor after turning.
- **Corridor_Angle** – Indicates new part of the corridor has been entered.
- **Wall_Distance** – A 2 element array containing distances to the walls in metres from the centre of the corridor.
- **Init_x** – Initial starting point of MARVIN in the x direction, in metres.
- **Init_y** – Initial starting point of MARVIN in the y direction, in metres.
- **Init_Heading** – Initial heading of MARVIN.

***Figure 4-8: Variables passed to the control system***

The function `Control_Feedback()` is used to calculate the `Wall_Distance` and the `IR_Weighting` values. As MARVIN tries to stay in the middle of the corridor which has non uniform wall distances, this function checks to see what grid space the device currently occupies. If the mechatron is approaching a part of the environment where a change in wall distance is going to occur, the `IR_Weighting` is lowered and the `Wall_Distance` variable is given the appropriate values. This

ensures the control system won't inadvertently try to alter MARVIN's course by trying to keep the mechatron in the middle when a temporary change in the width of the corridor has occurred.

The remaining variables passed to the control system are calculated in other functions further into the navigation system, these will be discussed later in this chapter.

### 4.3.4. Task Planning

The start and target coordinates entered into the main function determines which part of the `Task_Plan()` function is executed. This particular function is only entered if the `New_Task_Flag` is set high, indicating a new task has to be planned for the mechatron and is reset upon entry. Using this flag to limit entry into the function prevents the problem of a new task being calculated for the device every control cycle. Additionally, upon entering this function the `New_Instruction_Flag` is set high, indicating to the control system a new instruction is being sent.

The code written in this function breaks any journey required of MARVIN into small manageable segments, making it easier to debug and implement. These generated tasks enable the mechatron to travel anywhere in the initial environment. The segments are made up of straight lines and right angle standing turns. Standing turns were initially used in preference to integrated turns (see section 6.2.3) while testing the navigation algorithm. The difference between these two forms of turn can be seen below in Figure 4-9.

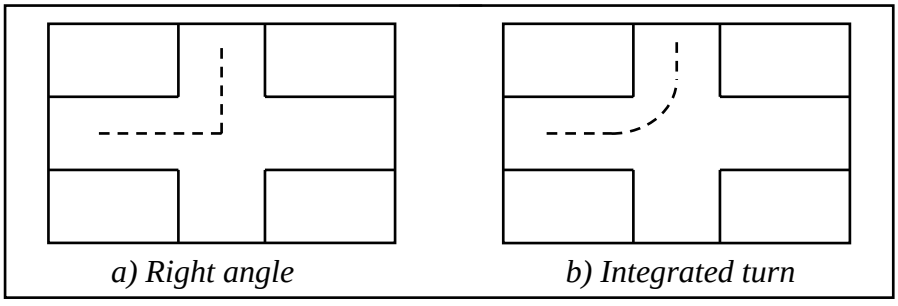



***Figure 4-9: Right angle and integrated turns***

The straight line distances are calculated using these two lines of code, depending on which axis is being considered.

```
Distance = abs(Start_x - Target_x) * 0.3;
Distance = abs(Start_y - Target_y) * 0.3;
```

By calculating the difference between where MARVIN is starting and where it wants to go, the difference in coordinates is found. As each grid is 0.3 m × 0.3 m, the multiplying factor of 0.3 converts the difference in coordinates to a distance value in metres, indicating how far the device is to travel. By setting the angle to zero when the distance command is passed to the control system, MARVIN will travel in a straight line.

The convention used by the control system in regard to turning direction is that a positive angle indicates turning to the right, whereas a negative angle indicates turning to the left. By passing a distance of zero with the particular angle, a standing turn will be performed in the desired direction. For an integrated turn to occur, a non zero angle and distance must be received by the control system.

Once a turning instruction has been formulated, the `Corridor_Angle` variable is assigned a value depending on where MARVIN is to travel. This variable is sent to the control system, indicating a new part of the corridor has been entered. The coordinate system is then altered so the control system can relate the infrared sensor values with the new wall positions. The direction that relates to each `Corridor_Angle` value is illustrated below in Figure 4-10, with respect to the orientation of the environment as shown in Figure 4-5.

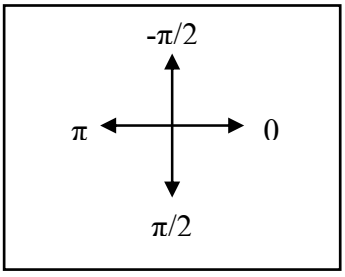



***Figure 4-10: Corridor angle with respect to direction***

The values stored in this variable are used in other parts of the navigation system to give an approximate indication of the heading of MARVIN. The absolute angle of the device is calculated from the line of code below:

```
Abs_Angle = adjust_angle(Abs_Heading + Angle);
```

where `Abs_Heading` is the mechatron's current heading as returned from the control system and `Angle` is the intended angle for the device to turn through. The function `adjust_angle()`was written by Chris Lee-Johnson and ensures that when two angles are added together, they stay within the range $-\pi$ to $\pi$.

As well as planning tasks in order to move MARVIN around the environment, this function checks to see if the `Off_Course_Flag` has gone high. If this is the case a course is calculated to get the device back on its intended path. A diagram showing a possible recalculated course can be seen in Figure 4-11.

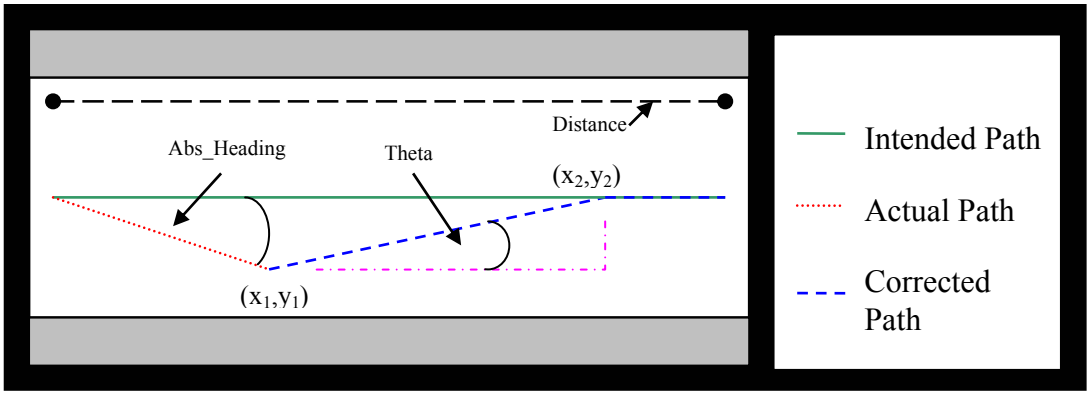



***Figure 4-11: Correcting MARVIN's course***

When MARVIN has been found to be off-course, the device is stopped and inside the `Task_Plan()` function, the angle needed to turn through in order to be facing the correct direction to get back on course is calculated. By knowing MARVIN's current position $(x_1,y_1)$ and where MARVIN must go $(x_2,y_2)$ as illustrated by Figure 4-11, the first part of angle can be found from the following code segment:

```
theta = adjust_angle(atan2((y2 - y1),(x2 - x1)));
```

where the `adjust_angle()` function was discussed earlier and the `atan2()` function performs a four quadrant inverse tan operation on the values passed to it. The relative *x* and *y* values returned from the control system are equated to $x_1$ and $y_1$ respectively and $x_2$ is made equal to the variable `Distance` that MARVIN was instructed to travel minus 0.5 metres. The 0.5 metres is subtracted as it enables half a metre for the course of the device to be altered again, so that the mechatron can be facing the direction it was originally and to be in the intended place for when the next instruction is calculated.

The last part of the angle to be calculated uses the piece of code below:

```
Angle = adjust_angle(theta - Abs_Heading);
```

where the `adjust_angle()` formula was discussed earlier and the angle `theta` was calculated above. `Abs_Heading` is the current angle the device is facing as returned from the control system.

Once the angle has been turned through, the distance to travel is calculated. As the distance of the vertical and horizontal magenta lines in Figure 4-11 are known, Pythagoras's theorem can be used to determine the diagonal distance of the blue line for MARVIN to travel in order to get back in the middle of the corridor. The code below is used to find this distance:

```
Distance = Distance - Rel_x_Metres - 0.5;
Travel_Distance = sqrt(Distance^2 + Rel_y_Metres^2);
```

where the `Distance` variable is equal to the horizontal magenta line and `Rel_y_Metres` is equal to the vertical magenta line. After this distance has been travelled, the device is instructed to face down the corridor, following the path of the blue line enabling MARVIN to continue on its intended course and complete the original set instruction.

### 4.3.5. Beacon Filtering and Correction

Due to the fact that an infrared light source and receiver have been used, filtering needs to occur on the received beacon data to eliminate positional errors. As the purpose of the beacons is to eliminate the accumulative error associated with odometry, the position calculated from the beacons needs to be as accurate as possible. The problem with using infrared light can be illustrated below in Figure 4-12 with the width of the beam exaggerated for clarity.

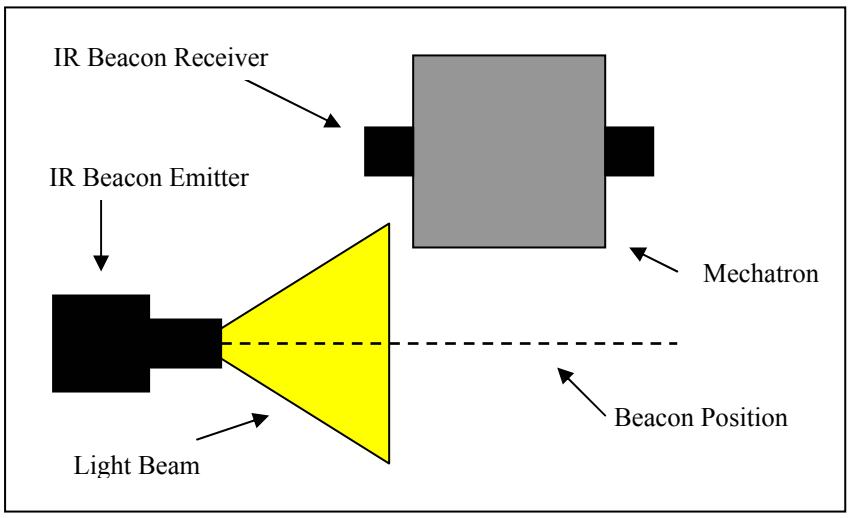



***Figure 4-12: Beacon problem illustration***

As the beam of light disperses, it gets broader, leading to the mechatron detecting the beacon sooner than intended. An infrared shielded cone is attached to the front of the emitter to help focus the light, limiting this effect. This focussing method only partially solves the problem; software filtering has to occur to limit it further. An additional focussing technique is to use a convex lens to project the emitted light into a parallel beam. As the cone attached to the receiver was sufficiently effective (section 5.3) this technique was not further explored.

When the `Beacon` value returned from the control system goes high, indicating a beacon has been passed, the distance MARVIN has travelled is recorded inside the function `Beacon_Filter()`. When the opposite happens and the `Beacon` value goes low, the distance travelled is recorded again. This generates two distance values

corresponding to the start and end of the light beam. The average of these two values gives the approximate distance travelled by MARVIN when the device passed the middle of the beacon. The code to achieve this can be seen in Figure 4-13.

```
if Corridor_Angle == 0 | Corridor_Angle == pi

      Beacon_Dist = (First_Dist + (Abs_x_Metres + x_Origin_Metres))/2;

elseif  Corridor_Angle == -pi/2 | Corridor_Angle == pi/2

      Beacon_Dist = (First_Dist + (Abs_y_Metres + y_Origin_Metres))/2;
end
```

***Figure 4-13: Beacon distance code***

The value of the `Corridor_Angle` variable is checked to see which part of the corridor MARVIN is in so the appropriate `Beacon_Dist` value is calculated. The variable `First_Dist` is the distance value recorded when the beacon was first detected.

Once the filtering of the beacon distance has occurred, a check is made inside the `Beacon_Check()` function to see if MARVIN is in the grid or neighbouring grids of the beacon. This eliminates the chance of any possible false triggered signals being misinterpreted as a beacon. The check also helps determine which beacon was detected due to the location of each on the map being known. A sample of the function can be seen below in Figure 4-14.

```
if (Start_x >= 26 & Start_x <= 30)

     Correction = 8.3 - Beacon_Dist;
     x_Correction = Correction;
     New_Instruction_Flag = 1
     Distance = (Distance - Correction) - Rel_x_Metres;
     Offset_Angle = adjust_angle(Abs_Angle-Abs_Heading);

 elseif (Start_y >= 9 & Start_y <= 11)
```

***Figure 4-14: Beacon correction factor code***

Once the beacon has been identified, a correction factor is calculated. As the beacons are located at fixed values, the calculated `Beacon_Dist` value as mentioned previously, is subtracted from the fixed value to find the correction factor. Additionally, a new distance for MARVIN to travel is calculated to eliminate the accumulated error with the setting of the `New_Instruction_Flag`. The `Offset_Angle` variable which was also calculated will be discussed further in section 4.3.7. The three generated correction factors are eventually passed to the `Data_Update()` function. Their use is illustrated below in Figure 4-15.

```
Rel_x_Metres = Rel_x_Metres + Correction;
Abs_x_Metres = Abs_x_Metres + x_Correction;
Abs_y_Metres = Abs_y_Metres + y_Correction;
```

***Figure 4-15: Applying the beacon correction factor***

As the `Rel_x_Metres` variable indicates the forward (or reverse) distance travelled by MARVIN independent of the *x* or *y* direction (if the corridor angle has been set) the `Correction` variable can be added to it, eliminating the accumulative odometry error associated with that variable. The `Abs_x_Metres` and `Abs_y_Metres` variables have the `x_Correction` and `y_Correction` factors applied to them respectively, as these values are specific to the *x* or *y* direction. Four beacons have been positioned around the environment; their positions are indicated by the grey squares shown earlier in Figure 4-5.

### 4.3.6. Off Course Checking

The purpose of this function is to determine if the device is off course. As the `Rel_x_Metres`, `Rel_y_Metres` and also the corresponding `Abs` values are calculated in the control system by fusing the various sensor data together, they give a good indication of MARVIN's position. The `Rel_y_Metres` variable shows the offset of MARVIN from the middle of the corridor and it can be used to check if the device has drifted to either side while conducting a task.

As the intention is to keep the mechatron in the middle of the corridor, with the narrowest points being 0.84 metres (from the middle of the corridor) and with MARVIN being 0.575 metres wide, a critical offset distance of 0.35 metres was decided upon. This will leave the device approximately 0.253 metres away from the wall if the mechatron drifts this amount. Figure 4-16 shows the code used to determine if MARVIN is off course.

```
if abs(Rel_y_Metres) >= 0.35 & Off_Course_Flag == 0

    Stop_Flag = 1;
    Off_Course_Flag = 1;

end


if abs(Rel_y_Metres) < 0.35 & Off_Course_Flag == 1

    Off_Course_Flag = 0;

end
```

***Figure 4-16: Off course checking code***

Once the device has been found to be off course, a variable `Stop_Flag` is set high. This will be detected during the next cycle of the navigation system, causing the appropriate variables to be set, enabling MARVIN to stop. The `Off_Course_Flag` is also set high. This variable will be sent to the `Task_Plan()` function as discussed earlier, allowing a new course to be plotted for the device.

The second `if` statement of the function resets the `Off_Course_Flag`. This occurs once the device has moved to a position that is less than the critical offset distance, ensuring MARVIN isn't found to be off course again.

