

Flinders University
School of Computer Science,
Engineering and Mathematics

National Instruments Autonomous Robotics Competition 2016: Programming of the robot using LabVIEW

Student: Celeste Mercado
B Engineering (Robotics)(Honours), M Engineering (Electronics)
Academic Supervisor: Dr. Nasser Asgari

Submitted to the School of Computer Science, Engineering and Mathematics
in the Faculty of Science and Engineering in partial fulfilment
of the requirements of the degree of B Engineering (Robotics)(Honours), M Engineering
(Electronics) at Flinders University - Adelaide Australia
October 17, 2016

Declaration

I certify that this work does not incorporate without acknowledgment any material previously submitted for a degree or diploma in any university; and that to the best of my knowledge and belief it does not contain any material previously published or written by another person except where due reference is made in the text.

Celeste Mercado

Due Date: 17/10/16

Acknowledgements

I would like to thank our supervisor, Dr. Nasser Asgari for supporting and assisting the team throughout the project, and also my team members Joel Kluske and Melissa Drogmuller for their contribution to the completion of the project. Furthermore, I would like to thank our mentor Ben Illman who helped with some design considerations and provided pointers with the programming, and also Xan Smith who helped resolve some LIDAR complications.

Abstract

The National Instruments Autonomous Robotics Competition (NI ARC) is an annual competition which encourages students to become innovative in the field of robotics. Teams across New Zealand and Australia are tasked with designing, creating, and programming an autonomous robot capable of completing specific tasks of the competition within six months.

With a focus in the medical field, the theme for the 2016 NI ARC competition was 'Hospital of the Future'. The course was modelled to represent a hospital and entailed the robot to navigate from a starting position and deliver medicine units to the Storage, The Wards, and the Operations Theatre, whilst avoiding collisions into walls and surrounding static and dynamic obstacles.

This is the fourth year Flinders University has competed in the competition. This year's team, Team FUTUREbot, consisted of three members, Joel Kluske, Melissa Drogmuller and Celeste Mercado, who were assigned the mechanical design, electronic design, and programming of the robot respectively.

The final design of the robot consisted of a four-wheel holonomic wheel base, with a robotic arm for object handing, and a rotating top plate to store the medicine units. The primary sensor utilised by the robot was a Light Detection and Ranging (LIDAR), which was used for navigation and obstacle avoidance. The myRIO was required to be used as the central processing unit and was used to communicate with, and control all the electronics selected for the robot.

A state machine was implemented to navigate the robot through the course. The robot would determine which state to transition to next depending on the conditions met in the current state. The majority of the navigation was heavily dependent on the manipulation and processing of the LIDAR data.

Overall, the team was pleased with the performance of the robot. The team placed first in their Qualifiers heat, however was knocked out of the competition in the Round of 16 knockouts. Out of all the competing robots, the team's robot was found to have the most reliable object handling system, and received many compliments on the final design.

Table of Contents

Chapter 1 Introduction.....	1
1.1 National Instruments Competition Overview	1
1.2 Competition Theme.....	1
1.3 Team FUTUREbot.....	3
1.4 Project Objective and Scope.....	3
Chapter 2 NI ARC 2016 Competition Requirements.....	4
2.1 Competition Structure.....	4
2.2 Competition Rules.....	6
2.3 NI myRIO 1990.....	6
2.4 LabVIEW 2015.....	7
2.5 Milestones	8
2.5.1 Milestone 1- Completion of Online Training Course.....	8
2.5.2 Milestone 2- Sensor Actuation and Data Acquisition.....	8
2.5.3 Milestone 3- Obstacle Avoidance	9
2.5.4 Milestone 4- Navigation and Localisation	10
2.5.5 Milestone 5- Navigation and Object Handling	11
2.6 Project Timeline.....	12
Chapter 3 Literature Review and Project.....	13
3.1 Localisation.....	13
3.2 Dead-Reckoning.....	17
3.3 Feature Extraction	22
Chapter 4 Final Robot Design.....	24
4.1 Mechanical Design	24
4.1.1 Robot Base	24
4.1.2 LIDAR Positioning	25
4.1.3 Object Handling	25
4.2 Electronic Hardware	26
4.2.1 Light Detection and Ranging (LIDAR).....	26
4.2.2 Motors.....	27
4.2.3 Wheels.....	27
4.2.4 Motor Controller	28
4.2.5 Servo Motors.....	28
4.2.6 Stepper Motor	28
Chapter 5 LabVIEW Programming.....	29

5.1 Basic Motor Control.....	29
5.1.1 Motor Control Adjustments.....	31
5.2 LIDAR.....	31
5.2.1 LIDAR Data Extraction.....	32
5.3 Obstacle Avoidance	34
5.4 Wall Following.....	35
5.5 Wall Alignment.....	37
5.6 Encoders	39
5.7 Placement of Medicine Units	40
5.8 Split and Merge	41
5.9 Corner Detection	45
5.10 Wireless Communication	46
5.11 Final Program Overview.....	47
Chapter 6 Results.....	50
6.1 Pre-competition testing.....	50
6.1 Competition Results	50
Chapter 7 Future Work and Conclusion.....	51
7.1 Future Work	51
7.2 Conclusion.....	52
References	53

List of Figures

Figure 1. Hospital Competition Track [2]	2
Figure 2. NI ARC medicine unit.....	2
Figure 3. NI myRIO 1990 [4].....	7
Figure 4. myRIO MXP connectors [3]	7
Figure 5. LabVIEW Block Diagram and Front Panel [5].....	8
Figure 6. Milestone 2-Controlling the speed of a motor	9
Figure 7. Milestone 3- Obstacle avoidance.....	9
Figure 8. Milestone 4 locations A and B	10
Figure 9. Milestone 4-Navigation to B to deliver a medicine unit.....	10
Figure 10.Milestone 5-A, B, C and D locations	11
Figure 11. Milestone 5-Navigation to C to deliver second medicine unit	11
Figure 12. Kalman Filter System [7]	14
Figure 13. Cartesian co-ordinates of experimental results [7]	16
Figure 14. Percentage error of localised systems [7].....	16
Figure 15. Robot and optical mouse sensor configuration [9]	17
Figure 16. Dead-reckoning trajectory test results [9]	20
Figure 17. Average error for ten dead-reckoning tests [9].....	21
Figure 18. Comparison of line extraction methods.....	23
Figure 19. Final robot design.....	24
Figure 20. LIDAR placement.....	25
Figure 21. LIDAR scanning range [12].....	26
Figure 22. Hokuyo URG-04LX-UG01 LIDAR [11].....	26
Figure 23. 50:1 gear ratio DC motor [13]	27
Figure 24. Sabertooth 2x5 motor controller [14].....	28
Figure 25. Hitec HS-422 servo motor [15]	28
Figure 26. Model to calculate robot trajectory in the forwards direction	30
Figure 27. Rotation of wheels to achieve specific trajectories	30
Figure 28. LIDAR data extraction.....	32
Figure 29. 0° point of LIDAR	33
Figure 30. Obstacle avoidance methodology.....	34
Figure 31.LabVIEW subVI for front detection	34
Figure 32. Wall following methodology	35
Figure 33. LabVIEW wall following subVI.....	36
Figure 34. Wall alignment methodology.....	37
Figure 35. LabVIEW wall alignment subVI	38
Figure 36. Distance travelled by one revolution of a wheel [16]	39
Figure 37. LabVIEW subVI to control the distance travelled by the robot.....	40
Figure 38. LabVIEW flat sequence for medicine unit placement	41
Figure 39. LabVIEW subVI for Split function	42
Figure 40. Theory of Split and Merge line segmentation [10].....	43

Figure 41. LabVIEW subVI for Merge function.....	43
Figure 42. Original LIDAR data plot.....	44
Figure 43. LIDAR data after Split and Merge.....	44
Figure 44. Split with threshold value of 10.....	44
Figure 45. LabVIEW subVI for corner detection.....	45
Figure 46. LabVIEW subVI for wireless communication.....	46
Figure 47. Navigation using a state machine.....	47
Figure 48. Excerpt of 'wall alignment' state.....	49

List of Tables

Table 1. Competition Knockout Structure.....	5
Table 2: Competition Points System.....	5
Table 3: Project Timeline.....	12
Table 4. Motor Control.....	29

Chapter 1

Introduction

1.1 National Instruments Competition Overview

The National Instruments Robotics Competition (NI ARC) is an annually held competition which encourages students to become innovative, and develop skills within the robotics field. Student teams from the top universities across Australia and New Zealand, create and develop an entirely autonomous robot within six months to compete in the live finals competition. The competition consists of pre-defined tasks which the robot must complete, whilst abiding by a set list of rules. The competition applies robotics to real world applications, with a focus on obstacle avoidance, object handling, localisation, navigation and localisation [1].

1.2 Competition Theme

Technological advancements in the robotics field has provided many improvements and benefits in the medical field. The use of robots is now being incorporated within the medical field for various applications such as: assisting doctors with surgeries, undertaking patient check-ups from remote locations, and providing aid with the delivery of supplies within the hospital [2]. The competition is now in its sixth year, and the theme for the 2016 NI ARC was 'Hospital of the Future'. With a focus in the medical field, the robot was tasked with delivering medicine units to various locations within the hospital, whilst avoiding any static and dynamic obstacles along the way.