### 4.3.7. New Task Checking

Before a new task can be calculated for MARVIN to travel, a check has to occur to ensure the previous task has been completed. The `New_Task_Check()` function ensures the instructed angle has been turned through and the intended distance to travel has been traversed.

During the first section of the function, a verification is done on the distance travelled to check if it has fallen short, been met, or overshot. The conditions for each check are similar but have slight variations to help determine each situation. In the case of the first two, the criteria are for the both wheels to be stationary, indicating the device has come to a stop. The second check relates to the difference between the distance set to travel and the distance travelled by MARVIN. Values greater than 0.3 metres indicate the device has fallen short, causing a new distance to be calculated to finish the task. If the difference is less than 0.3 metres, the task has been completed successfully, setting the `New_Task_Flag` high for the next task to be calculated later in the program.

In the last situation, to determine if the device has overshot the intended distance, both wheels are checked to see if they are still moving. The last part of the check relates to the difference between the distances. If this is less than -0.15 metres, the device is instructed to stop by setting the `Stop_Flag` high as discussed earlier. In all three of these situations, the variable `Turning_Flag` is part of the checks, it is set in the `Task_Plan()` function to 0 for straight instructions and 1 for turning instructions. Using this variable helps to differentiate between the two.

The second part of the function checks to see if the angle has been turned through. It is comprised of two levels of checks. The first level makes sure the wheels have both stopped, the `Turning_Flag` is equal to one and a variable `Delay_Flag` has gone high. The purpose of having a delay flag is to ensure the wheels have had a chance to move, otherwise the first level of checks would be meaningless.

The second level of checks makes use of a function written for the control system. The function `is_inside_arc()` requires four variables to be passed to it. These are the angle the device wants to face and two error values that are equal to ± π/2 respectively. The last value is the angle MARVIN is facing. If this is within the error band the function returns a zero, indicating the device has turned through the intended angle. If this is not the case, then the remaining angle for device to turn is calculated and sent to the control system to be executed.

Upon turning through the correct angle, the `Delay_Flag` is reset and the `New_Task_Flag` is set high, with the `Corridor_Offset` and `Offset_Angle` to be passed to the control system being calculated as well. The `Corridor_Offset` indicates the device's offset distance from the middle of the corridor's centre axis once the turn has been completed with MARVIN entering a new section of corridor. The `Offset_Angle` is a non zero angle that gives an initial heading error. The control system uses this angle to make small heading adjustments, resulting in smoother motion. An extra benefit of this variable is that if the angle instructed of MARVIN is too small to turn though, the `is_inside_arc()` function will assume the angle has been turned through and the `Offset_Angle` will be calculated. When the device's next instruction is carried out, a small integrated turn will be preformed as the mechatron moves. The calculation of these two variables can be seen in Figure 4-17.

```
Offset_Angle = adjust_angle(Abs_Angle-Abs_Heading);

if Corridor_Angle == pi/2 | Corridor_Angle == -pi/2
    Corridor_Offset = y_Corridor_Middle - Abs_x_Metres;
elseif Corridor_Angle == pi | Corridor_Angle == 0
    Corridor_Offset = x_Corridor_Middle - Abs_y_Metres;
end
```

***Figure 4-17: `Offset_Angle` and `Corridor_Offset` calculations***

# 4.4. Obstacle Avoidance Navigation Software

## 4.4.1. Front End

The interface to the obstacle avoidance code is another GUI similar to the one used by the navigation software discussed earlier. As this software is separate to the corridor navigation system, it is independent of the corridor map.

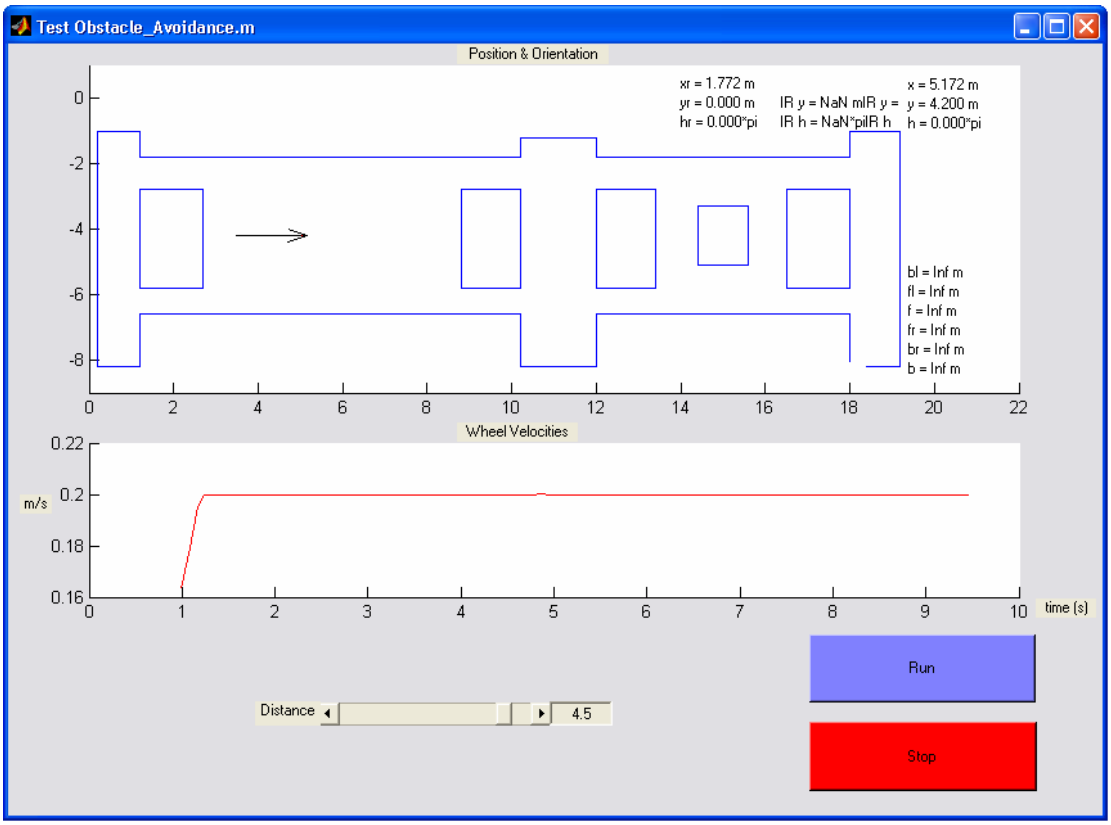



***Figure 4-18: Obstacle avoidance GUI***

The GUI as seen in Figure 4-18 contains two graphs. The first graph shows the dimensions of the open environment inside the Electronics Laboratory that is used to avoid obstacles. The open area used is 6.1 m × 4.7 m. The second graph shows the velocity profile of the device.

The inputs from the GUI have the same run and stop buttons as previously but now have only one slider. The input from the slider is a distance value in metres to travel.

The GUI code will call the obstacle avoidance program until the device has travelled the straight distance set by the slider.

### 4.4.2. Obstacle Avoidance and Detection

The layout of the code structure for this program is very similar to the corridor navigation layout, with persistent variables again being made use of. Similar functions and methods are used as previously mentioned in this chapter to negotiate MARVIN around obstacles and back onto the device's original course. The original task set by the GUI will instruct the mechatron to conduct a straight line movement. The program will only plot another course and be interrupted if MARVIN stops due to an obstacle being detected.

If an obstacle is detected in front of the device, the control system stops MARVIN. Inside the function `Obstacle_Detection()`, a check is made to see if the mechatron is stationary. The wheel velocities have to be zero and the distance reading of the front IR sensor less than 0.6 metres. If this is the case, a further check is made to see if the obstacle is there after 2 seconds, to ensure it is still in place and wasn't moved, or if a human moved by its own accord.

With the object still blocking MARVIN's path after the designated time, the side IR sensors are checked to see if the device is in the middle of the room. If this is not the case the mechatron will be instructed to move towards this area to avoid the obstacle. If the device was already in the middle, the IR sensors are checked again to determine which direction has the most room to manoeuvre. Once the particular situation has been ascertained, the function sets the appropriate angle for the device to turn.

The `Obstacle_Task_Check()` function operates in a similar fashion to the `New_Task_Check()` function discussed earlier. Due to an open environment being used, the `Coridor_Angle` variable discussed earlier becomes obsolete. Now when a right hand turn is implemented, the `Rel_y_Metres` variable increases, (becomes increasingly negative moving to the left and increasingly positive moving to

the right) causing distances travelled in both the relative *x* and *y* direction to be checked for completion.

Upon completion of the turn, the `Obstacle_Task_Plan()` function is entered where a succession of instructions will be formulated for MARVIN to both avoid the object and get back on course. The first distance instruction sent to the control system instructs the mechatron to move a distance of 1 m. As MARVIN moves this distance, the side IR proximity sensors check to see if the object is still present. Once it has been passed, the device moves an extra 0.3 m for clearance purposes. The basic course plotted for the device to avoid an object can be seen below in Figure 4-19.

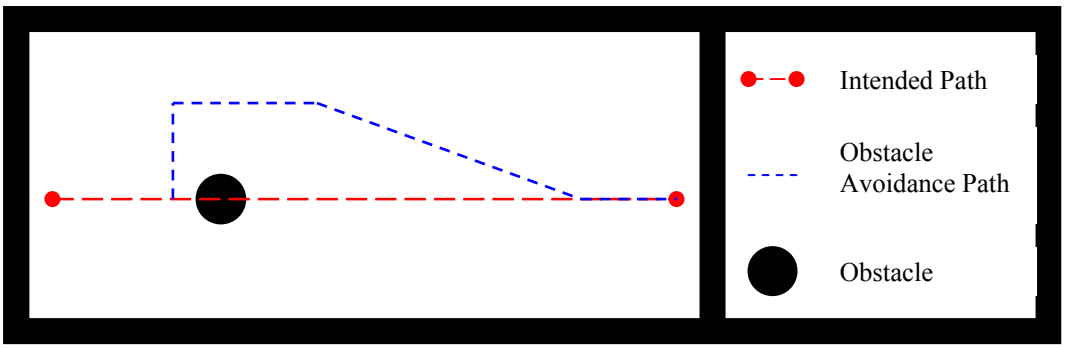



***Figure 4-19: Obstacle avoidance path***

Once the obstacle has been avoided, the device has to be put back on the correct path as shown by the diagonal line in Figure 4-19. The code to achieve this is the same in section 4.3.4. Once the mechatron is instructed to get back on course, any other obstacle detected will be avoided in the same manner.

# 5. RESULTS

## 5.1. Odometer Calibration

During this testing phase it was crucial that MARVIN not be put at risk. If the motor drivers blew up, or MARVIN was damaged in a collision, two master's projects other than this would be compromised. Hence it was decided to limit the PWM signal sent from the mircocontroller to half duty cycle. By limiting the PWM signal to half duty cycle, the device now has a top speed slightly greater than 0.6 metres per second.

The conversion factor initially used to convert the odometry counts to a distance value in metres was calculated by the designer of MARVIN, Daniel Loughnane [Loughnane, 2001]. This original factor (24867 pulses per metre) was calculated by multiplying the drive system reduction ratio with the number of slots on the odometer code wheel and dividing this result by the circumference of the wheels, giving the number of pulses per metre (section 3.2.3.2). This value proved to be inaccurate and needed to be recalculated which could have been due to changes in the pressures of the wheels, altering their circumference. A tape measure was fixed to the floor and the mechatron was instructed to travel a distance of four metres over various velocities. The distance the device thought it had travelled was then divided by the measured distance, giving a distance ratio that was plotted against the velocity of the mechatron. This is illustrated in Figure 5-1.

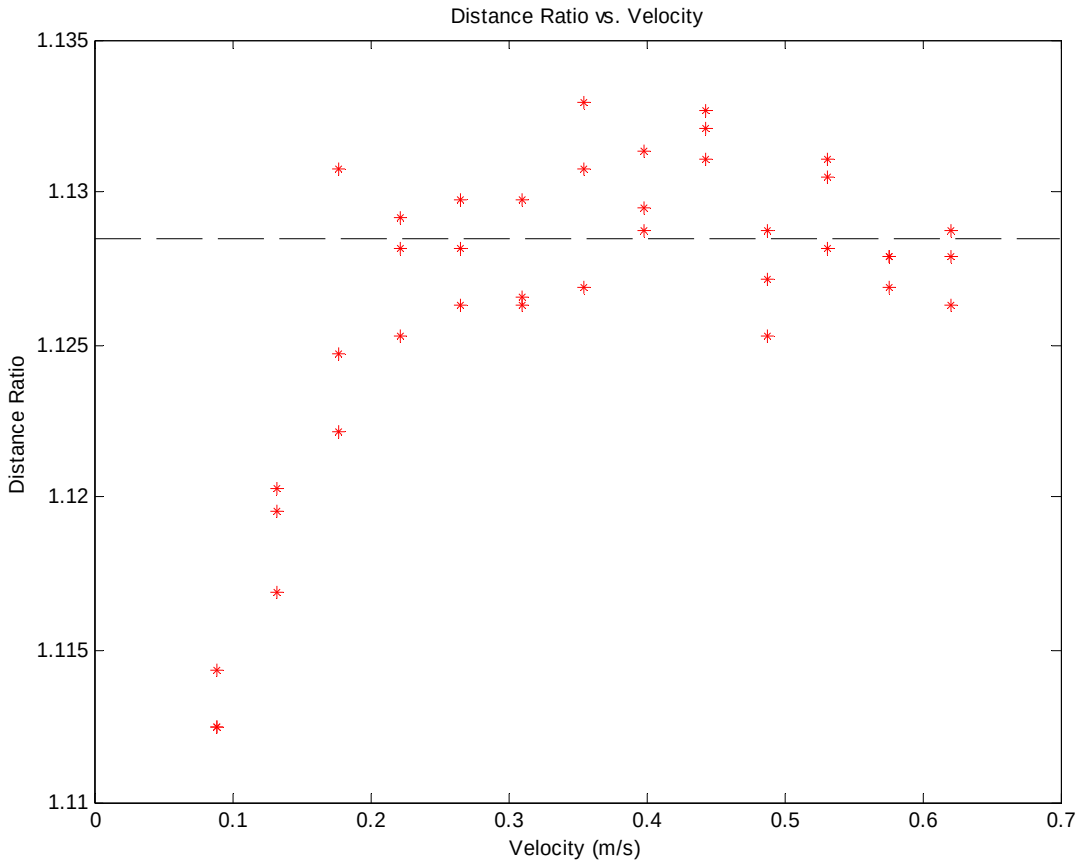



***Figure 5-1: Distance ratio vs velocity graph***

The dotted black line shows the average distance ratio (1.1285) of MARVIN for velocity values greater than 0.15 metres per second. As the average distance ratio was greater than one, the mechatron thought it was travelling further than it actually was. The low distance ratio values encountered at velocities less than 0.15 metres per second indicate the device was travelling closer to the intended set distance. A possible explanation for this could be that as the velocity of MARVIN increased, the mechatron experienced a higher degree of wheel slippage until it levelled out to a constant value, at speeds greater than 0.2 metres per second. As the device will be operated at velocities between 0.2 and 0.6 metres per second, this average distance ratio is used as a multiplier to calculate a new more accurate conversion factor of 28062 pulses per metre.

## 5.2. Corridor Navigation

Journeys through the corridors of C Block were conducted to test the effectiveness of the navigation algorithm. The first test conducted was to navigate MARVIN from the graduate students' room, down the corridor, performing a left hand turn and stopping outside the Electronics Laboratory. A sample journey of each of the three velocities used (0.2, 0.4 and 0.6 metres respectively) during the test can be seen in the following three figures, along with the generated velocity profile. In each case MARVIN was instructed to travel 12.6 metres in the *x* direction from a starting point of (5.94, 1.50) metres with respect to the origin of the map (note that due to the nature of MATLAB arrays which the map is based on, the origin is located in the top left hand corner intead of the conventional bottom left), then to move 4.2 metres in the *y* direction.

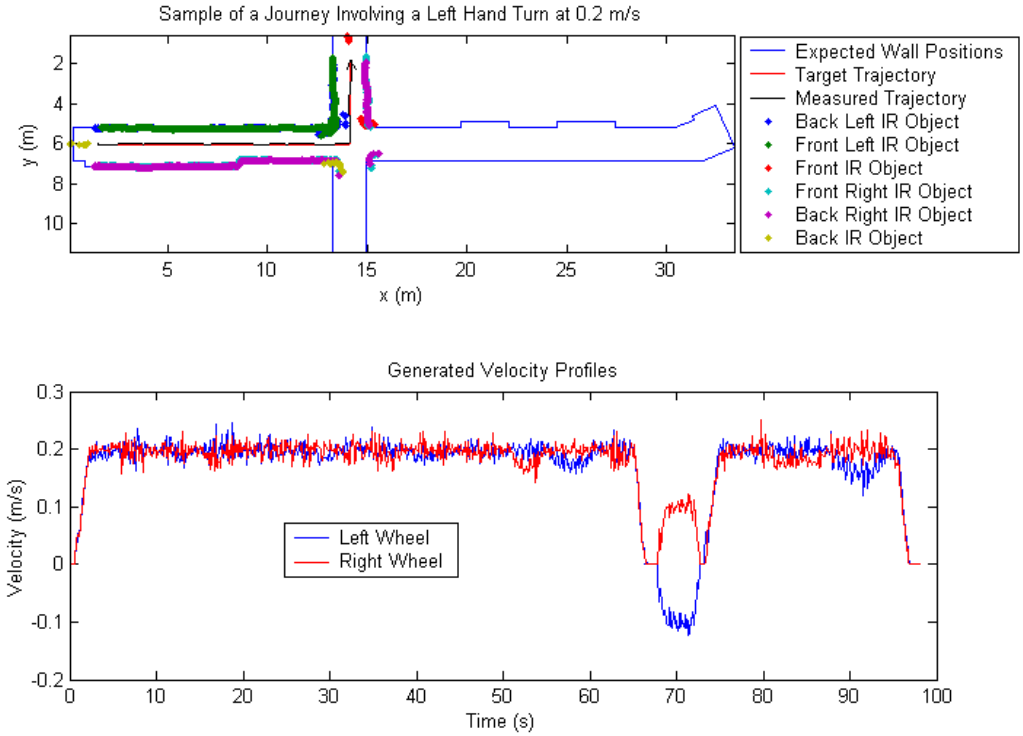



***Figure 5-2: Sample journey to the Electronics Laboratory at 0.2 m/s***

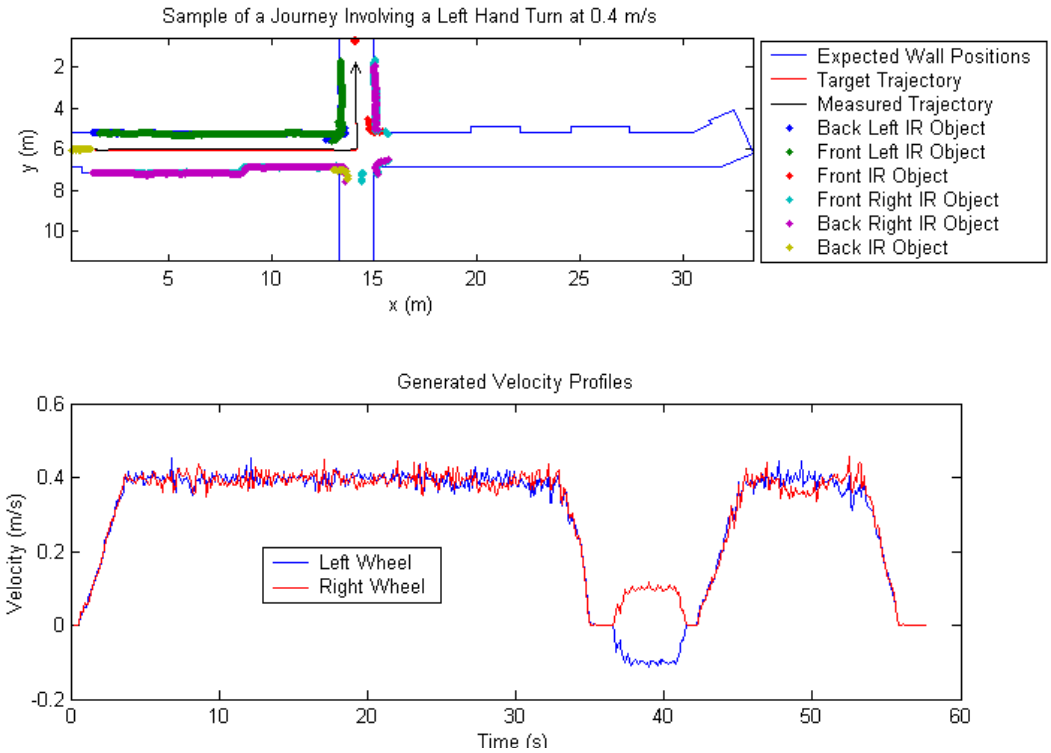



***Figure 5-3: Sample journey to the Electronics Laboratory at 0.4 m/s***

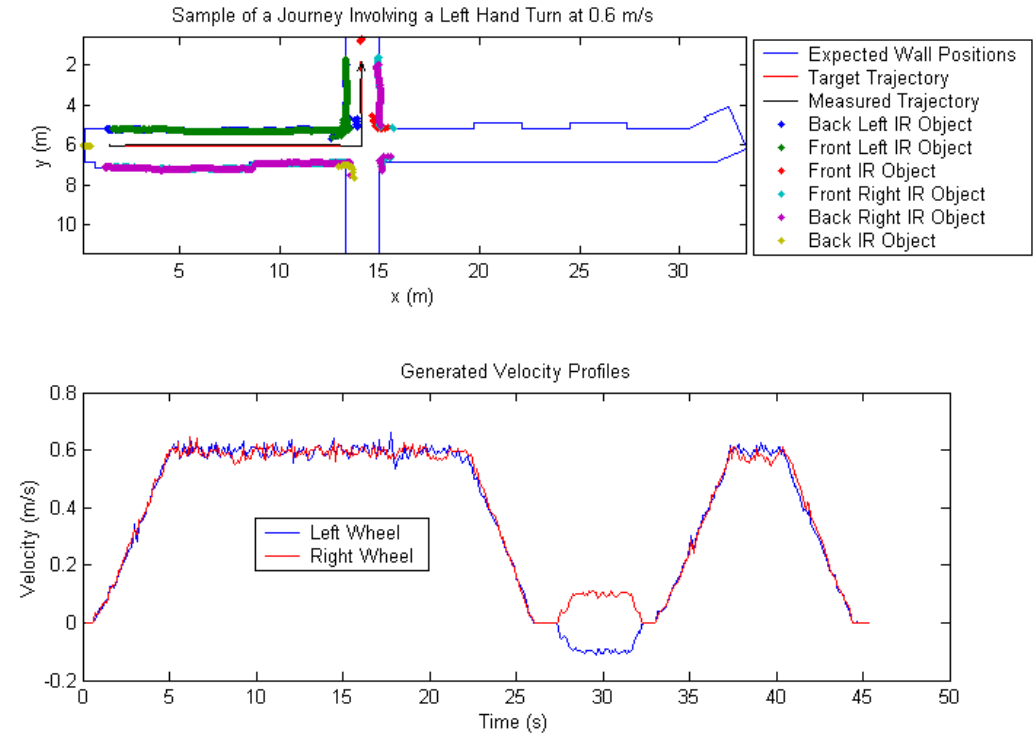



***Figure 5-4: Sample of a journey to the Electronics Laboratory at 0.6 m/s***

In each of the three sample journeys shown, the plotted measured trajectory shows MARVIN kept to the middle of the corridor, successfully completing the designated instructions, stopping outside the Electronics Laboratory as intended. The generated velocity profile of each sample journey shows that the intended velocity the mechatron was travelling at and also the three stages of the journey which were a straight distance of 12.6 metres, a standing turn and another straight distance of 4.2 metres.

After completion of these tests, measurements were taken from the intended destination of MARVIN, to where the device actually ended up at the completion of the journey. These results are illustrated in Figure 5-5. Note the figure is a zoomed-in version of the main environment, with the wall positions indicated on the figure to compare with the sample journeys shown in the three previous figures.

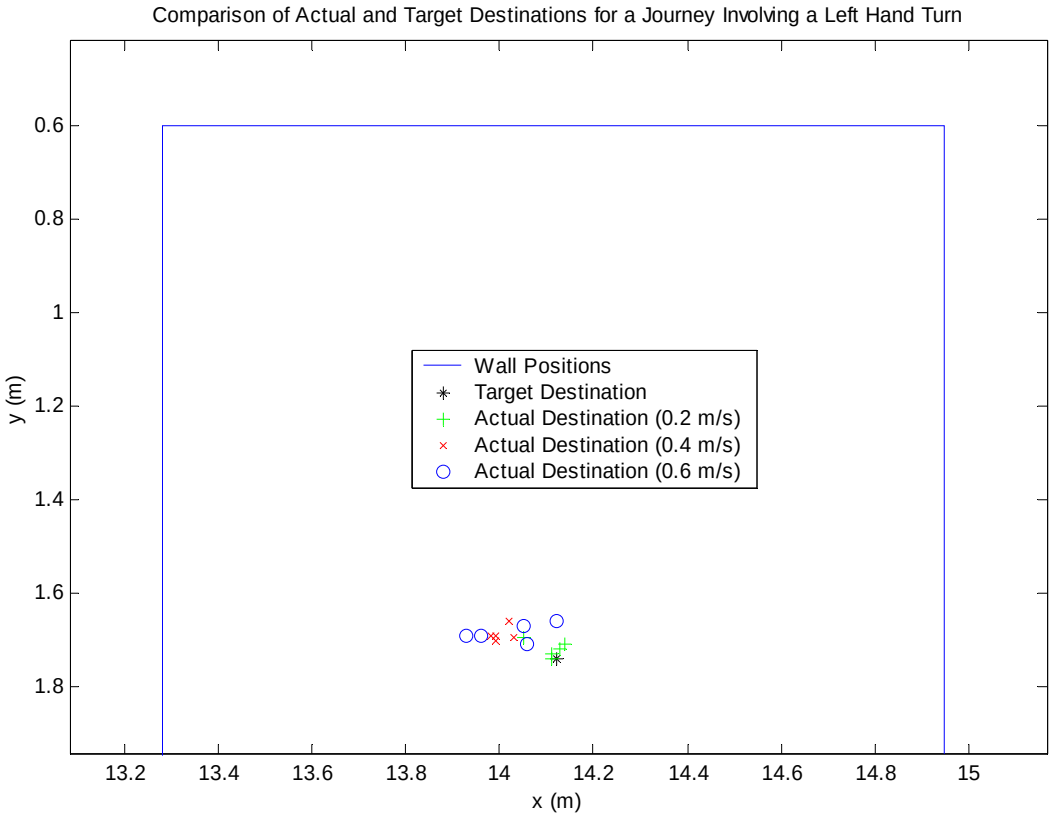



***Figure 5-5: Comparison of actual and target destinations resulting from a journey to the Electronics Laboratory.***

Figure 5-5 shows that MARVIN stops just past the intended point with a tendency to be slightly to the left at velocities greater than 0.2 metres per second. The overshooting error can be attributed to a slight drift occurring before the left hand standing turn was conducted. The fact that the actual destinations are to the left of the intended point at the higher velocities can be explained due to there being less time for the infrared proximity sensors to achieve readings in order to correct any offset from the middle of the corridor. This can be corrected by travelling further than 4.2 metres down this part of the corridor after the standing turn, allowing the device to gather more infrared readings and reorientate itself closer to the middle of the corridor.

The target destination of MARVIN was at (14.12, 1.74) metres with respect to the origin. Overall, after travelling a combined distance of 16.8 metres and conducting a left hand standing turn, the device got very close to its target destination. The greatest deviation was -0.19 metres off in the *x* direction at 0.6 metres per second, with this position still being within a grid space of the target point.

The second test conducted was to navigate MARVIN down the length of the corridor from the graduate students' room, to stop near the door to the office of Dr Dale Carnegie. Sample movements of the mechatron for each of the three velocities (0.2, 0.4 and 0.6 metres respectively) can again be seen in the following three figures with the generated velocity profile. In each sample case the device was instructed to travel a straight distance of 29.1 metres in the *x* direction from a fixed starting point of (5.94, 1.50) metres with respect to the origin of the map.