The hospital track is depicted in Figure 1. The course is 6m by 4m and contains three areas in which the medicine units can be delivered to; Storage, The Wards, and the Operations Theatre. The two sides, and back wall of the Storage and Operations Theatre are made of glass Perspex. This also applies to The Wards, however, each of the five wards are also divided by Perspex. The three delivery areas consist of an elevated carrier which the robot must carefully place the medicine units onto.

In addition, the track also comprises of two out of bounds areas; the Hospital Reception along with the Dynamic Obstacle Path. In the Dynamic Obstacle Path, a National Instruments (NI) Robot will be present; it will be either stationary or moving, depending on the round of the competition. Static obstacles will also be placed within the Main Area in random locations; however, they will be at least 1m away from the nearest wall, other obstacles, and vicinities of the three delivery areas [2].

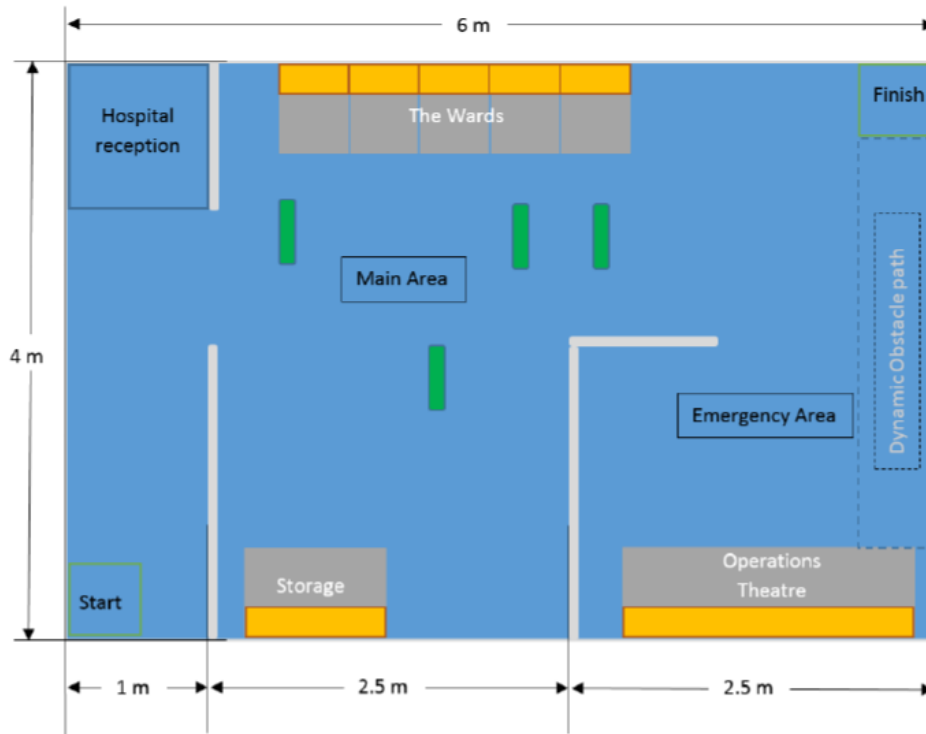


Figure 1. Hospital Competition Track [2]



Figure 2. NI ARC medicine unit

1.3 Team FUTUREbot

The 2016 team consisted of 3 members; Joel Kluske, Melissa Drogmuller, and Celeste Mercado. The work load was divided into three categories; mechanical, electronic, and software (programming). Joel was assigned the mechanical section where he was responsible for designing and constructing the robot. Melissa was assigned with the electronics section where she was in charge of selecting the required electronics for the robot, such as sensors and motor controllers, and, in addition, how these electronics would be interfaced. I was tasked with the programming, which consisted of determining how to navigate and control the robot using the selected electronics. In addition, Ben Illman, who was a participant in the 2015 NI ARC assisted as the group's mentor. As programming was a large section of the project, Melissa also assisted with some sections of programming. Although each group member was assigned a specific role, team members worked cohesively and were not hesitant in discussing ideas with one another.

1.4 Project Objective and Scope

The objective of this project is to design, build, and program an autonomous robot which abides by the competition rules, and accomplishes the set of tasks as defined by NI.

This thesis aims to provide an overview of the rules and requirements of the 2016 NI ARC, and a brief insight into the mechanical design and hardware involved in creating the robot. However, the main scope of the thesis is to explain the algorithms and methods implemented to control and navigate the robot through the course to achieve the competition goals.

Chapter 2

NI ARC 2016 Competition Requirements

2.1 Competition Structure

The competition was structured as a knockout tournament. The basis for each round required the robot to begin within the Start area, then navigate its way to The Wards, then to the Operations Theatre, and finally park itself completely within the Finish area.

Each round of the competition becomes progressively difficult. The number of medicine units required to be delivered to both The Wards and Operations Theatre gradually increase, the amount of static obstacles also increase, and the NI Robot gradually becomes more active. Table 1 explains the requirements and nature of each round. Delivering medicine units to Storage is not a requirement, however delivering the stated number of medicine units to The Wards as per Table 1 is mandatory before making deliveries to the Operations Theatre. However, only one medicine unit per ward is permitted; delivering multiple units to a single ward does not contribute to additional points being awarded. The points system details are described in Table 2, and the final score is calculated via applying the equation below (1)[2].

$$\textit{Total Points} = [(150 \textit{ sec} - \textit{ time taken}) \times \textit{ multiplier}] + \textit{ points added} - \textit{ points deducted} + \textit{ bonus points} \quad (1)$$

Table 1. Competition Knockout Structure [2]

Name of Round	Round	Format	Time Limit (mins)	# of Medicine Units	Min # of units in wards before Operations Theatre Delivery	# Obstacles in the Main Area	NI Robot in Dynamic Obstacle Path
Qualifiers	1	Group Stages (Best of 3)	2.5	2	1	Max 3	Stationary
Round of 16	2	Knockout	2.5	2	1	Max 3	Stationary
Quarter-Finals	3	Knockout	2.5	3	2	Max 4	Moving slowly in one direction
Semi-Finals	4	Knockout	2.5	4	2	Max 5	Moving fast in one direction
Playoffs for 3rd place	5	Knockout (Best of 3)	2.5	5	3	Max 7	Moving fast omnidirectional
Championship Round	6	Knockout (Best of 3)	2.5	6	3	Max 7	Moving fast omnidirectional

Table 2: Competition Points System

Points Awarded for:	Quantity Awarded	Point Deductions for:	Quantity Deducted	Round	Multiplier
Delivery to Storage	+75	Bumping into Walls/Obstacles	-25	Qualifiers	1
Delivery to Each Ward	+150	False Starts	-25	Round of 16	1.5
Delivery to Operation Theatre	+300	Crossing Dynamic Obstacle Path	-30	Quarter-Finals	2
(Bonus Points) Finishing without any obstacle collisions	+200	Collision with NI Robot	-300	Semi-Finals	2.5
(Bonus Points) for Wi-Fi messaging to server	+200	Entering Hospital Reception	-25	Playoffs for 3rd place and Championship Round	3

2.2 Competition Rules

National Instruments provided a set of rules in which the design and programming of the robot must abide by to qualify for the competition. Additional rules which were not previously discussed are as follows:

- The robot must be entirely autonomous, self-powered, and must move i.e. a conveyor system cannot be used.
- The NI myRIO provided must be used and utilised as the main Central Processing Unit (CPU). Other CPUs may be used if embedded within a sensor or actuator for signal conditioning purposes.
- The LabVIEW software provided must be used for the majority of code written and implemented.
- The robot must include a hardware switch to trigger the robot to start, and also an emergency switch to stop the robot.
- The robot must not damage the medicine units, as well as any components of the track. At the competition judges' discretion, the robot may be disqualified if deemed destructive.

2.3 NI myRIO 1990

As identified in the Chapter 2.2, a competition requirement was for the NI myRIO provided, to be utilised as the robot's main CPU. The myRIO 1990 is an embedded portable hardware device which is designed for students to utilise for a wide variety of projects. It has a dual-core ARM Cortex -A9, real time-processing capabilities, and customizable Xilinx FPGA [3]. It is a reconfigurable input/output (I/O) device containing two identical sets of 33 pin MXP connectors, and a 20 pin Mini System Port (MSP). It also contains an accelerometer, USB host port, USB device port, and four programmable LEDs. The myRIO also has wireless capabilities, such that it can connect to a server and connect to a host computer via Wi-Fi.