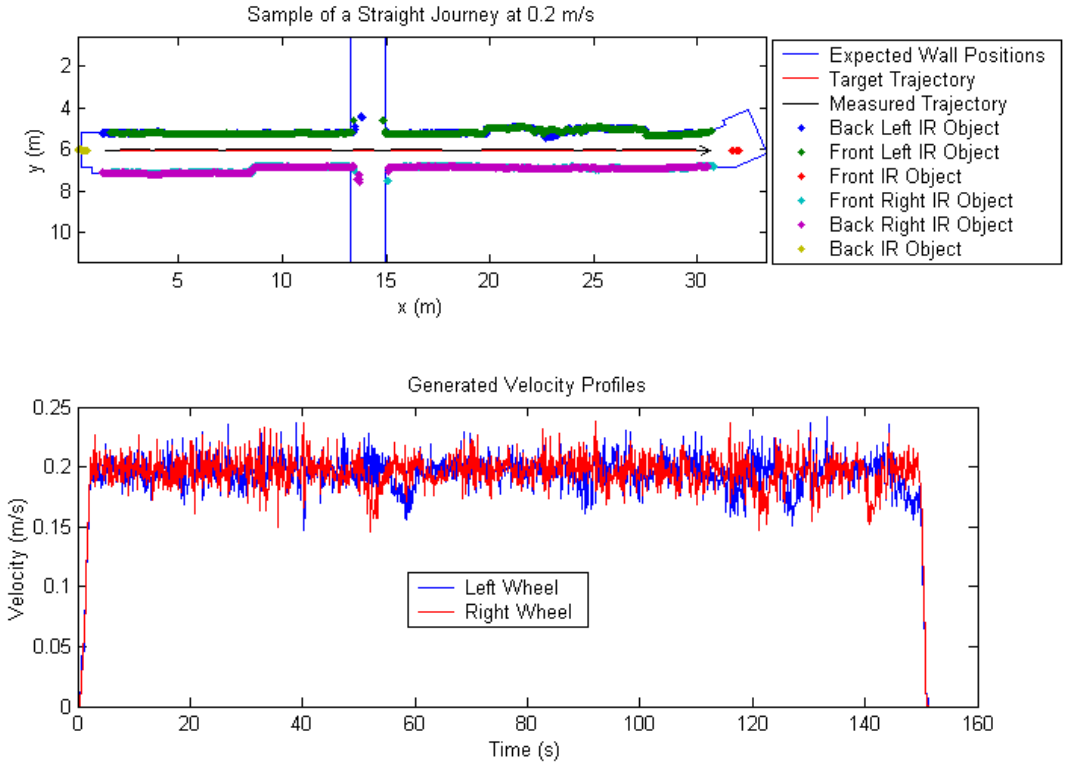



***Figure 5-6: Sample of a journey to Dr Dale Carnegie's office at 0.2 m/s***

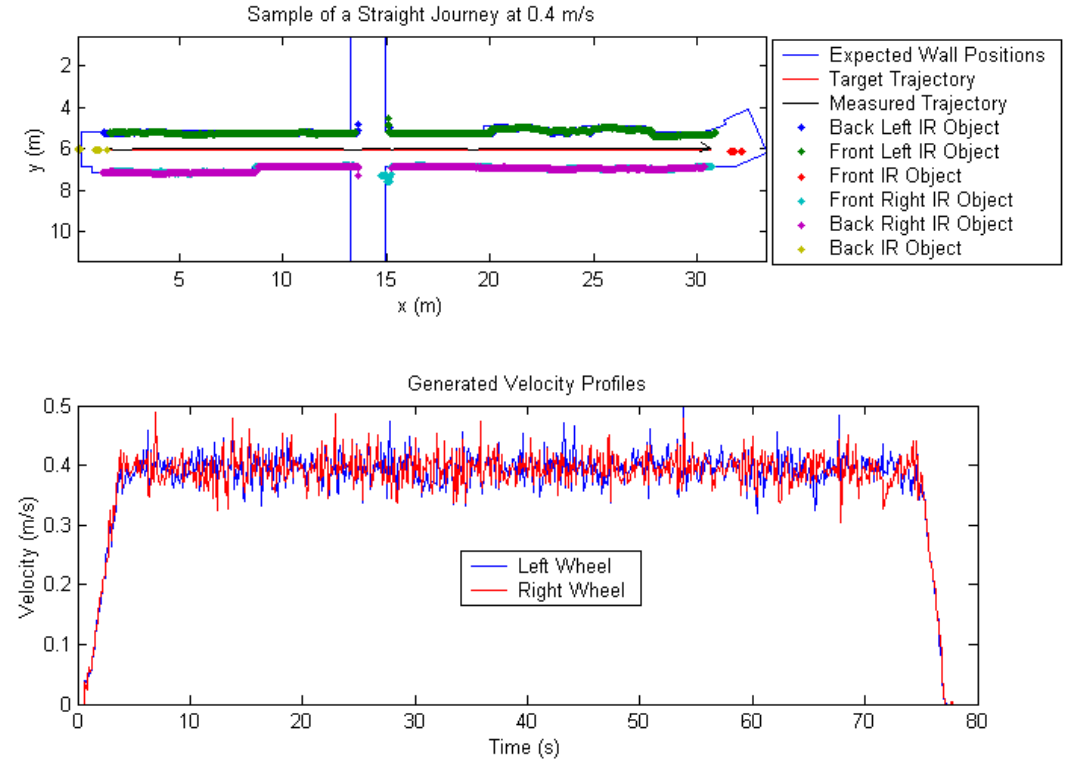



***Figure 5-7: Sample of a journey to Dr Dale Carnegie's office at 0.4 m/s***

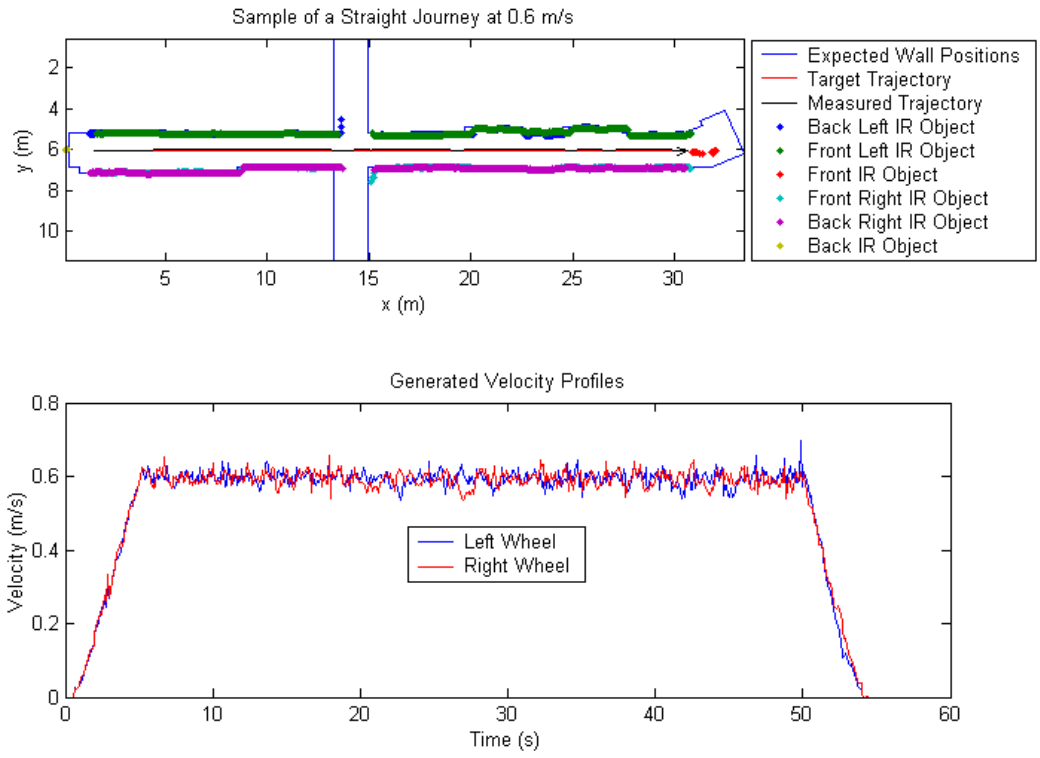



***Figure 5-8: Sample of a journey to Dr Dale Carnegie's office at 0.6 m/s***

From the three sample journeys shown, the measured trajectory shows that MARVIN again kept to the middle of the corridor even considering the changes in corridor width as illustrated by the sample journeys. The infra sensors accurately show the wall positions for the entire journey in each sample with MARVIN successfully stopping near Dr Dale Carnegie's office as intended. The velocity profiles of each sample journey again show that the intended velocity set for each case was adhered to.

Figure 5-9 shows the intended destination of MARVIN and where the device finished after the completion of the designated task. Again the figure is a zoomed-in version of the main environment, showing the wall positions with respect to the intended and target destinations.

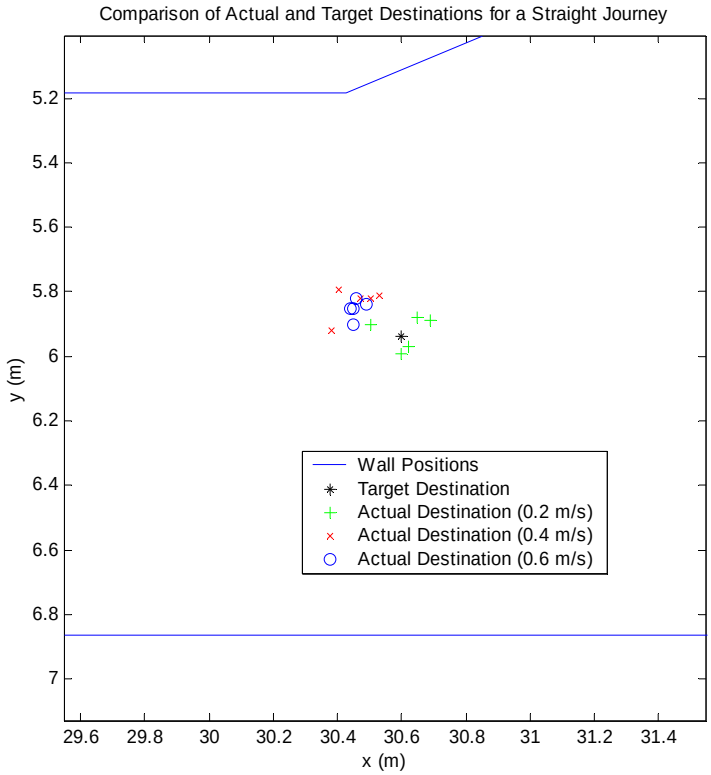



***Figure 5-9: Comparison of actual and target destinations resulting from a journey to Dr Dale Carnegie's office***

From Figure 5-9 it can be seen MARVIN again preformed better at a velocity of 0.2 metres per second with results from the other two velocities falling slightly short of the intended destination. This could be due to changes in the width of the corridor which were taken into account by the software, but MARVIN still drifted slightly after passing these points. With the target destination being at the point (30.60, 5.94) the furthest point was -0.220 metres away at 0.4 metres per second in the *x* direction, which is again very close and within a grid space of the target point.

Overall the results from these two journeys show MARVIN can reasonably accurately navigate from a known start point, to an intended destination at a range of velocities.

## 5.3. Beacon Integration

With the correction factor calculated in section 5.1 applied to the generated odometry counts, tests were conducted using the beacons to ascertain their accuracy and

reliability. A beacon was placed in a fixed position, 8 metres from MARVIN. The mechatron was instructed to travel past the beacon from a fixed position each time, with the generated beacon correction factor (discussed in section 4.3.5) being plotted against the velocity of the mechatron as seen in Figure 5-10.

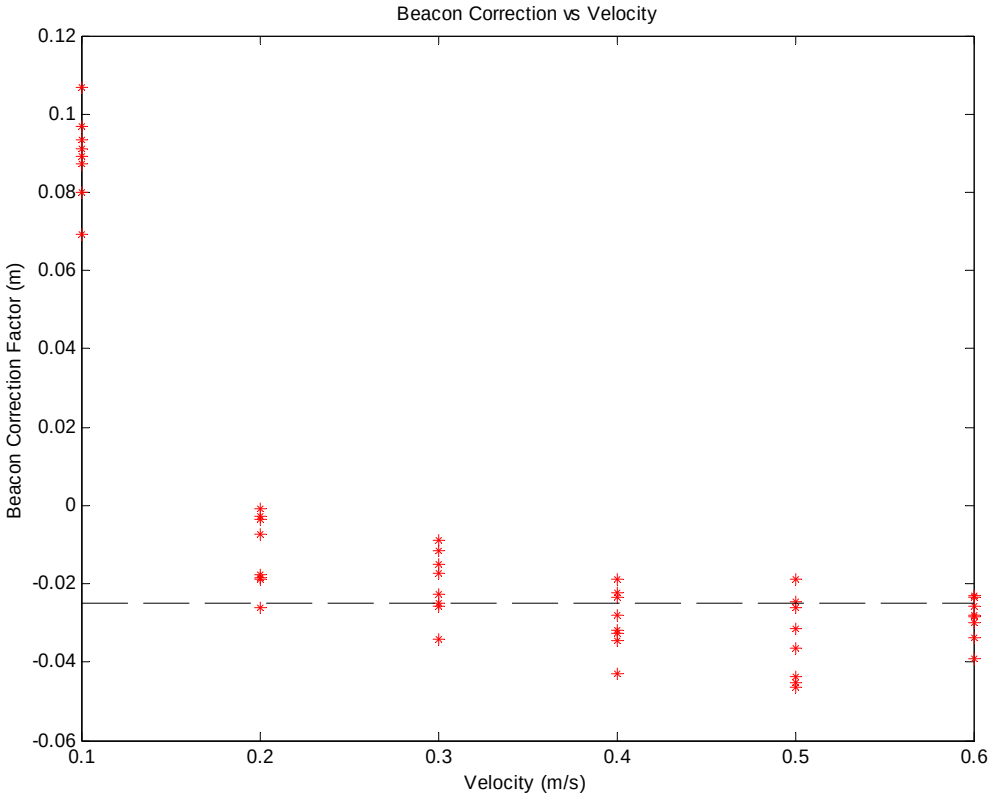



***Figure 5-10: Beacon correction factor vs maximum velocity graph***

As the odometry conversion factor calculated in section 5.1 relates to velocities greater than 0.15 metres per second, the values obtained at 0.1 metres per second are not used. The dotted black line shows the average correction factor (-0.025 m) generated by the beacons for MARVIN's operating velocities.

A follow up to this test was conducted. This involved MARVIN starting in a position 3 metres in front of where the device actually thought it was. This position was at (4.50, 5.94) metres whereas MARVIN thought it was starting at the point (1.50, 5.94) metres. The purpose of this test was to determine if a beacon could correct this initial distance offset. The device was instructed to travel a straight distance of 17.1 metres.

A sample result of this test for the two velocities of 0.4 and 0.6 metres per second, along with the generated velocity profiles can be seen in the following two figures.

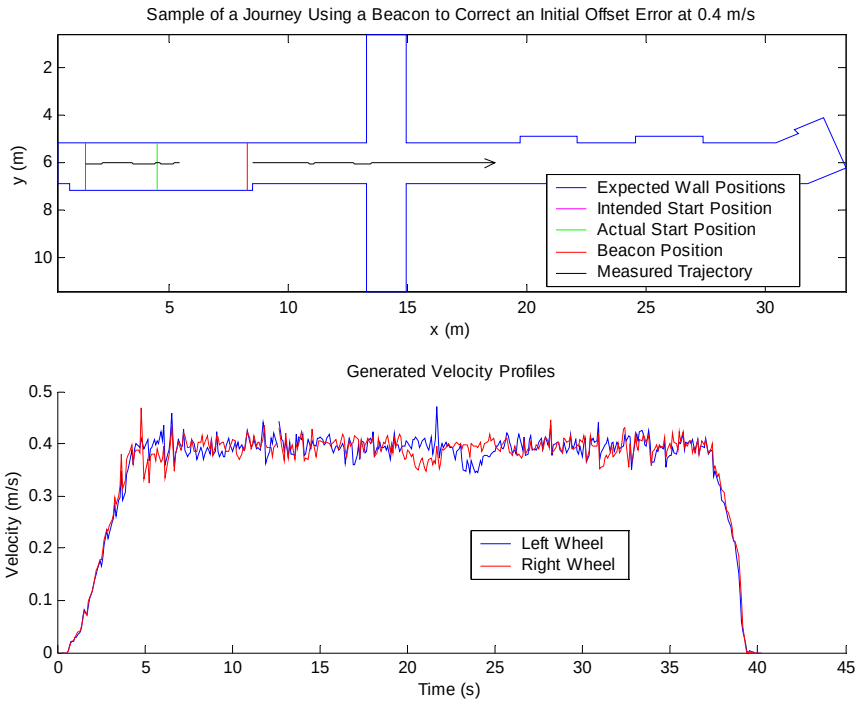



***Figure 5-11: Sample of a journey using a beacon to correct an initial offset error at 0.4 m/s***

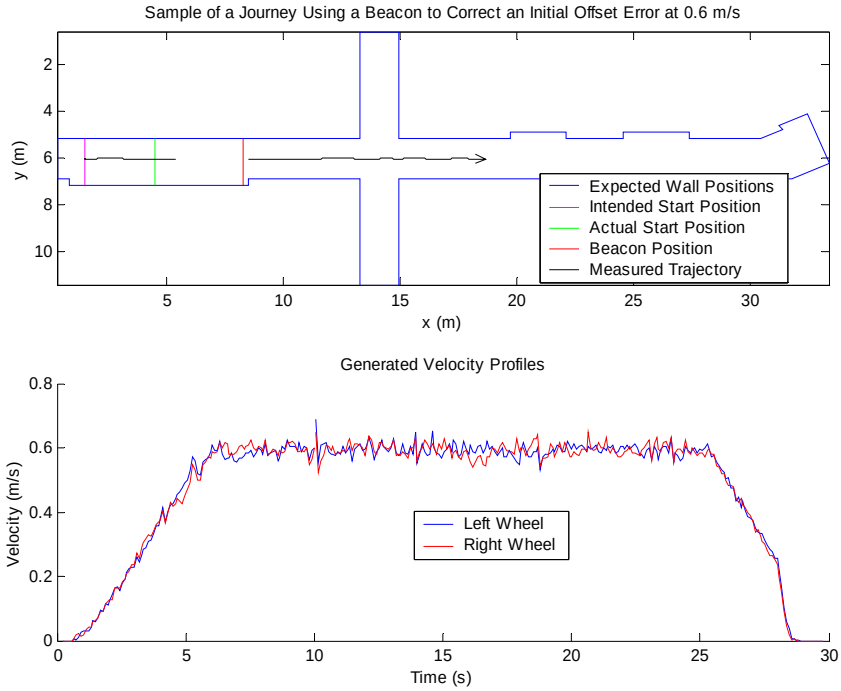



***Figure 5-12: Sample of a journey using a beacon to correct an initial offset error at 0.6 m/s***

A velocity of 0.2 metres per second was not used as an intermittent noise problem while travelling at this velocity occurred towards the end of testing, causing the beacon signal to continuously trigger, generating false results. This is discussed further in section 6.2.3.