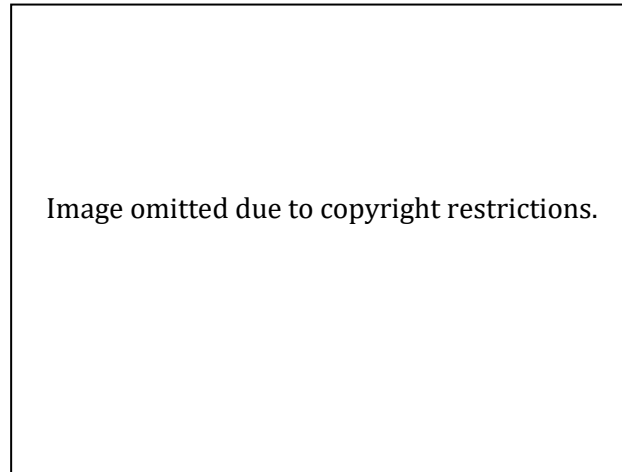


Figure 3. NI myRIO 1990 [4]

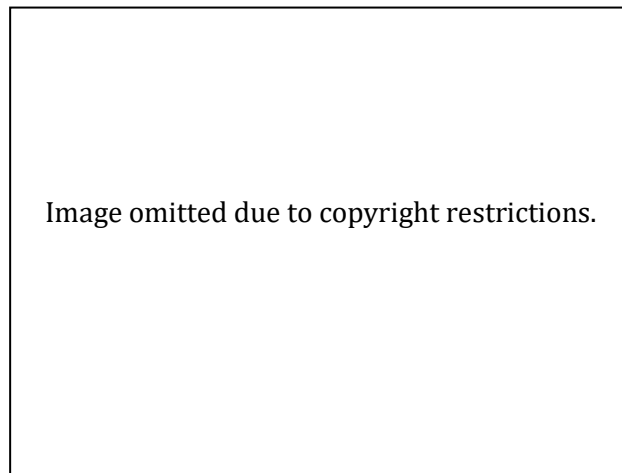


Figure 4. myRIO MXP connectors [3]

2.4 LabVIEW 2015

One of the rules of the competition is that the majority of the programming must be written and implemented using LabVIEW. As opposed to text based languages such as, C, C++ and Java, LabVIEW is a graphical software environment. It uses wires and icons to represent the flow of data through the program.

LabVIEW uses Virtual Instruments (VIs) to write the programs. A VI consists of a Front Panel and a Block Diagram. The Front Panel is the graphical user interface (GUI) which contains the controls and indicators of the program. It allows the user to interact with the program in real time and/or monitor the outputs of the program.

The Block Diagram is where the graphical source code is implemented [5]. Data is processed by using wires to connect the inputs and outputs of components within the block diagram.

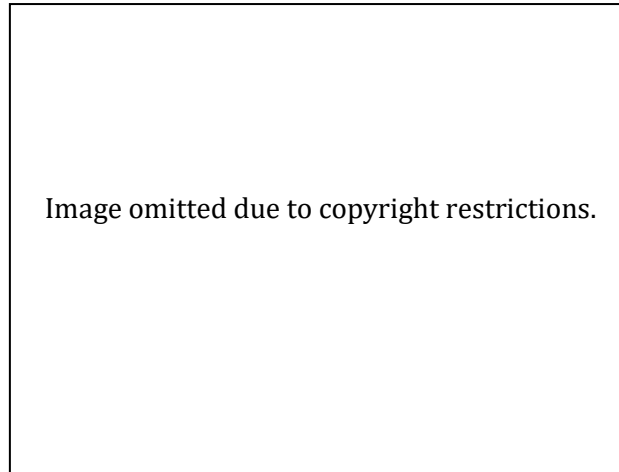


Figure 5. LabVIEW Block Diagram and Front Panel [5]

2.5 Milestones

In the lead up to the competition, five milestones were required to be completed. Each milestone required a deliverable to be assessed by the NI ARC team.

2.5.1 Milestone 1- Completion of Online Training Course

The first milestone required two members of the team to complete Core 1 and Core 2 of the online NI LabVIEW training course by April 27, 2016. These online courses aimed to introduce and familiarise the competition participants to different features of the LabVIEW environment.

Each core involved watching a series of videos broken down into modules, each covering a different aspect of LabVIEW. A quiz question was then required to be answered at the end of each module, along with completing a final quiz at the end of each course. As proof of completion of the online course, the Core 1 and Core 2 course certificates were to be provided to the NI ARC team.

2.5.2 Milestone 2- Sensor Actuation and Data Acquisition

Milestone 2 required a submission of a project proposal of 300-500 words, along with a demonstration of the use of myRIO and LabVIEW, to either: control at least one actuator or motor, or acquire data from at least one sensor. The due date for this milestone was May 9, 2016.

The team submitted a video demonstrating the control of a DC motor using an Infra-Red (IR) sensor, where the value read from the sensor would control the speed of the motor.

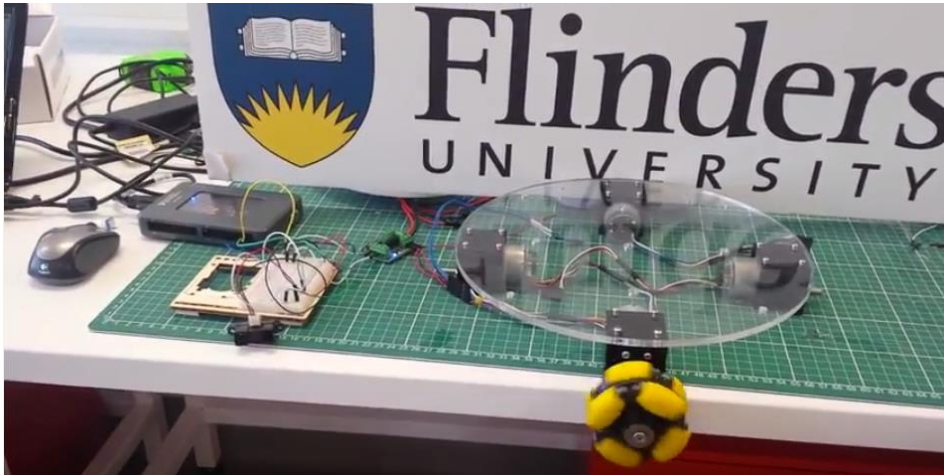


Figure 6. Milestone 2-Controlling the speed of a motor

2.5.3 Milestone 3- Obstacle Avoidance

Milestone 3 was due on June 8, 2016. This milestone required a preliminary design and construction of the robot with obstacle avoidance implemented. A video of the robot moving autonomously whilst avoiding obstacles was submitted. The team set up a replica of the competition track, and placed obstacles in random positions for the robot to avoid.

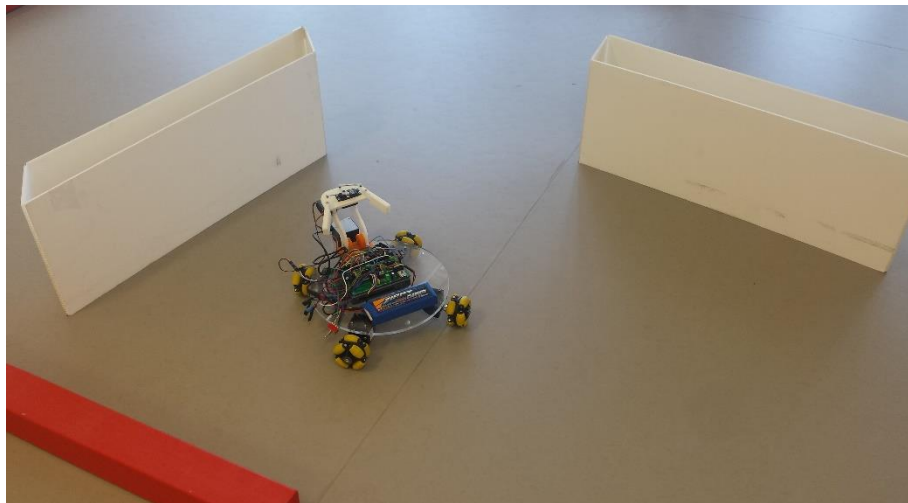


Figure 7. Milestone 3- Obstacle avoidance

2.5.4 Milestone 4- Navigation and Localisation

The due date for milestone 4 was July 25, 2016. The requirement for this milestone was for the robot to demonstrate navigation/localisation and obstacle avoidance. The robot was to begin loaded with a medicine unit at position A, then navigate to location B. At location B the robot was to carefully deliver the medicine unit onto an elevated platform. Refer to Figure 8 for a visual representation of locations A and B. A video submission of the robot completing the task was required, along with a submission of the code and description of the VIs used within the code.



Figure 8. Milestone 4 locations A and B



Figure 9. Milestone 4- Navigation to location B to deliver a medicine unit

2.5.5 Milestone 5- Navigation and Object Handling

The final milestone required the demonstration of a combination of obstacle avoidance, navigation and object handling, and was also a requirement to qualify for the final competition. The robot was to begin at location A with two medicine units loaded onto it, then move to location B and carefully deliver the first block onto an elevated platform. The robot must then navigate itself to location C whilst avoiding obstacles, where it must carefully deliver the second medicine unit onto an elevated platform. After delivering both units, the robot must park in location D whilst avoiding obstacles, to complete the task. Refer to Figure 10 for a visual representation of locations A, B, C and D.

A video submission of the robot completing the task, along with the code and description of the VIs utilised was to be submitted by August 29, 2016.