As shown by the two sample journeys, after passing the beacon's position, MARVIN's position was updated. A new distance was sent to the control system using the generated correction factor, giving a distance that would enable MARVIN to end up in the position it would have been in if an initial offset wasn't applied to the starting position, as explained earlier in section 4.3.5. This section of code effectively took into account the offset distance and set a new appropriate distance to nullify the initial 3 metre offset.

The results of where MARVIN finished, compared to the intended finishing point are shown in Figure 5-13, again zoomed-in as explained earlier.

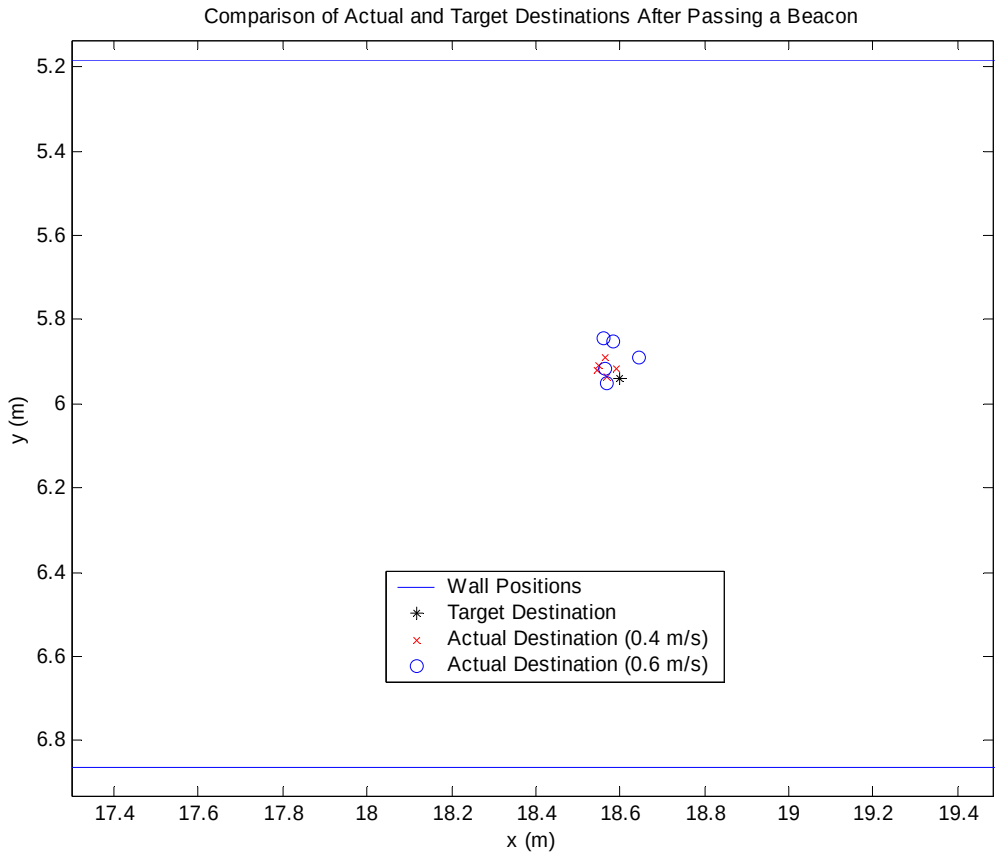



***Figure 5-13: Comparison of actual and target destinations resulting from a journey using a beacon to correct an initial offset error***

As shown by Figure 5-13, the actual destinations were very close to the target destination which was at the point (18.6, 5.94) metres. MARVIN performed slightly better at velocities of 0.4 metres per second with all of the results from this test very close to the target destination. Taking into account the initial starting offset of three metres the results were extremely good, with the worst actual destination being only -0.05 metres away in the *x* direction.

As the intended use of the beacons is to eliminate any accumulative odometry error built up over long journeys through the environment, these result shows how they can be used to accurately update the mechatron's position.

## 5.4. Obstacle Avoidance

To test this algorithm, MARVIN was instructed to travel a fixed straight distance of 4.5 metres. If a dynamic obstacle was detected during this time, instructions were planned out and executed to avoid the obstacle and get the mechatron back on course, stopping in the originally intended position. The dynamic obstacle used in this test was a person standing in front of the mechatron. A sample at the two velocities (0.2 and 0.4 metres per second) used for this test can be seen in the following two figures. Note this test wasn't conducted at the top speed of 0.6 metres per second due to the short distances involved. For each of these samples, MARVIN started from a fixed point of (3.4, 4.2) metres with respect to the origin.

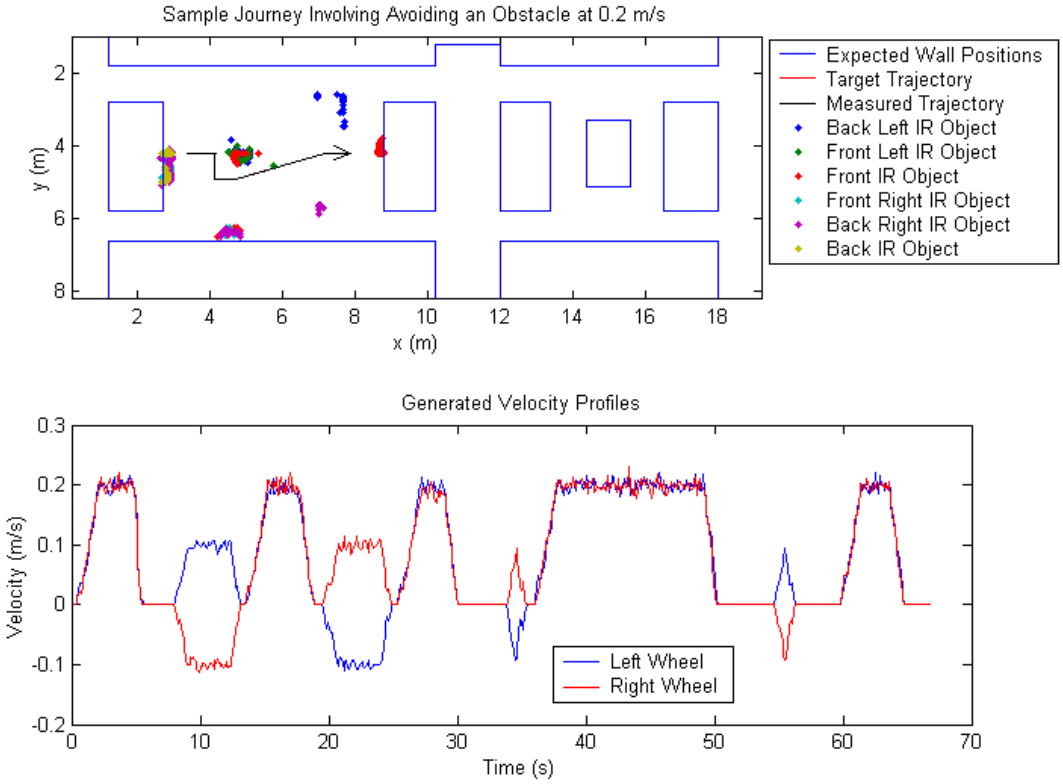



***Figure 5-14: Sample of a journey to avoid an obstacle at 0.2 m/s***

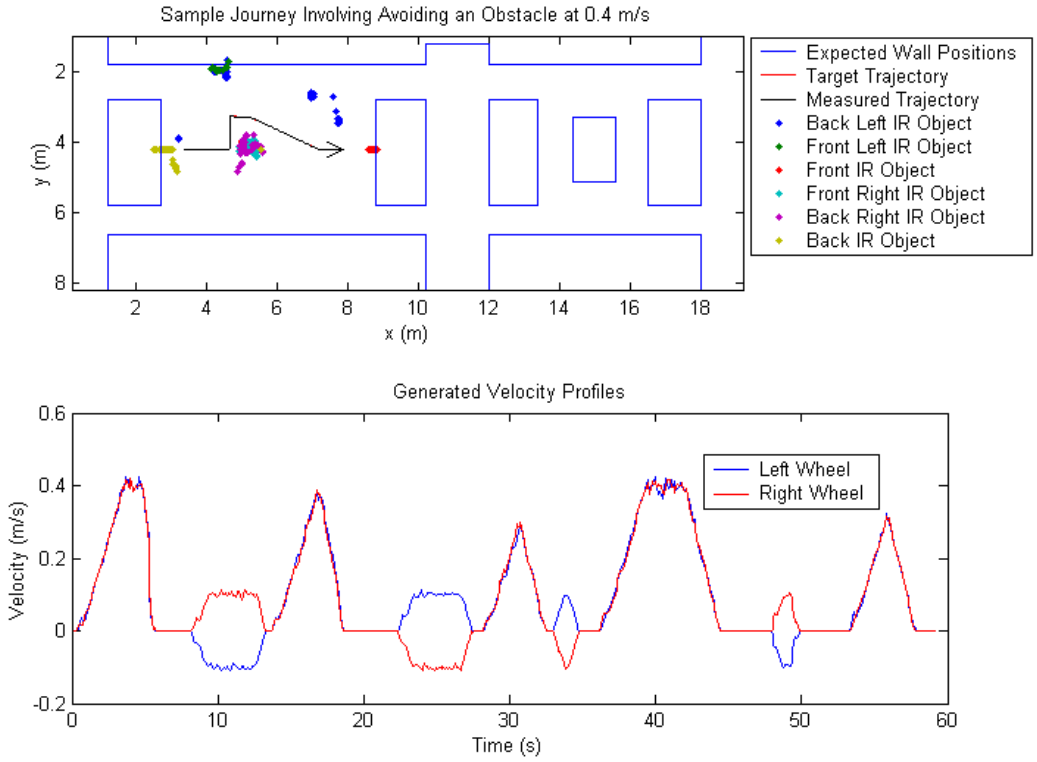



***Figure 5-15: Sample of a journey to avoid an obstacle at 0.4 m/s***

The sample obstacle avoidance journeys show the trajectory of MARVIN as the device avoids the dynamic obstacle. The infrared values shown in front of the mechatron before it turns indicate the position of the human obstacle to avoid. The generated velocity profiles show how a variety of instructions were carried out to avoid the obstacle and get MARVIN back on course.

Figure 5-16 shows how far the device stopped from the intended point after avoiding the human obstacle and getting back on course.

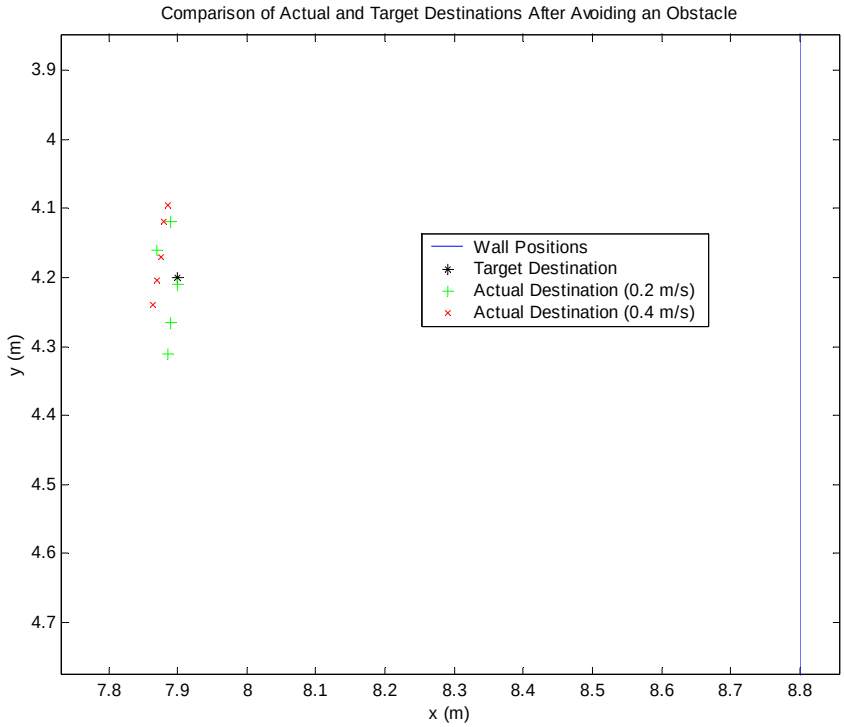



***Figure 5-16: Comparison of actual and target destinations resulting from avoiding an obstacle***

From Figure 5-16 the actual destinations were scattered around the target destination of (7.90, 4.20) metres with neither of the two velocities used being closer than the other overall. The worst result with respect to the *x* direction was -0.035 metres which is very close to the target point, this occurred at a velocity of 0.2 metres per second. In the *y* direction the worst result was 0.110 metres away from the target point, this occurred at a velocity of 0.4 metres per second. Due to the fact that only odometers were used for distance and heading information (the infrared sensors were only used for obstacle detection as this environment was void of walls within range to

obtain heading information) and considering the number of turns involved in avoiding the obstacle to get back on course, these results are very good.

After effectively avoiding one person, a second test was performed to see how the software would cope having to avoid two people. A sample of these results at the two velocities of 0.2 and 0.4 metres per second can be seen in the next two figures.