Figure 10. Milestone 5-locations A, B, C and D

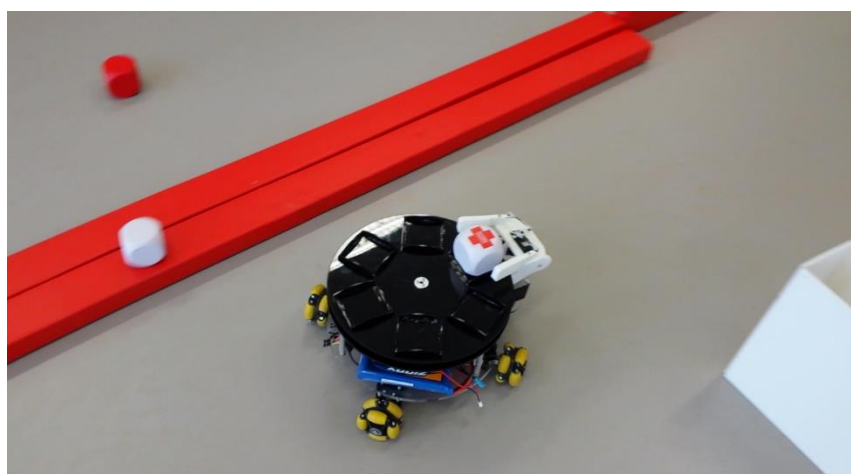


Figure 11. Milestone 5- Navigation to location C to deliver second medicine unit

2.6 Project Timeline

The milestones discussed in Chapter 2.5 provided the timeline for the project. Table 3 provides a summary of the of the deadlines and deliverables for the project.

Table 3: Project timeline

Task	Due Date (2016)	Deliverables
Competition begins	April 11	-
Milestone 1	April 27	-LabVIEW Core 1 and Core 2 training certificates
Milestone 2	May 9	-Project Proposal -Video demonstration of motor/ actuator control or data acquisition from a sensor
Milestone 3	June 8	-Prototype of robot -Video submission of prototype demonstrating obstacle avoidance
Milestone 4	July 25	-Video submission of robot demonstrating navigation/localisation and obstacles avoidance -Code submission -VI descriptions
Milestone 5	August 28	-Video submission of robot demonstrating navigation/localisation and obstacles avoidance and object handling -Code submission -VI descriptions
Live Competition	September 26	-Construction of final robot design -Program to compete in competition

Chapter 3

Literature Review and Project

There are three questions which formulate the problem of robot navigation;

- **Where am I?** - For the robot to make useful decisions, it must know its location. Localisation is thus used to determine where the robot is located.
- **Where am I going?** - The robot must know where it is going in order to achieve its task. Identifying its goal is known as goal recognition.
- **How do I get there?** - Once the robot knows its location and where it wants to go, it must decide on how it will get there. Deciding on how it will get to its desired location is known as path planning [6].

In this chapter, localisation and feature extraction techniques will be explored to provide insight into which algorithms could be implemented for the project to navigate the robot through the course.

3.1 Localisation

The position of a mobile robot is one of the most important state parameters; the robot must know its position at every moment. This is known as localisation of the robot. A robot perceives its environment through the use of sensors which provide the robot with data about its environment. The robot can then use this information to determine its position in relation to the environment. However, robot localisation is affected due to environmental noise and interference. These sources of error cause a loss in precision and accuracy when determining the robot's position [7].

A conference paper by Sangale, V & Shendre, A 2013, uses a multiple sensor fusion technique with the Extended Kalman Filter (EKF) to attain precise localisation, where the EKF combines readings from multiple sensors to predict the robot's next state. The paper discusses how odometry and gyroscope measurements are used to accurately localise the robot.

The Kalman Filter (KF) uses a "Gaussian probability density representation of robot position and scan matching for localisation" [7]. Where a single well-defined Gaussian probability density function represents the robot's belief state. Although the KF is precise and efficient in tracking the robot from an initial known position, the filter itself is not sufficient for global localisation. Another issue with the KF is that it assumes the system model is perfect, and noise is white Gaussian [7].

Figure 12 depicts the basic logic of the Kalman Filter. It is a recursive algorithm which “estimates the state of a noisy linear dynamic system” [7] using the system measurements it receives.

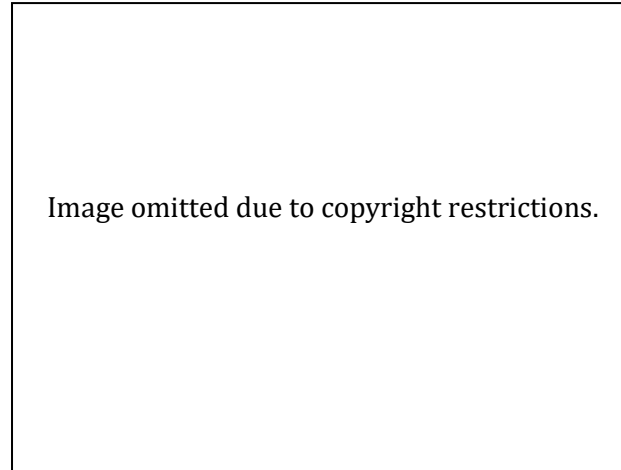


Figure 12. Kalman Filter System [7]

The EKF overcomes the problems of the basic KF, as it can be applied to non-linear systems by using partial derivatives to linearize the estimation around the current estimate [7]. There are three stages to the EKF algorithm.

Initialisation. The EKF is initialised at time step 0 with a posterior state estimate \hat{x}_0^+ and uncertainty P_0^+ .

Prediction. At every time step, the EKF predicts the current step by propagating the previous state and uncertainty of the system. The prediction equations are described by equations (2) and (3) [7].

$$\hat{x}_k^- = f(\hat{x}_{k-1}^+) \quad (2)$$

$$P_k^- = A_k P_{k-1} A_k^T + Q_{k-1} \quad (3)$$

A_k is the Jacobian Matrix containing the partial derivatives of the function $f(\cdot)$ with respect to the states previous posterior state estimate \hat{x}_{k-1}^+ [7].

$$A_k = \left. \frac{\partial f(x)}{\partial x} \right|_{x=\hat{x}_{k-1}^+} \quad (4)$$

Correction. The prior state estimate is corrected by the EKF with a full measurement z_k . The correction equations are described by equations (5) to (6) [7].

$$K_k = P_k^- H_k^T (H_k P_k^- H_k^T + R_k)^{-1} \quad (5)$$

$$\hat{x}_k^+ = \hat{x}_k^- + K_k (z_k - h(\hat{x}_k^-)) \quad (6)$$

$$P_k^+ = (I - K_k H_k) P_k^- \quad (7)$$

H_k is the Jacobian Matrix containing the partial derivatives of the function $h(\cdot)$ with respect to the state previous posterior state estimate \hat{x}_k^- [7].

$$H_k = \left. \frac{\partial f(x)}{\partial x} \right|_{x=\hat{x}_k^-} \quad (8)$$

The experiment was conducted using two methods: firstly, using only two wheel encoders, and secondly with a gyroscope and two wheel encoders combined by the EKF. The results of the experiment can be found in Figure 13, and the corresponding percentage errors are shown in Figure 14. The results show that the average percentage error is considerably reduced when localisation with sensor fusion by EKF is utilised. As the EKF uses a comparison of different sensors, it allows the system to not be completely disturbed; the system is able to follow the sensor which is less erroneous. Overall, it can be concluded that the use of sensor fusion via EKF yields an acceptable state estimation of the robot, where nominal errors lie within the tolerance level of the system [7].

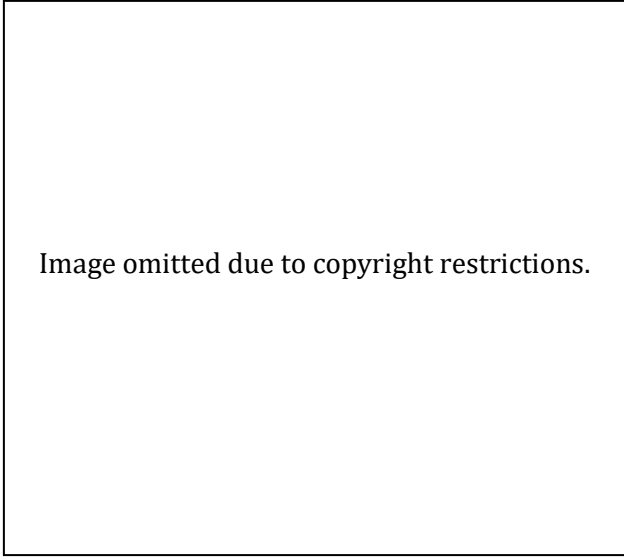


Image omitted due to copyright restrictions.

Figure 13. Cartesian co-ordinates of experimental results [7]

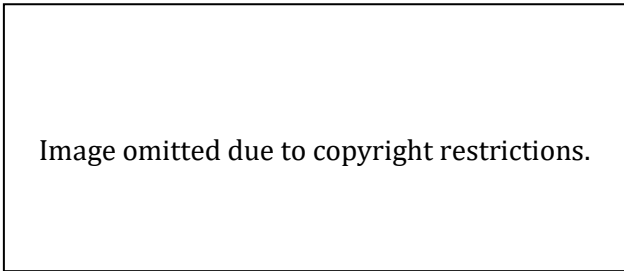


Image omitted due to copyright restrictions.

Figure 14. Percentage error of localised systems [7]

The experimental results have shown that the EKF is a worthy consideration for the implementation of robot localisation for the competition. In particular, the study has helped identify the technique of sensor fusion to provide more accurate localisation. It has highlighted the benefits of integrating two sensors into the EKF such that it has a greater correcting capability in comparison to only using the wheel encoder data for localisation.

3.2 Dead-Reckoning

Dead-reckoning is a relative positioning localisation method used to estimate the positioning of the robot. It utilises only its internal sensors such as encoders and inertial measuring unit (IMU) to estimate the relative position of the robot. As dead-reckoning does not rely on external signals, its major advantage over absolute positioning methods is that it is: much simpler to implement, lower in cost due to reduction on hardware and software needs, eliminates the need to implement landmarks, and is faster in calculating the robot's position in real time. However, one disadvantage of this method is that the accuracy of estimations decreases due to the accumulation of transformation errors and slippage as the robot's movement increases [8].

A journal paper by Sekimori, D & Miyazaki, F 2007, showed an effective implementation of a dead-reckoning system by utilising optical mouse sensors, where the robot's movements are tracked from the floor using optical mouse sensors. Depending on the conditions of the flooring, and amount of shaking of the robot, there will be some expected errors in the read movements. Hence, to eliminate this problem, multiple optical mouse sensors were utilised to compare sensor values. The optical mouse measures non-contact movements. A sensor irradiates the floor with an LED through a lens, and the sensor captures the floor images. The optical mouse sensor then processes the pictures and transforms them into distance information [9].

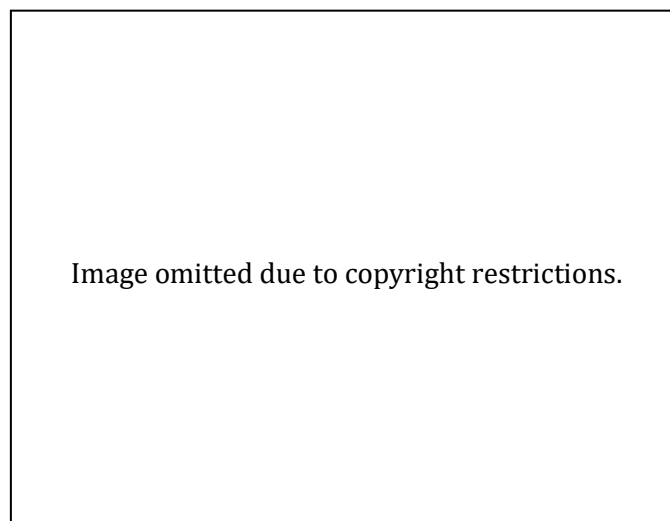


Figure 15. Robot and optical mouse sensor configuration [9]

Figure 15 depicts the configuration of robot and optical mouse sensors. Movement of the robot (translational, and rotational movement) on a plane is measured using two optical mouse sensors. The relation movement measured by each sensor, $[\Delta\xi_i, \Delta\eta_i]^T$, $[\Delta\xi_j, \Delta\eta_j]^T$, and the movement $[\Delta x, \Delta y, \Delta\theta]^T$ of the robot centre is described by equations (9) and (10) [9].

$$\begin{bmatrix} C\varphi_i & -S\varphi_i \\ S\varphi_i & C\varphi_i \end{bmatrix} \begin{bmatrix} \Delta\xi_i \\ \Delta\eta_i \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \Delta\theta \begin{bmatrix} -diS\varphi_i \\ -diC\varphi_i \end{bmatrix} \quad (9)$$

$$\begin{bmatrix} C\varphi_j & -S\varphi_j \\ S\varphi_j & C\varphi_j \end{bmatrix} \begin{bmatrix} \Delta\xi_j \\ \Delta\eta_j \end{bmatrix} = \begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} + \Delta\theta \begin{bmatrix} -djS\varphi_j \\ -djC\varphi_j \end{bmatrix} \quad (10)$$

The equations are rearranged to provide equation (11), which is described by equations (12) to (14) [9].

$$\mathbf{A}\mathbf{u} = \mathbf{a} \quad (11)$$

$$\mathbf{u} = [\Delta x, \Delta y, \Delta\theta]^T \quad (12)$$

$$\mathbf{A} = \begin{bmatrix} 1 & 0 & -diS\varphi_i \\ 0 & 1 & diC\varphi_i \\ 1 & 0 & -djS\varphi_j \\ 0 & 1 & djC\varphi_j \end{bmatrix} \quad (13)$$

$$\mathbf{a} = \begin{bmatrix} \Delta\xi_i C\varphi_i - \Delta\eta_i S\varphi_i \\ \Delta\xi_i S\varphi_i + \Delta\eta_i C\varphi_i \\ \Delta\xi_j C\varphi_j - \Delta\eta_j S\varphi_j \\ \Delta\xi_j S\varphi_j + \Delta\eta_j C\varphi_j \end{bmatrix} \quad (14)$$

The squared error of movement E_{ij} is expressed by equation (15) [9].

$$E_{ij} = \sum_{p=1}^4 (Ap1\Delta x + Ap2\Delta y + Ap3\Delta\theta - ap)^2 \quad (15)$$

To determine the movement of the robot u which results with a minimised square error E_{ij} , the following equation (16) [9] is used.

$$u = A^{-1}a \quad (16)$$

After determining the movement u of the robot, dead-reckoning can be computed via equation (17) [9].

$$\begin{bmatrix} X_t \\ Y_t \\ \Theta_t \end{bmatrix} = \begin{bmatrix} X_{t-1} + \Delta x C \Theta_{t-1} - \Delta y S \Theta_{t-1} \\ Y_{t-1} + \Delta x S \Theta_{t-1} + \Delta y C \Theta_{t-1} \\ \Theta_{t-1} + \Delta \theta \end{bmatrix} \quad (17)$$

There are various factors which contribute to incorrect measurements by the optical mouse sensor such as: robot speed, floor conditions, and shaking. When monitoring only the error of one optical mouse sensor, error cannot be detected. A method was proposed to calculate the robot's movements by comparing and selecting reliable sensor readings. The accuracy of a measurement is evaluated based on equation (18) [9].

$$r_i = \sum_{\substack{j=1 \\ j \neq i}}^N \delta_{ij}, \quad \delta_{ij} = \begin{cases} 1 & (E_{ij} \leq E_{th}) \\ 0 & (E_{ij} > E_{th}) \end{cases} \quad (18)$$

Where r_i is the reliability of the optical mouse sensor m_i , N is the number of optical mouse sensors, E_{ij} is the calculated squared error and E_{th} is the threshold. Reliability is calculated for each optical mouse sensor m_α, m_β etc., and a threshold value r_{th} is used to select the sensors with high reliability [9].

The movement of the robot is now calculated with the equation (19) [9].

$$u = B^{-1}b \quad (19)$$

$$B = \begin{bmatrix} 1 & 0 & -d\alpha S\varphi\alpha \\ 0 & 1 & d\alpha C\varphi\alpha \\ 1 & 0 & -d\beta S\varphi\beta \\ 0 & 1 & d\beta C\varphi\beta \\ \dots & \dots & \dots \end{bmatrix} \quad (20)$$

$$b = \begin{bmatrix} \Delta\xi\alpha C\varphi\alpha - \Delta\eta\alpha S\varphi\alpha \\ \Delta\xi\alpha S\varphi\alpha + \Delta\eta\alpha C\varphi\alpha \\ \Delta\xi\beta C\varphi\beta - \Delta\eta\beta S\varphi\beta \\ \Delta\xi\beta S\varphi\beta + \Delta\eta\beta C\varphi\beta \\ \dots \end{bmatrix} \quad (21)$$

The dead-reckoning method was tested using a robot driven with three omnidirectional wheels. A camera was installed in the ceiling to provide information on the robot's true trajectory, and tests for dead-reckoning using the wheel encoders, two optical sensors, and four optical sensors were conducted and compared for two different speeds [9]. Trajectory results for each speed can be observed in Figure 16.

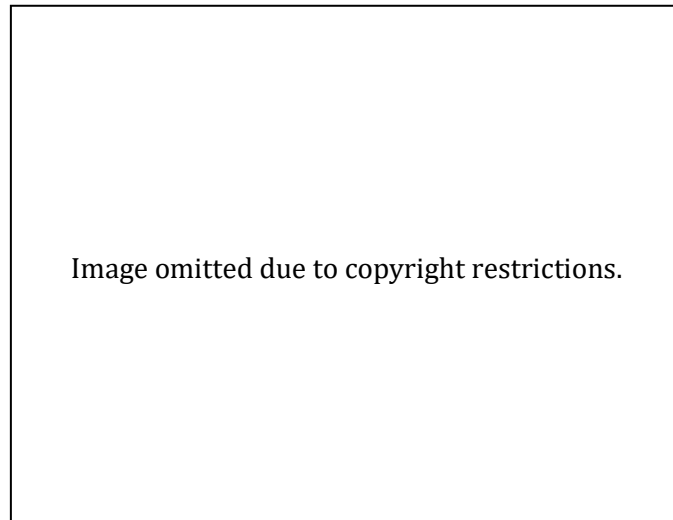


Figure 16. Dead-reckoning trajectory test results [9]

These dead-reckoning tests were performed ten times whilst keeping the same conditions. Results from the ten tests resulted to similar results as depicted in Figure 16, hence the dead-reckoning method proposed proved to be confirmed. The average of the estimated errors is depicted in Figure 17, and show that the use of four optical mouse sensors provides the most accuracy when following the actual robot trajectory [9].

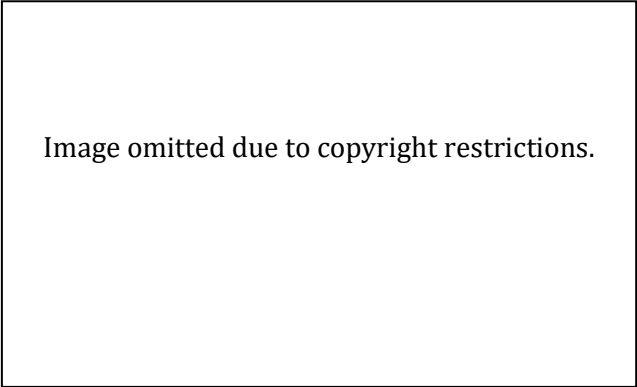


Image omitted due to copyright restrictions.