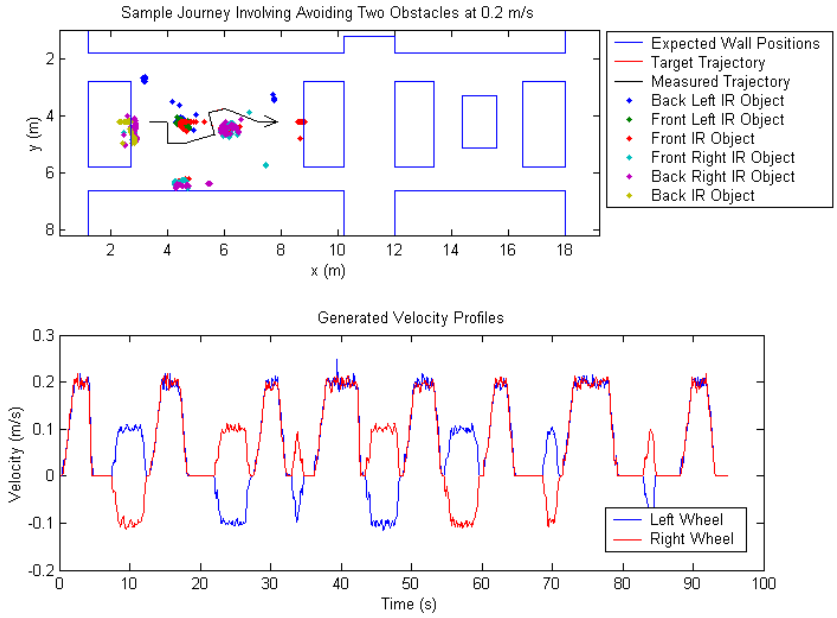



***Figure 5-17: Sample of a journey to avoid two obstacles at 0.2 m/s***

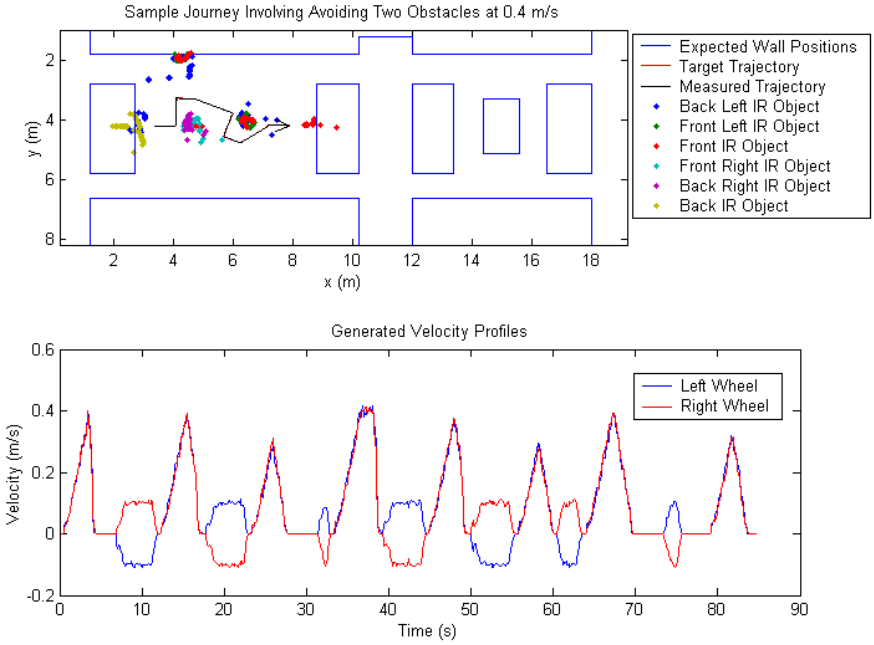



***Figure 5-18: Sample of a journey to avoid two obstacles at 0.4 m/s***

The results of these two sample journeys involving avoiding two dynamic obstacles are similar to Figure 5-14 and Figure 5-15 when one obstacle was avoided. The measured trajectory again shows how MARVIN avoided the obstacles placed in front of the device, successfully manoeuvring around them. The infrared sensors clearly show the position of the two obstacles from the mechatron's perspective. From the generated velocity profiles it can be seen that several different instructions were required to get the device to navigate around the obstacles and back on course.

Again measurements were taken to see how close the device was to its target destination after avoiding the obstacles as shown in Figure 5-19.

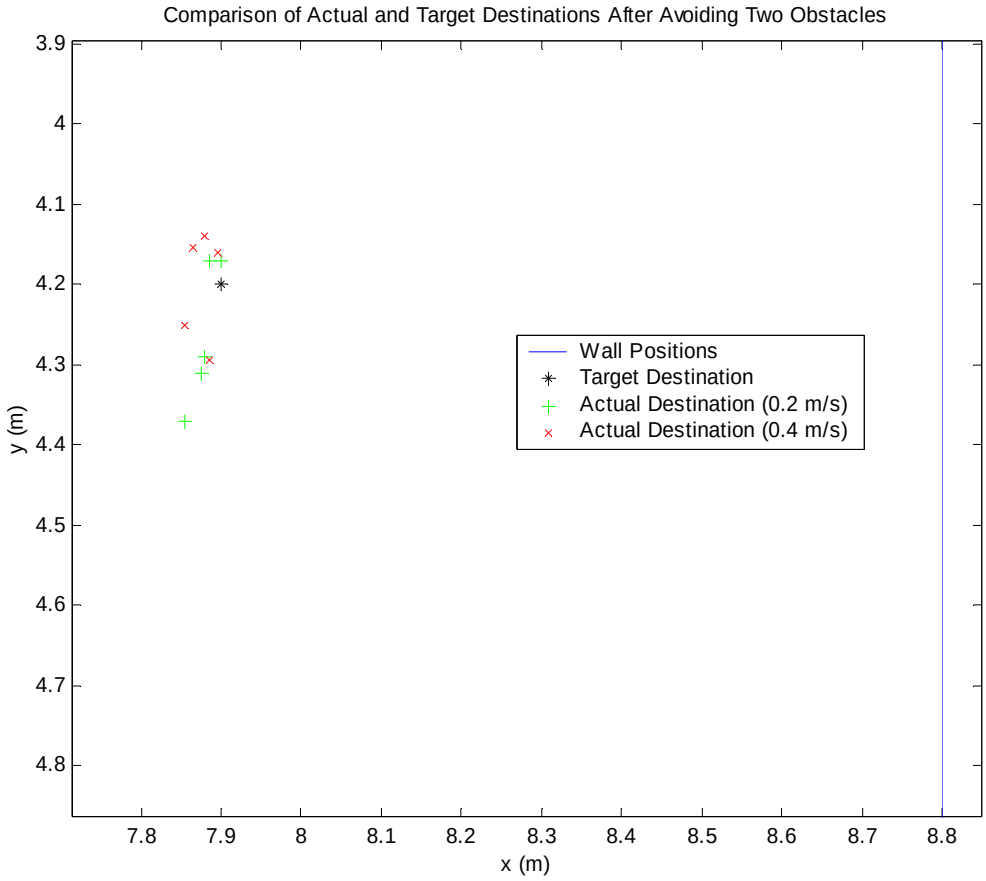



***Figure 5-19: Comparison of actual and target destinations resulting from avoiding two obstacles***

As can be seen in Figure 5-19, MARVIN got closer to the target destination of (7.90, 4.20) metres while travelling at 0.4 metres per second. Again the results were very close to the target destination with respect to the *x* direction as the worst result was only -0.045 metres away. In the y direction the results were again slightly further

away from the target point with the worst being -0.170 metres away. Overall these results were very good considering twice as many instructions were carried out to avoid the second obstacle as compared with the first and also taking into account only odometers were used again for heading and distance information

After observing these results, tests were conducted to see how far the actual destination would be from the target point if no obstacle was avoided at all. MARVIN was instructed to travel the same distance of 4.5 metres without placing a dynamic obstacle in front of the mechatron's path. MARVIN conducted these tests at the same velocities that the obstacles were avoided at, 0.2 and 0.4 metres per second. Sample journeys at these velocities can be seen in the next two figures.

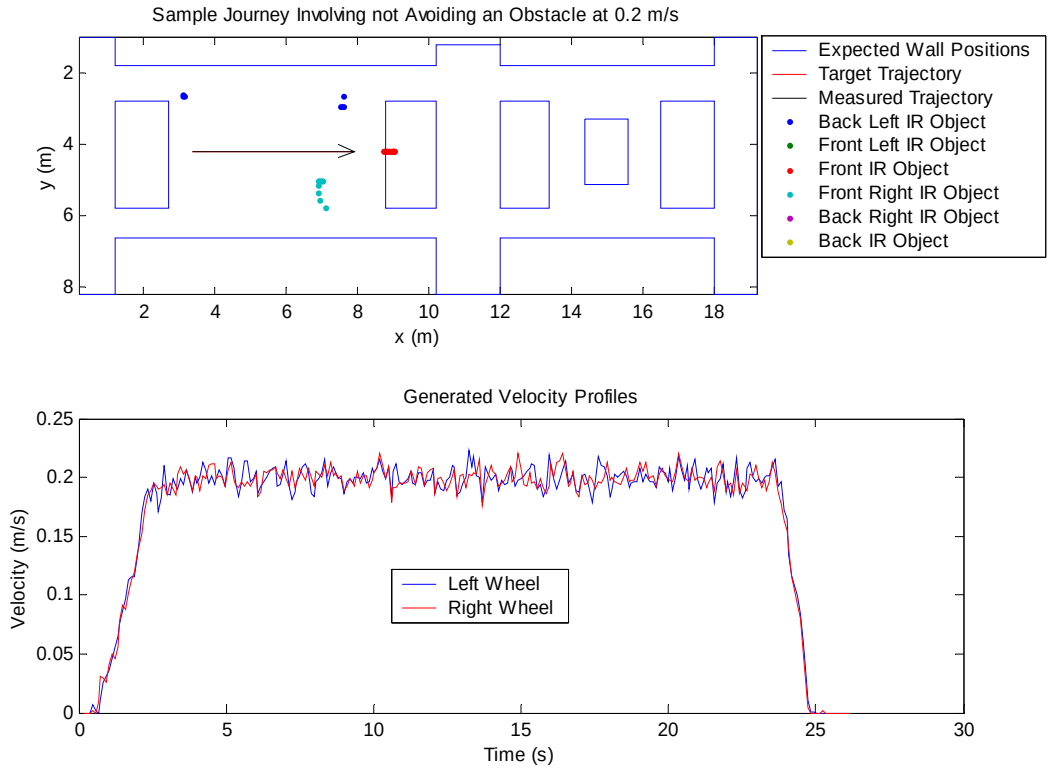



***Figure 5-20: Sample journey involving not avoiding an obstacle at 0.2 m/s***

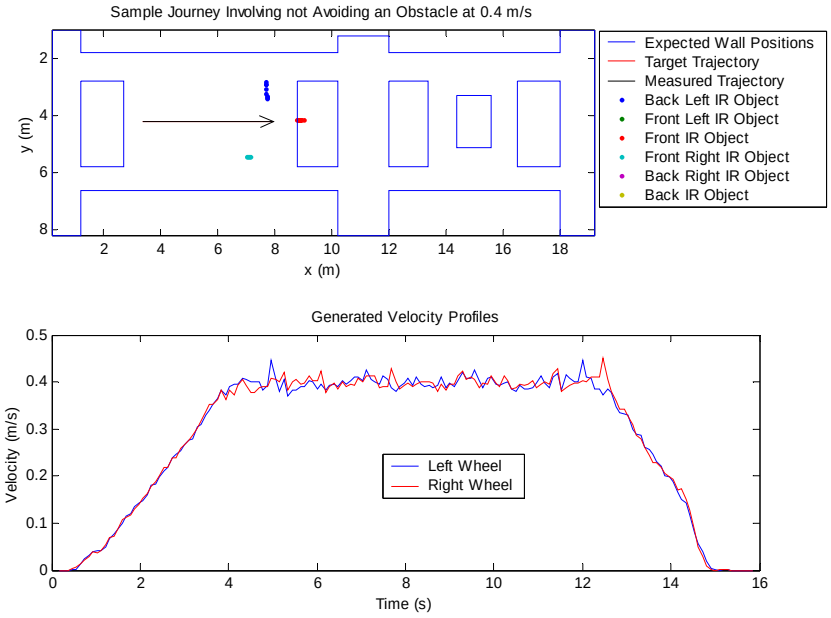



***Figure 5-21: Sample journey involving not avoiding an obstacle at 0.4 m/s***

From the two sample journeys it can be seen MARVIN travelled in a dead straight line coming to a stop after the set distance. The generated velocity profiles show the velocity the instruction was carried out at.

The comparison of the target to actual destinations can be seen in Figure 5-22.

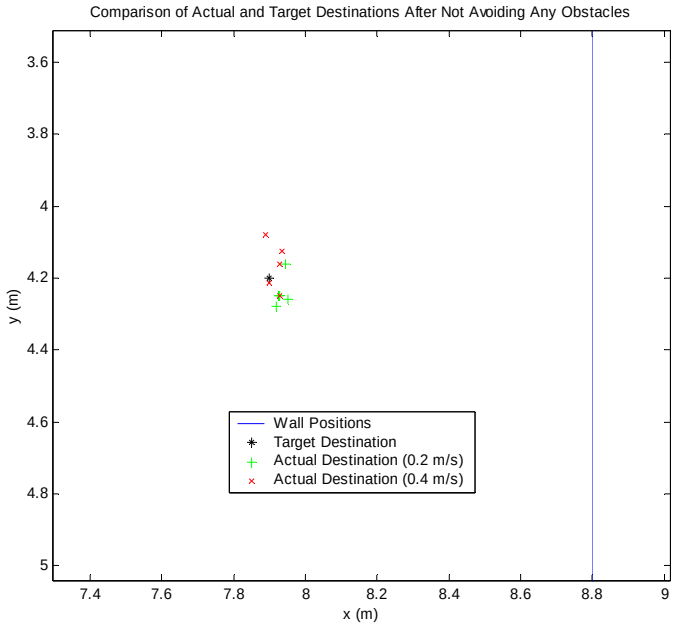



***Figure 5-22: Comparison of actual and target destinations resulting from avoiding no obstacles***

As only odometry encoders were being used for distance and heading information during this test, care had to be taken with aligning MARVIN. If the device was aligned slightly off, this heading error would be amplified as the mechatron moved the set distance of 4.5 metres resulting in an increased offset in the *y* direction.

From Figure 5-22 it can be seen the actual destinations were very close to the target destination, with most of the actual destinations slightly over shooting the target point. The worst result in the *x* direction was 0.05 metres with the worst in the *y* direction being -0.12 metres.

The results achieved from this test are similar to those shown by Figure 5-16 and Figure 5-19 where actual obstacles were avoided. It can be therefore be concluded that the obstacle avoidance algorithm is not introducing a significant additional error in MARVIN's positioning.

Overall these results show the obstacle avoidance algorithm can effectively navigate MARVIN around not just one, but two dynamic obstacles at different velocities. Figure 5-22 reinforces the effectiveness of the obstacle avoidance algorithm and its ability to get MARVIN back on course stopping very close to where the device was intended to if an obstacle hadn't been present. There is no reason why this algorithm wouldn't be able to cope in a situation that involved avoiding multiple obstacles.

# 6. CONCLUSIONS

## 6.1. Summary of Results

The odometers were recalibrated after initial tests found noticeable discrepancies between the measured distance MARVIN had travelled and the distance the device itself thought it had travelled. The difference in these initial tests could have been due to the tyres of the mechatron being re-inflated which would have altered the circumference of the wheels.

After taking experimental results of how far the device thought it had travelled and dividing this by the actual distance travelled by MARVIN, a distance ratio was obtained. This was plotted against velocity as shown in Figure 5-1. Using values from this graph in the velocity range between 0.15 metres per second and 0.6 metres per second, an average distance ratio of 1.1285 was determined. This value was then used as a multiplier to alter the original odometer counts per metre into a more accurate version. By using this multiplier the conversion factor changed from 24867 pulses per metre to a value of 28062 pulses per metre.

Two tests were conducted using the corridor navigation algorithm to test its effectiveness. These tests involved navigating MARVIN from a known start point to two different target destinations. The first test required MARVIN to travel a straight line distance of 12.6 metres in the *x* direction, conduct a standing left turn, and then move a further 4.2 metres in the *y* direction, stopping outside the target destination of the Electronics Laboratory, a total distance of 16.8 metres. This was conducted at velocities of 0.2, 0.4 and 0.6 metres per second. During these journeys the mechatron followed the intended trajectories, conducting the movements in a stable fashion as illustrated by Figure 5-2, Figure 5-3 and lastly Figure 5-4. The resulting actual destinations proved to be reasonably close to the target destination as shown in Figure

5-5, with worst result in the *x* direction 1.1% (-0.19 m) of the total distance travelled and the worst result in the y direction 0.5% (-0.08 m) of the total distance travelled.