Figure 17. Average error for ten dead-reckoning tests [9]

As the known path for the robot can be pre-identified for the NI ARC, dead-reckoning is a plausible method which could be utilised. The experimental results from the proposed dead-reckoning method have shown that the robot can be made to move accurately to pre-set locations with the use of optical mouse sensors. Optical mouse sensors would be beneficial to provide a way to accurately track the robot's movements without being influenced by floor friction, which is a main contributing factor to errors when using wheel odometry. Overall, based on the results presented, it can be concluded that the use of four optical mouse sensors is required if the robot is to be driven at faster speeds. Since fastest time to complete competition objectives is an element of the competition, it is most likely four optical sensors will be required to be utilised if accurate dead-reckoning is wanted.

3.3 Feature Extraction

Odometry data alone is not sufficient enough to localise the robot due to the positioning error associated with it. A solution to this can be implemented via the use of a range of sensors such as a sonar, LIDAR or infrared. One of the most popular sensors is the 2D laser rangefinder, which is used for localisation, dynamic map building and collision avoidance. It has an advantage to other sensors such as: providing more accurate measurements, high sampling rate, higher angular resolution and has good range distance and resolution [10].

A major issue in robotic localisation is accurately matching sensor data to information on a priori map, or to previous information collected. Two matching techniques used in mobile robotics are: point based matching, and feature based matching. However, feature-based algorithms are preferred as they are more compact, thus require less memory storage whilst providing accurate information, and are also more efficient [10].

A conference article written by Nguyen *et al.* 2005 evaluates six popular line extraction algorithms in mobile robotics, and compares each method to one another to highlight the advantages and disadvantages of each algorithm. Comparisons between each method will allow for easy identification as to which methods are preferable for specific applications. The six-line extraction methods tested were, Split and Merge, Line Regression, Incremental, Random Sample Consensus (RANSAC), Hough-Transform (HT), and Expectation-Maximization (EM).

The paper provides an overview of the algorithm associated with each extraction method, and details the process of how the experiments were conducted. Algorithm specific parameters were tuned based on experimental results, such that the best performance of each algorithm could be utilised for comparison purposes. The correctness of each algorithm was determined by comparing them to manually extracted lines from the scan defined as “truth lines” [10]. To compare the results of the six algorithms, four quality measures were defined and evaluated: complexity, speed, correctness and precision. Figure 18 depicts the results of the six-line extraction algorithms combined with the clustering algorithm and the other five are the basic versions of the algorithms. The clustering algorithm was integrated to filter out noisy points and divide the raw scan into clusters. This allowed the scan to be segmented into contiguous point clusters [10].




Image omitted due to copyright restrictions.

Figure 18. Comparison of line extraction methods

In terms of speed: Split and Merge, Incremental and Line Regression are the top three fastest, with Split and Merge being the superior. Incremental based algorithms have shown to perform best in terms of correctness, as they have a low number of false positives which is important for simultaneous localization and mapping (SLAM). Due to a higher percentage in true positives, Split and Merge combined with the clustering algorithm would be best for localisation with a priori map. Whilst having slow speed and bad correctness, RANSAC, HT, and EM+Clus algorithms are able to “produce relatively more precise lines” [10]. Overall, Split and Merge, and Incremental would be the best candidates for SLAM, due to their fast speed and correctness. For real time applications, Split and Merge would be the ultimate algorithm due to its superior speed, and would also be a preferred choice for localisation with a priori map where false positives are not so important [10].

Feature extraction would be a useful feature to implement for the NI ARC competition. The primary sensor which will be utilised on our robot will be a LIDAR which will essentially provide similar information as the laser sensor used in the experiment. By implementing a feature extraction algorithm, walls and obstacles can be easily identified, as well as provide the robot with information about its location by identifying certain features of the track. As the experiments conducted by Nguyen et al. 2005 ensured each algorithm was compared to one another based on their best performances, and with as many common parameters as possible, the results provided good and clear comparisons as to which algorithms would provide the best performance for certain applications, along with the downfalls of each algorithm. Due to its fast computational time and good correctness, Split and Merge would be the most suitable algorithm for the requirements of the competition, as the robot can identify features within the track in real time.

Chapter 4

Final Robot Design

In this chapter the final robotic design will be briefly discussed by breaking down the design into its mechanical and electronic design.

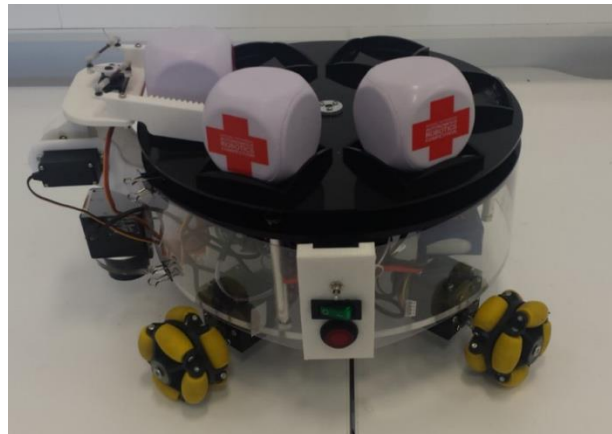


Figure 19. Final robot design

4.1 Mechanical Design

A brief overview of the final mechanical design will be provided in this chapter. However, greater detail can be found in the final project report written by Joel Kluske who was responsible for the mechanical design of the robot.

4.1.1 Robot Base

The robot was designed with a circular base driven by four omnidirectional wheels spaced 90° apart. Although the previous team encountered complications with a four wheeled design; where only three wheels would be in contact with the floor, the weight of the robot was substantial enough to ensure all four wheels remained in contact with the ground.

4.1.2 LIDAR Positioning

The placement of the LIDAR required to be low enough to sense the walls, elevated carriers and surrounding obstacles, whilst high enough such that it did not falsely detect the ground as an obstacle (due to the LIDAR's beam diameter). If the LIDAR was placed the right way up, it would fail to detect the walls and obstacles of the track. Hence, the optimal position of the LIDAR was to mount it upside down. Due to the wheel configuration, the LIDAR required to be extruded out from the robot's base as to utilise the full detection range of the LIDAR.

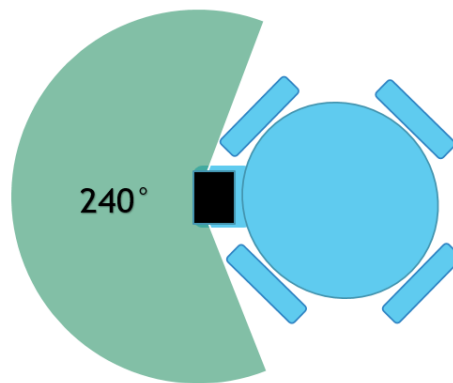


Figure 20. LIDAR placement

4.1.3 Object Handling

The maximum number of medicine units the robot was required to carry was six. A rotational top was opted for to achieve this requirement. A rotating top meant the footprint of the robot could remain unchanged; as six blocks could be evenly spaced onto a disk of equal size as the base of the robot. A border/frame was placed around each of the six unit slots to ensure the units would remain secure whilst the robot was moving. A simple pick and place mechanism was then chosen for the placement of the medicine units, where a robotic arm with linear movement and a pair of grippers could easily grab each block from the rotational top. This placement method ensured the medicine units would be carefully and stably placed onto the carrier without them rolling off.

4.2 Electronic Hardware

This chapter aims to provide an overview of the electronics utilised which are related to the programming of the robot. More detail to the robot's electronic design can be found in the thesis written by Melissa Drogmuller.

4.2.1 Light Detection and Ranging (LIDAR)

The only sensor utilised by the robot was the Hokuyo URG-04LX-UG01. This LIDAR has a detection area of 240°, with an angular resolution of 0.36°, and produces approximately 683 data points per scan. It completes a single scan every 100msec with a scanning range of 0.02 m to 4m. At 4m, the beam diameter is approximately 40mm [11].

The LIDAR was selected for various reasons. Firstly, its large detection area and precision would provide adequate vision for the robot without the need for additional sensors. Secondly, it could be connected to, and powered through the myRIO without the need for a separate power source. This allowed for ease of connectivity and decrease in electronic footprint. Thirdly, LabVIEW contained pre-defined VIs for this particular LIDAR, providing a simple way to interface the LIDAR, and allow the LIDAR data to be easily extracted. Lastly, the Hokuyo URG was readily available from previous years.