The second test involved MARVIN travelling a straight distance of 29.1 metres from a fixed point down the corridor, stopping near the office of Dr Dale Carnegie. This was conducted at the same velocities as the previous test, and again resulted in a stable movement as the mechatron traversed down the corridor as shown in Figure 5-6, Figure 5-7 and lastly Figure 5-8. The resulting actual destinations were very close to that of the target point. The worst result in the *x* direction was 0.76% (-0.22 m) of the total distance travelled and was 0.50% (-0.145 m) in the *y* direction. The results from these two tests demonstrate the accuracy of the navigation algorithm and its ability to navigate from a known start position to a required destination point.

The average beacon correction factor was found after conducting a test using a beacon positioned at a fixed point of 8 metres from MARVIN's starting point. The resulting beacon correction factor was plotted against the mechatron's velocity and showed the beacons updated MARVIN's position to within an accuracy of -0.025 metres.

A further test was conducted to integrate the beacons with the navigation system to try and eliminate an initial offset error of three metres. Figure 5-13 illustrates how close MARVIN finished to the target destination point with the worst result in the *x* direction being 0.35% (-0.05 m) of the total distance travelled and 0.67% (-0.095 m) in the *y* direction of the total distance travelled (note the total distance travelled for this test was 14.1 metres after factoring in the 3 metre starting offset). This was an extremely good result and proved the value of the beacons as a sensor to update MARVIN's position, eliminating any accumulative error.

Tests involving the obstacle avoidance algorithm were conducted in the Electronics Laboratory with a human acting as the dynamic obstacle to be avoided. The tests were conducted at velocities of 0.2 and 0.4 metres per second with the results of avoiding one obstacle shown in Figure 5-14 and Figure 5-15, and the results of avoiding two obstacles shown in Figure 5-17 and Figure 5-18. The shape of the obstacle can be seen from the mechatron's perspective in these figures from the infrared sensor values. Also illustrated is the trajectory used by MARVIN to

successfully avoid the obstacles and get back on the required course, with the velocity profiles showing the array of instructions carried out during this manoeuvring. Figure 5-16 and Figure 5-19 show the actual destinations of MARVIN after avoiding one and then two obstacles respectively in comparison with the target destination.

In the case of avoiding one obstacle, the worst result in the *x* direction was -0.035 metres and 0.110 metres in the *y* direction. When two obstacles were avoided -0.045 metres was the worst case in the *x* direction with 0.017 metres the worst in the *y* direction. In both of these cases this was a very pleasing result considering only odometers were used for distance and heading information, generating these minimal errors. A further test was conducted to see if the resulting actual destinations from avoiding obstacles were comparable with travelling the intended distance without encountering an obstacle.

The results from this test can be seen in Figure 5-22 with the worst result in the *x* direction being 0.05 metres and the worst in the *y* direction being -0.12 metres. These results are similar to those achieved when avoiding an obstacle which indicates the obstacle avoidance algorithm does not significantly add to positioning error. In retrospect due to the odometers being the only sensor used for heading and distance measurements, more care should have been taken in aligning the device before conducting the tests. This would have helped eliminate any initial heading error that may lead to smaller offsets in the *y* direction. Overall these results prove MARVIN can effectively avoid two obstacles and get back on course. As more than one obstacle can be avoided there is no reason why this algorithm couldn't avoid multiple obstacles.

## 6.2. Future Work

### 6.2.1. Dynamic Mapping

MARVIN currently makes use of a built-in map of the initial environment for navigation and localisation tasks, this could be further modified or replaced to

incorporate dynamic mapping. The advantage of this technique is that it will enable the device to keep track of changes in the environment, altering the map as the mechatron moves. Another future extension of this could be to remove the built-in map altogether, enabling MARVIN to create its own map, constructed through movements in the environment, based on the received sensor data. However, this is an extremely difficult task. The odometers in conjunction with the IR proximity sensors already fitted to the mechatron can be used to help achieve dynamic mapping. An example of how they can be used is illustrated below in Figure 6-1.

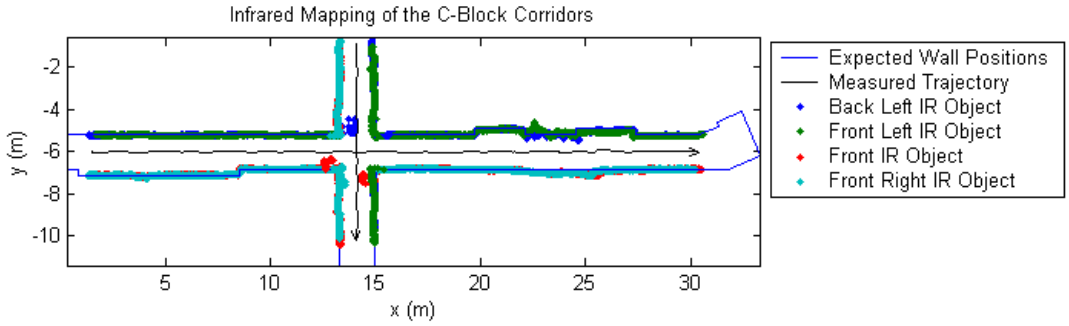



***Figure 6-1: IR mapping of the C Block corridors***

Figure 6-1 shows two straight line journeys of MARVIN superimposed upon each other. The light blue and red dots show the values of the two right side IR proximity sensors, with the green and dark blue dots representing the two left hand side IR sensors. The values give a relatively accurate representation of MARVIN's environment. The future installation of the laser range finder, discussed earlier in this thesis, will contribute further to this task being achieved.

### 6.2.2. Program Merging

There are two programs that need to be merged with the navigation system. These are the obstacle avoidance system and the voice recognition system. Due to the dimensions of the corridor environment, MARVIN cannot make use of the obstacle avoidance software. Once more sections of the science block are made accessible to the mechatron which aren't as constrained, these two programs can be combined. A check to see if there is room to avoid an unexpected obstacle can be performed. As

the structure of both sets of code is very similar, most of the appropriate functions can be merged together. Due to testing and debugging of the obstacle avoidance algorithm having only recently been completed, time did not permit the merging to be attempted.

The voice recognition system was still under development at the time of writing of this thesis. An interface to merge these two programs has been written but hasn't been tested as discussed briefly in section 4.3.1. Due to the Humanisation program (which includes the voice recognition) wanting to be able to possibly detect unauthorised personnel at all times, with the navigation system wanting to receive sensor data at all times as well, these two programs will need to be multi threaded.

### 6.2.3. Other Improvements

As MARVIN presently makes use of standing turns, these can be modified to encompass integrated turns. By sending a non zero distance and angle to the control system, an integrated turn will be preformed as discussed in section 4.3.4. The sending of this instruction will have to occur earlier than a standing turn to enable the device enough room to conduct the turn. The position of the device relative to the centre of the corridor will also have to be checked, ensuring it is not too close to the wall before a turn is implemented.

Several further hardware installations still need to occur on MARVIN. A compass needs to be located and installed on the mechatron to give a more accurate heading. The laser range finder also needs to be installed and integrated with the device. With these two additional sensors installed, more information can be gathered about MARVIN's position, aiding the navigation and control systems.

The last install involves the generic motor drivers currently underdevelopment by Craig Jenson. If they prove more reliable and efficient than the current drivers being used, they can be installed to replace Andrew Payne's design, as initially intended.

A modification to the beacon emitter could also be explored. This could possibly involve using a convex lens instead of the shielding cone already in place to see which method produces the best focussing results. This has not been explored as the shield cone focussing method worked more than adequately. A diagram showing the two methods can be seen below in Figure 6-2.

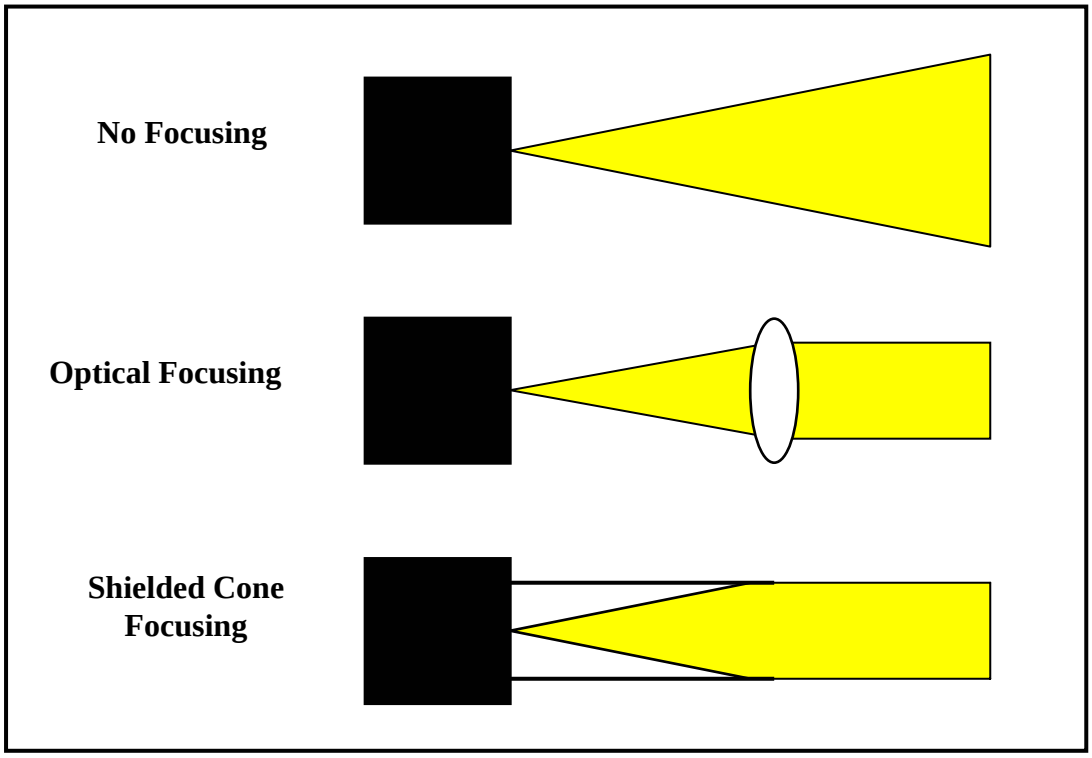



***Figure 6-2: Light focussing methods***

An additional improvement to the beacon system can be to focus on the intermittent noise problems that caused false triggers to occur at low speeds. Due to this problem occurring recently, a thorough investigation was not undertaken into possible reasons behind it.

## 6.3. Evaluation and Conclusions

MARVIN has been completely overhauled since the initiation of this project. Along with the upgrading and fitting of the DAQ card, PC and new odometry encoders, the device has had infrared proximity sensors and beacon receivers mounted and

installed. With the help of Andrew Payne, a microcontroller and two motor drivers were successfully constructed and integrated with the device, enabling the mechatron to be fully operational. Once this had occurred, the objectives set out for this project could be accomplished.

An interface between the voice recognition and navigation system has been written with limited testing due to Ashil Prakash still working on his humanisation project in Wellington with Robotechnology [www.robotech.co.nz]. When these programs are retested, they should be fully compatible after some minor alterations.

The `Task_Plan()` function has successfully broken journeys into small manageable sections that MARVIN can carry out, with the `Control_Feedback()` function using the built-in map to indicate to the control system if the wall distances have changed, setting the infrared weightings to an appropriate value. These two functions together help MARVIN traverse through the intended environment following a relatively stable trajectory, stopping within a maximum error of 1.13% of the total distance travelled from the target destination

A Beacon system has been implemented. The system uses a modified infrared light barrier kit from Jaycar Electronics, comprised of a separate infrared emitter and receiver. This system has been successfully integrated with the navigation algorithm, increasing its performance, enabling any accumulative error to be reset with device then stopping within a maximum error of 0.67% of the total distance travelled, from the target destination.

The obstacle avoidance algorithm that was tested in the Electronics Laboratory, effectively avoids any obstacle that appears in front of the device. After the avoidance stage has been completed, the algorithm is then able to get MARVIN back on its intended course. This has been successfully achieved at different velocities and for more than one obstacle.

Overall the navigation algorithm has met all its initial objectives, efficiently navigating MARVIN around its initial environment and being able to avoid dynamic

obstacles. This project forms a solid base and provides the first step towards the future goal of full autonomy for MARVIN.

# APPENDIX A - DEFINITION OF TERMS

## A.1 Gold Codes

Some pairs of maximum sequences (m-sequences) that have the same degree can be used to generate Gold codes by linerarly combining two m-sequences with different offset in Galois Field (refer to Appendix A.2). Though not all pairs of m-sequences can be used to generate Gold codes but those that can are called preferred pairs.

Gold codes have three-valued autocorrelation and cross-correlation function with values:

$$\{-1, -t(m), t(m)-2\}$$

Where

$$t(m) = \begin{cases} 2^{\frac{(m+1)}{2}} + 1 & \textbf{for odd m} \\ 2^{\frac{(m+2)}{2}} + 1 & \textbf{for even m} \end{cases}$$

Using two preferred m-sequence generators of degree r, with a fixed non-zero seed in the first generator, $2^r$ Gold codes are obtained by changing the seed of the second generator from 0 to $2^r - 1$. [http://www-mobile.ecs.soton.ac.uk/bjc97r/pnseq/node3.html]

# A.2 Galois Field

**Arithmetic Operations in a Power-of-Two Galois Field**

Most modern 2D matrix symbologies use Reed-Solomon encoding in a power-of-two Galois field (also known as an "extension field") to provide both error detection and correction. The closed field arithmetic needed both to encode and decode the Reed-Solomon blocks, described in the specifications as "bit-wise modulo 2 and word-wise modulo P", is quite different from everyday arithmetic, and even from the straight modulo arithmetic used in prime Galois fields. This can make it difficult to replicate the encoding examples found in those specifications.

This paper attempts to teach the appropriate operations by building upon C-language encoding functions published in several AIM standards (e.g., "ISS - Aztec Code"). For illustration we will take the widely-used case of GF(256), the Galois field for 28, with a "prime polynomial" of x8 + x5 + x3 + x2 + 1 whose equivalent value P is binary 100101101 or decimal 301. [Several possible prime polynominals can create a GF(256); 301 is the value selected for use with all GF(256) symbologies except QR Code, which uses 285.]

The nature of closed-field arithmetic is that all operations between values in the field produce another value in the field. GF(256) for example contains all the integers between 0 (zero) and 255. The "bitwise modulo 2" aspect of this arithmetic is best seen in the Addition and Subtraction operations, which BOTH are a bit-wise exclusive-or of the two numbers. For example:

given: A = 14110 = 100011012
and: B = 4310 = 001010112
then: A + B = A - B = 101001102 ( = 16610, not the everyday result!).

Again, note that both the sum and difference between two numbers in GF(2N) arithmetic is the exclusive-or of their corresponding bits, expressed in C by the caret, namely "A ^ B". Thus they are realized in the following two functions:

```
int Sum (int A, int B) { return (A ^ B); }
int Difference (int A, int B) { return (A ^ B); }
```

[That the Sum and Difference are identical functions has the odd but true corollary that in such a Galois field each number is its own negative! That is, A equals minus A... which of course works out because then the sum of "A" and "-A" is the exclusive-or of the same two values which equals zero, as it must!]