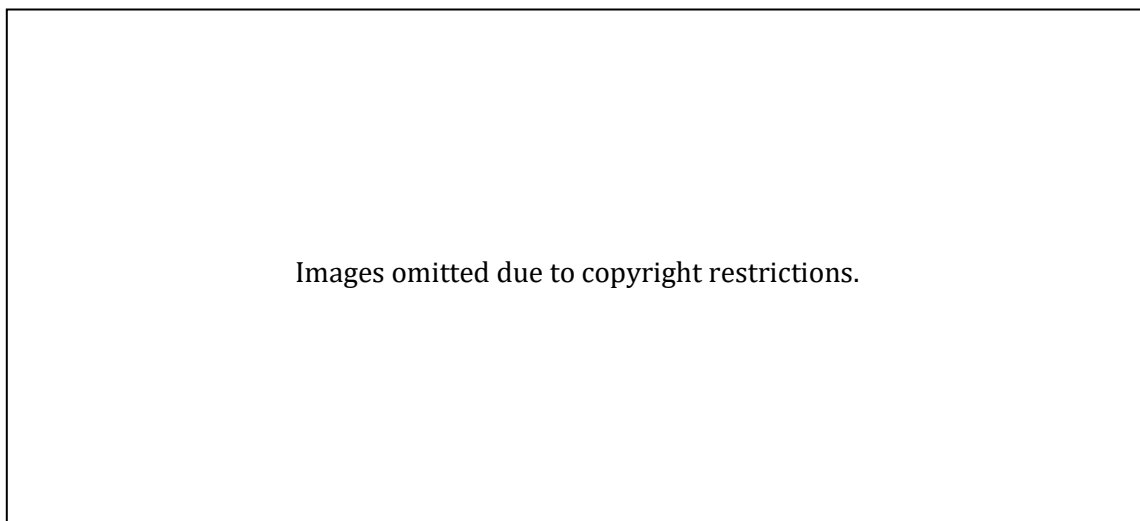


Figure 22. Hokuyo URG-04LX-UG01 LIDAR [11]

Figure 21. LIDAR scanning range [12]

4.2.2 Motors

The motors selected to drive the robot were 12 Volt brushed DC Pololu Metal Gearmotors with a 50:1 gearbox ratio, free run current of 300Ma and stall current of 5A. The motor also has integrated quadrature encoders which have a “resolution of 64 counts per revolution of the motor shaft” [13]. This equates to an output of 3200 counts per revolution of the gearbox shaft.



Figure 23. 50:1 gear ratio DC motor [13]

4.2.3 Wheels

With the course track known to be a flat vinyl surface, wheeled locomotion was selected. A locomotion system with high manoeuvrability, precision and speed was required. Omnidirectional wheels were selected as the robot wheels, due to their ability to move in any direction regardless of the robot’s orientation. The Flinders team from the 2015 NI ARC also used omnidirectional wheels and were 45mm in diameter. However, the sizing of these wheels caused the robot to move relatively slowly compared to other robots in the competition. In addition, its small size also caused the robot to be caught in an unexpected ‘bump’ in the track. To improve from the previous year, the wheel sizing was increased to 70mm in diameter. This would in turn also increase the robot’s speed by a factor of 1.5.

4.2.4 Motor Controller

The motor controllers selected were the Sabertooth 2x5. They are capable of controlling two independent motors providing up to 5A per channel, and peak loads up to 10A per channel for a couple of seconds [14]. The Sabertooth has 4 four operating modes: Analog Input, R/C Input, Simplified Serial and Packetized serial. These modes are selected by configuring the six DIP switches to either ON or OFF as required.

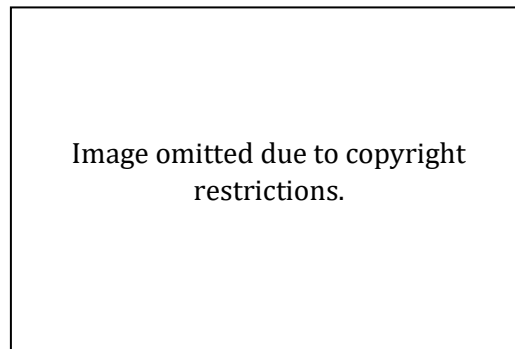


Figure 24. Sabertooth 2x5 motor controller [14]

4.2.5 Servo Motors

For the robotic arm movement two Servo-Hitec HS-422 were used. The servo has a 180° rotational range, and operates in-between 4.8V and 6V [15]. These servos were used to control the robot arm up and down, along with opening and closing the gripper.

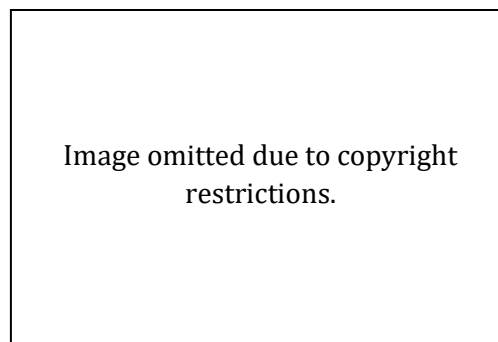


Figure 25. Hitec HS-422 servo motor [15]

4.2.6 Stepper Motor

To control the rotational top of the robot, a stepper motor was utilised with a full revolution of 360°. This would provide precision when rotating the rotational plate which holds the medicine units.

Chapter 5

LabVIEW Programming

5.1 Basic Motor Control

A simple technique was found to control the direction of motion of the robot using the Sabertooth motor controllers. By sending a voltage between 0V to 5V to the motor controllers, the speed and direction of the motors could be controlled. A voltage from 0V to 2.5V would rotate the motors in the reverse direction, with 0V being the maximum speed then decreasing to a stop at 2.5V. Alternatively, a voltage from 2.5V to 5V would rotate the motors in the forwards direction, with 5V corresponding to the maximum speed, with the speed decreasing to a stop at 2.5V.

Table 4. Motor Control

Direction	Voltage(V)
Reverse	0 → 2.5
Stationary	2.5
Forwards	2.5 → 5

Using Figure 26 as an example, the overall trajectory of the robot can be identified. The red arrows are the directions of the overall force on the wheel, the yellow arrows represent the x-component of force, and the purple arrows represent the y-component of force. To determine the direction the robot will be projected, the total force of each wheel are summed together. This is given by equation (22). The length of each force represents the magnitude of force, where in this particular example, the force applied to each wheel are equivalent. Equation (23) and (24) describes the total magnitude of force of the x and y components respectively. It can be seen that the x-components cancel each other out, and what is left is the summation of the y-components which are all in the forwards direction, hence the trajectory of the robot is forwards.

$$\vec{F}_T = \vec{F}_1 + \vec{F}_2 + \vec{F}_3 + \vec{F}_4 \quad (22)$$

$$F_x = \sum_{n=1}^4 F_n \times \sin(a_n) \quad (23)$$

$$F_y = \sum_{n=1}^4 F_n \times \cos(a_n) \quad (24)$$

By applying these principles, the robot was able to be driven in any direction by applying the correct voltages to each wheel. In the competition, a total of six directions were utilised to navigate the robot around the course. These directions are depicted in Figure 27, where the arrows portray the direction of rotation required for each wheel to achieve each a certain trajectory. For these particular cases, the magnitude of force (or speed) of each wheel are equivalent.