Multiplication and Division operations within GF(2N) are performed using Log and Antilog tables that are generated by the following C routine:

```
#define GF 256 // define the Size & Prime Polynomial of this Galois
field
#define PP 301
int Log[GF], ALog[GF]; // establish global Log and Antilog arrays

// fill the Log[] and ALog[] arrays with appropriate integer values
void FillLogArrays (void) {
      int i;
      Log[0] = 1-GF; ALog[0] = 1;
      for (i=1; i<GF; i++) {
            ALog[i] = ALog[i-1] * 2;
             if (ALog[i] >= GF) Alog[i] ^= PP;
             Log[ALog[i]] = i;
       }
}
```

Then the Product of two values is the antilog of the mod (GF-1) sum of their logs, namely:

```
int Product (int A, int B) {
      if ((A == 0) || (B == 0)) return (0);
       else return (ALog[(Log[A] + Log[B]) % (GF-1)]);
}
```

...and the Quotient of two values is the antilog of the mod (GF-1) difference between their logs:

```
int Quotient (int A, int B) { // namely A divided by B
      if (B == 0) return (1-GF); // signifying an error!
      else if (A == 0) return (0);
      else return (ALog[(Log[A] - Log[B] + (GF-1)) % (GF-1)]);
}
```

These log and antilog operations closely resemble everyday arithmetic, but with modulo operations to keep all values between 0 and GF-1, and the results are distinctly different! For example, the product of 14 and 33 is not 206, the result from everyday modulo 256 arithmetic, but instead:

given: Log[14] = 54

and: Log[33] = 219

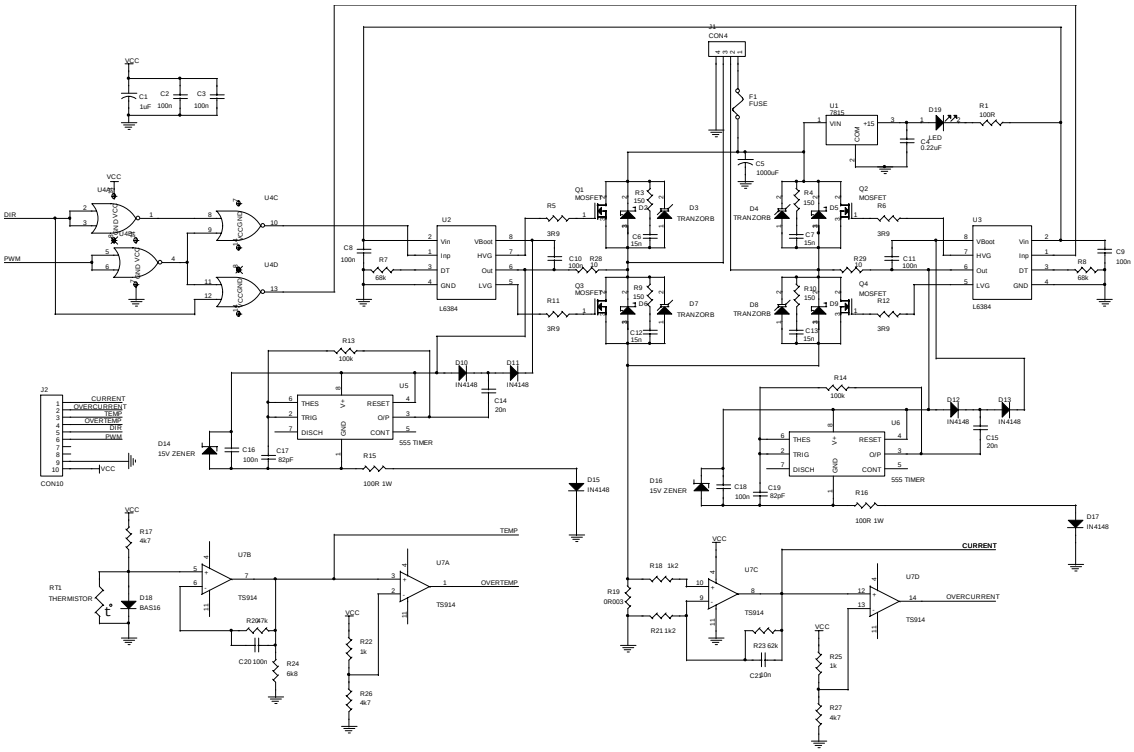
then: (54 + 219) mod 255 = 273 mod 255 = 18

thus: ALog[18] = 227 (again, not the expected product!)

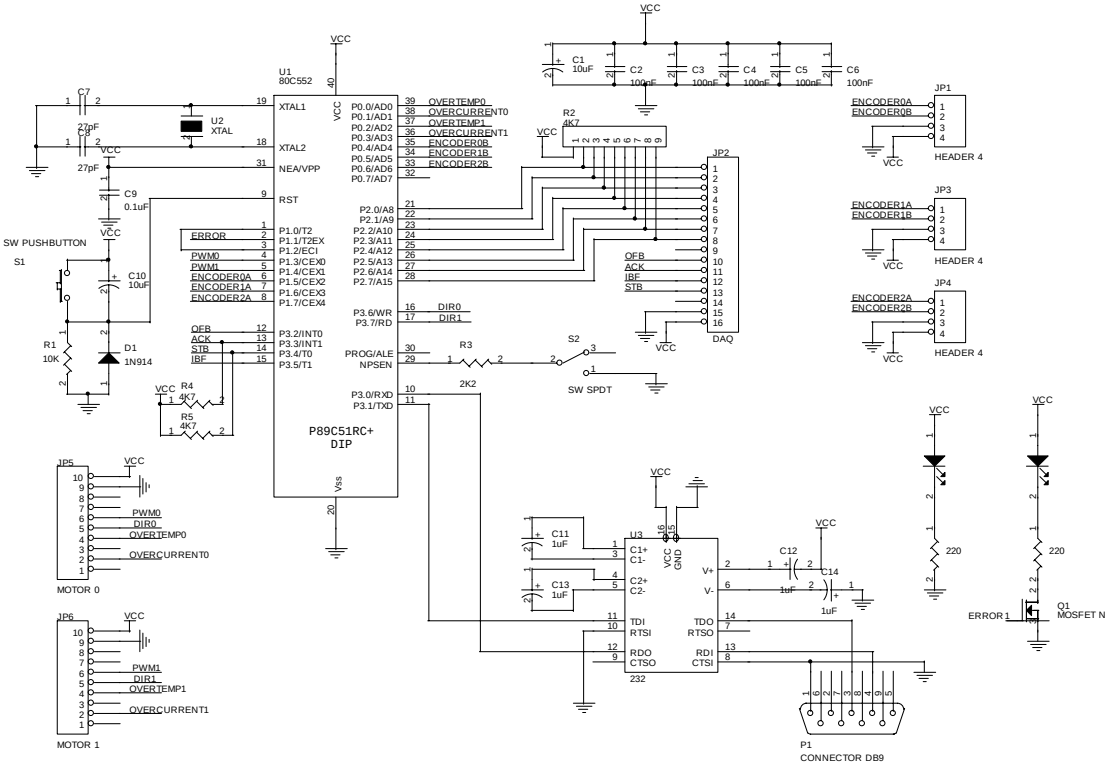
Employing these five routines - Sum(A,B), Difference(A,B), FillArrays(), Product(A,B), and Quotient(A,B) - will allow the encoding examples in the matrix symbology specifications to be confirmed. [http://www.aimglobal.org/aimstore/Galois%20Math.pdf]

# APPENDIX B - SCHEMATICS

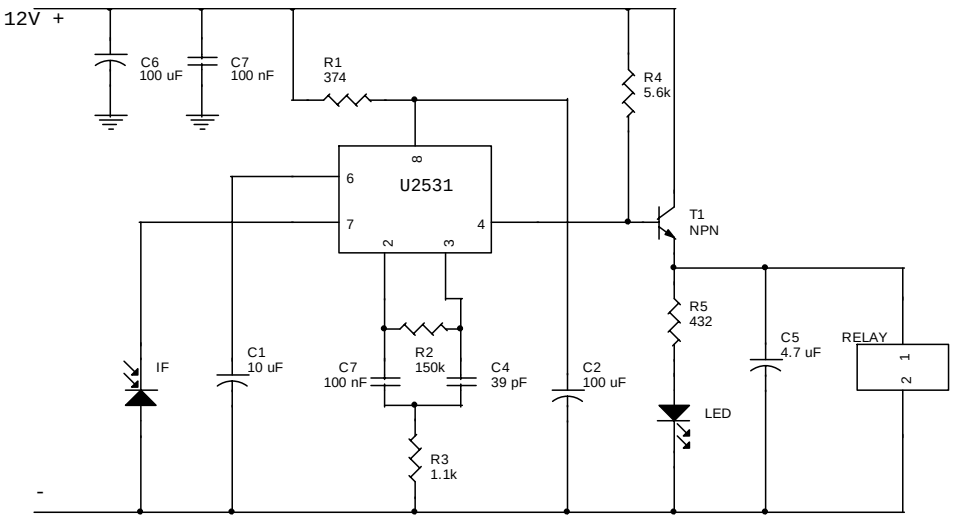
## B.1 Motor Driver



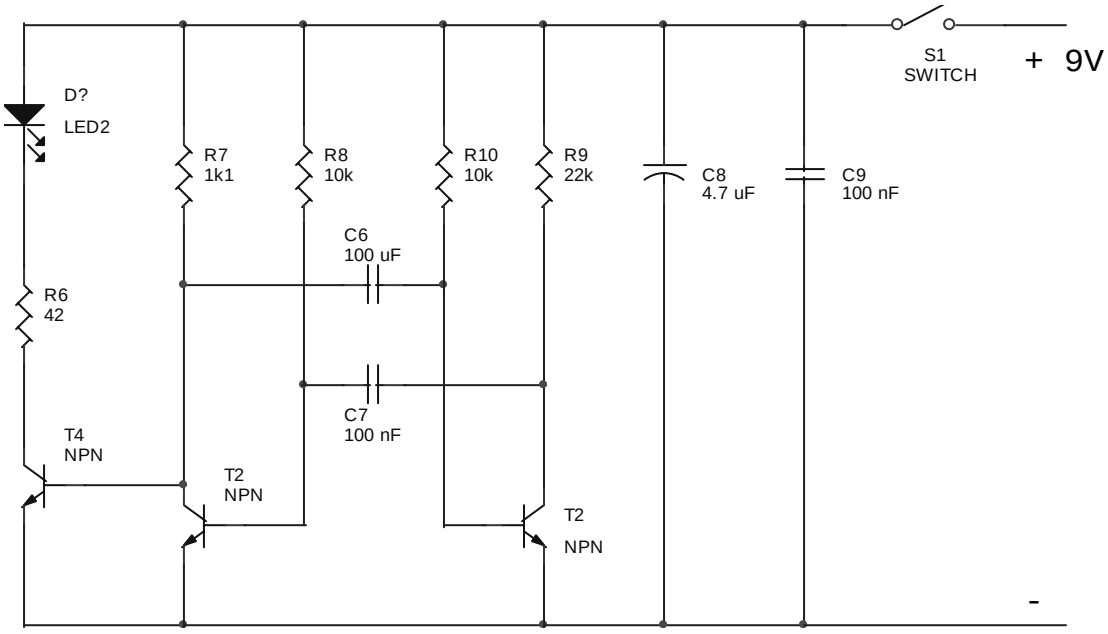
# B.2 MicroController




# B.3 Beacon Receiver



# B.4 Modified Beacon Emitter



# APPENDIX C - CD CONTENTS

The attached CD contains the following:

- **Software**
  - Navigation Algorithm
  - Obstacle Avoidance Algorithm

- **Datasheets**

- **Video Excerpts of Test Journeys**

- **MARVIN photo gallery**

# BIBLIOGRAPHY

Anton, H., *Calculus with Analytic Geometry (Third Edition)*, Wiley, 1988.

Bandera, A., Urdiales, C., & Sandoval, F., **"Autonomous Global Localisation using Markov Chains and Optimised Sonar Landmarks"**, *Proceedings of 2000 IEEE International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, October, 2000.

Borenstein, J., & Feng, L., "**Measurement and Correction of Systematic Odometry Errors in Mobile Robots**", *IEEE Transactions on Robotics and Automation*, October, 1996.

Cordes, J.C., "**The Creating of an Autonomous Multi-Terrain Mechatron**", MSc Thesis, Department of Physics and Electronic Engineering, University of Waikato, 2002.

Carnegie, D.A., "**Towards a Fleet of Autonomous Mobile Mechatrons**", *Proceedings of the 2nd IFAC Conference on Mechantronic Systems*, California, USA, December, 2002.

Dudek, G., & Jenkin, M., "**Computational Principles of Mobile Robotics**", Cambridge University Press, 2000.

Hurd, S.A., "**Laser Range Finding for an Autonomous Mobile Security Device**", MSc Thesis, Department of Physics and Electronic Engineering, University of Waikato, 2001.

Jenson, C.H., Carnegie, D., & Gaynor, P., "**Universal Battery Powered Pulse Width Modulated H-Bridge Motor Control for Robotic Applications**", *Proceedings of the 10th Electronics New Zealand Conference*, Hamilton, New Zealand, September 2003.

King, J.C., "**The Development of an AUV**", MSc Thesis, Department of Physics and Electronic Engineering, University of Waikato, 2002.

Kortenkamp, D., Bonasso, R.P, & Murphy, R., "**Artificial Intelligence and Mobile Robots**", MIT Press, 1998.

Launay, L., Ohya, A., & Yuta, S., "**A Corridors Lights based Navigation System including Path Definition using a Topologically Corrected Map for Indoor Mobile Robots**", *Proceedings of 2002 IEEE International Conference on Robotics and Automation*, Washington, USA, May, 2002.

Lee, S., Amato, N.M., & Fellers, J., "**Localization based on Visibility Sectors using Range Sensors**", *Proceedings of the 2000 IEEE International Conference on Robotics and Automation,* San Francisco, USA, April, 2000.

Lee-Johnson, C.P., & Carnegie, D., "**The Development of a Control System for an Autonomous Mobile Robot**", *Proceedings of the 10th Electronics New Zealand Conference*, Hamilton, New Zealand, September 2003.

Loughnane, D.J., "**Design and Construction of an Autonomous Mobile Security Device**", MSc Thesis, Department of Physics and Electronic Engineering, University of Waikato, 2001.

Payne, A., & Carnegie, D., "**Design and Construction of a Pair of Tricycle Based Robots to Investigate Cooperative Robotic Interaction**", *Proceedings of the 10th Electronics New Zealand Conference*, Hamilton, New Zealand, September 2003.

Prakash, A., Carnegie, D., & Chitty, C., "**The Humanisation of an Autonomous Mobile Robot**", *Proceedings of the 10th Electronics New Zealand Conference*, Hamilton, New Zealand, September 2003.

Prigge, E., & How, J., "**An Indoor Absolute Positioning System with No Line of Sight Restrictions and Building-Wide Coverage**", *Proceedings of 2000 IEEE*

*International Conference on Robotics and Automation*, San Francisco, USA, April, 2000.

Thompson, S., Matsi, T., & Zelinsky, A., "**Localisation using Automatically Selected Landmarks from Panoramic Images**", *Proceedings of Australian Conference on Robotics and Automation,* Melbourne, Australia, 2000.

Sikking, L.J., & Carnegie, D., "**The Development of an Indoor Navigation Algorithm for an Autonomous Mobile Robot**", *Proceedings of the 10th Electronics New Zealand Conference*, Hamilton, New Zealand, September 2003.

Suksakulchai, S., Thongchai, S., Wilkes, D.M., & Kawamura K, "**Mobile Robot Localization using an Electronic Compass for Corridor Environment**", *IEEE Transactions on Systems, Man, and Cybernetics* Nashville, Tennessee, USA, October, 2000.

Victorino, A.C., Rives, P., & Borrelly, J.-J., "**Localization and Map Building Using a Sensor-Based Control Strategy**", *Proceedings of 2000 IEEE International Conference on Intelligent Robots and Systems*, Takamatsu, Japan, October, 2000.

Yamauchi, B., & Beer, R., "**Spatial Learning for Navigation in Dynamic Environments**", *IEEE Transactions on Systems, Man, and Cybernetics - Part B: Cybernetics*, Special Issue on Learning Autonomous Robots, Vol. 26, No. 3, June 1996.