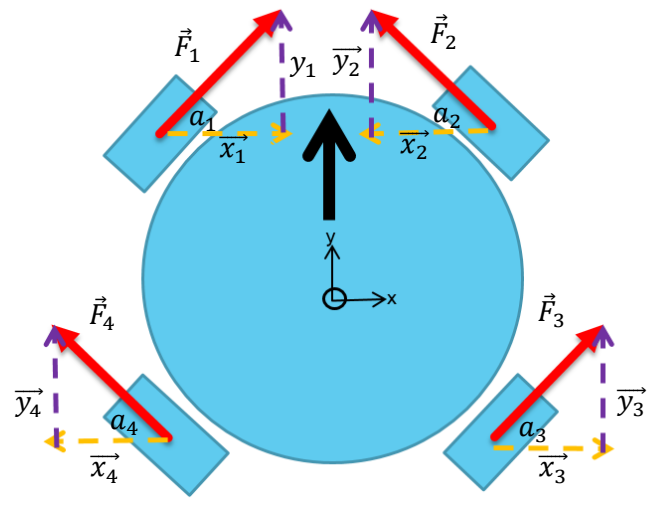


Figure 26. Model to calculate robot trajectory in the forwards direction

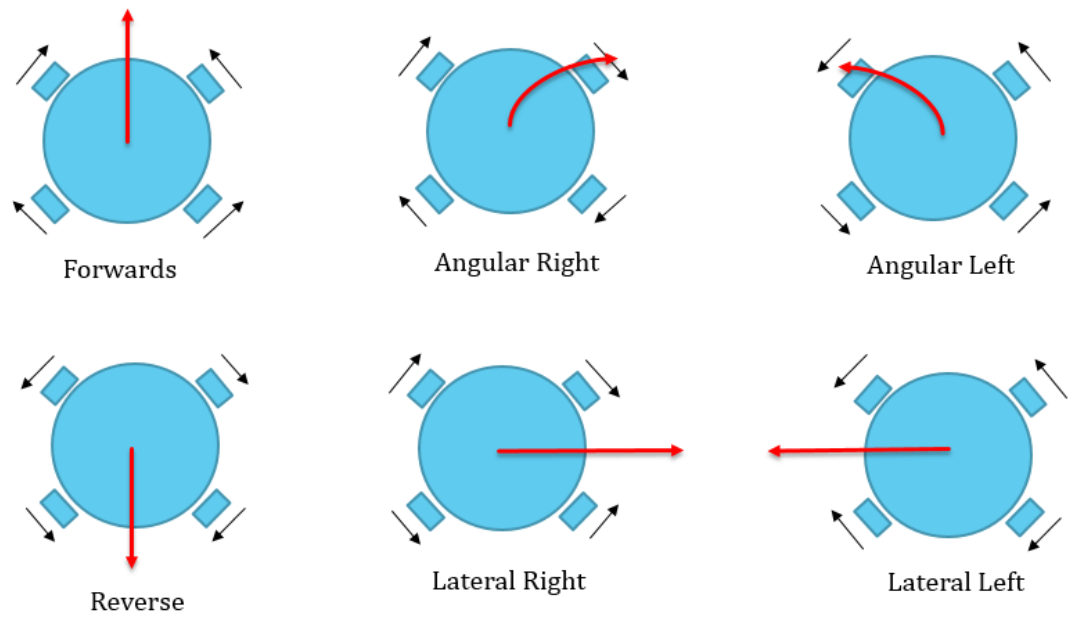


Figure 27. Rotation of wheels to achieve specific trajectories

5.1.1 Motor Control Adjustments

Although the motors were all sent equivalent values of voltage for each direction of movement depicted in Figure 27, the characteristics of each motor varied slightly to one another. Due to these differences, when driving forward, reverse, and laterally left or right, the robot would veer off to a tangent. This caused a major complication when attempting to drive the robot to specific locations on the track. For instance, when using an encoder count to determine how far to turn the robot angularly left or right, depending on the orientation of the robot, this would cause the robot to turn either further or less than expected. As a result, this would cause the robot to travel in the wrong direction and become lost when progressing to the next state of the program. Another area where misalignment of the robot caused issues was when the robot attempted to deliver medicine units onto the elevated platforms. If the robot approached the platform on an angle this would cause the robot to stop either too close or too far from the platform, hence failing to appropriately deliver the medicine unit. Stopping too far would cause the unit to fall off the platform, or fall short from landing onto the platform, while stopping too close from the platform would cause the robot to knock the unit off, or overshoot the placement of the unit.

To prevent the robot veering off from its required direction of travel, the motors causing the problem were identified, and the voltages applied to them were adjusted as required.

5.2 LIDAR

The Hokuyo LIDAR was connected to the myRIO through the USB device port. A pre-defined VI in LabVIEW was then used to establish communication with the LIDAR. However, an initial complication which occurred was that the required VI for LabVIEW to communicate with the LIDAR could not be found. The reason for this was that each time the LIDAR was connected; a USB module would take control over the LIDAR connection such that LabVIEW was unable to access it. To solve this problem, the 'cdc_acm' module within the LIDAR was blacklisted to avoid it from loading each time the LIDAR was connected to the myRIO. This was achieved by accessing the myRIO through 'putty' and entering the following script:

```
> echo blacklist "cdc_acm">/etc/modprobe.d/cdc_acm/uvccvideo.conf  
>reboot
```

With this problem resolved, another problem occurred where the LIDAR was found to update very slowly. The solution to this problem was found with the help of Xan Smith who was part of the Flinders 2014 NI ARC. To resolve this issue, the buffer within one of the subVIs contained in one of the Hokuyo LIDAR VIs required to be changed from 99999999 bytes down to 1000 bytes.

5.2.1 LIDAR Data Extraction

As mentioned in Chapter 4, the Hokuyo LIDAR was selected as the robot's primary sensor, providing it with 240° of vision. Figure 28 shows how data from the LIDAR was extracted using a pre-defined VI specifically for the Hokuyo LIDAR, where the VI outputs the distances and corresponding angles as 1D arrays. A graph of the LIDAR scan could then be plotted by converting the corresponding distances and angles to their real and imaginary components.

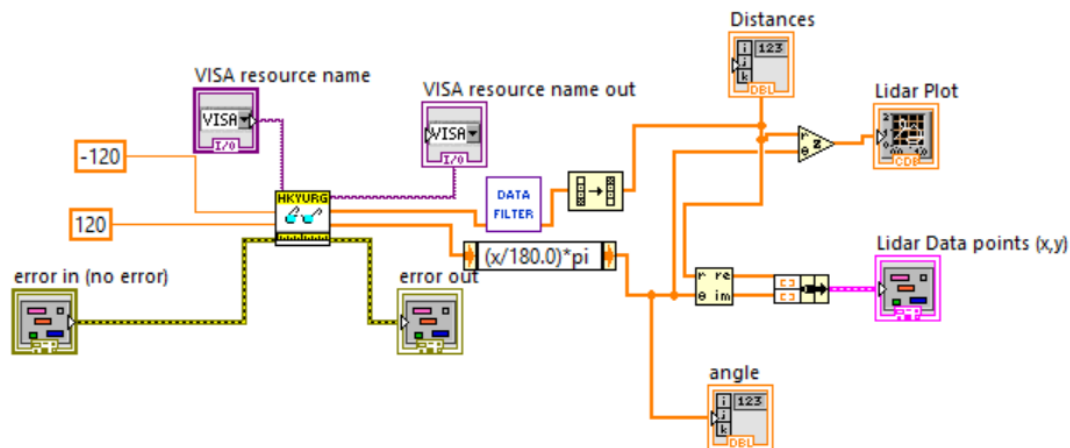


Figure 28. LIDAR data extraction

When testing the obstacle avoidance of the robot, it was witnessed the robot would sometimes not react as required. The cause was found due to the LIDAR scan producing readings of 0 (or close to 0), depending on the material of the object placed in front of it. To correct this, a LIDAR data filter was required to eliminate these faulty readings from the LIDAR. The simplest way to achieve this was to replace distances below a certain threshold with the distance from the previous point in the array.

The 0° point of the LIDAR is located directly at the centre front of the LIDAR and corresponds to scan number 342 (or index number 324 of the data array). Due to the LIDAR being mounted upside down, the outputted distance array was reversed to remain consistent with the original scan orientation of -120° to 120° from left to right respectively.

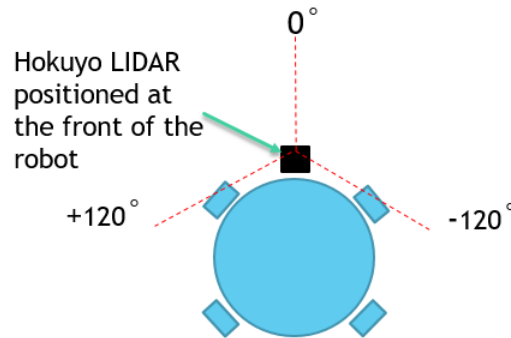


Figure 29. 0° point of LIDAR

With each scan increment equal to 0.36°, this results to 1° of vision to be equal to approximately 3 scans. Hence, to access a particular point or angle of from the LIDAR, equation (25) was used.

$$\text{scan number} = 342 \pm (3 \times \text{desired angle}) \quad (25)$$

Portions of the LIDAR scan can be isolated by extracting the appropriate data points within the data arrays, by supplying the starting index and the length of data required. This was accomplished via calculating the corresponding scan number (index) of the start and end angles required, then determining the data length between the two. As an example, if data from 90° of the front of the robot is desired, this can be obtained by accessing the data from -35° to +35°. See calculation below.

$$\text{Starting Point at } -35^\circ = 342 - (3 \times 45) = 207$$

$$\text{End Point at } +35^\circ = 342 + (3 \times 45) = 477$$

$$\text{data length} = \text{End Point} - \text{Starting Point} \quad (26)$$

$$\therefore \text{data length} = 477 - 207 = 270$$

5.3 Obstacle Avoidance

The methodology behind obstacle avoidance was that if the robot sensed an obstacle in front of it, it would then check to see if there was also an obstacle to its left. If an obstacle to its left was present, the robot would veer to the right, or else it would veer to the left.

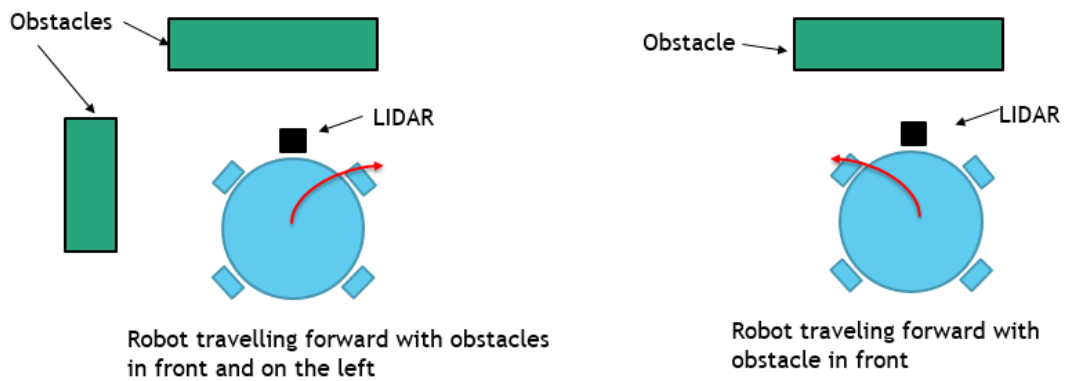


Figure 30. Obstacle avoidance methodology

An adequate detection range for the robot was required to account for the various sizes and positioning of the objects it may encounter. Thus, a range of 90° vision in front of the robot was selected for its front detection, and 30° to its left was used for its left detection. The LIDAR would continuously cycle through the distances at each increment of the specified scan range, and compare the distances read to a threshold value. If the distance read was less than the threshold value, the robot would respond as described in Figure 30.

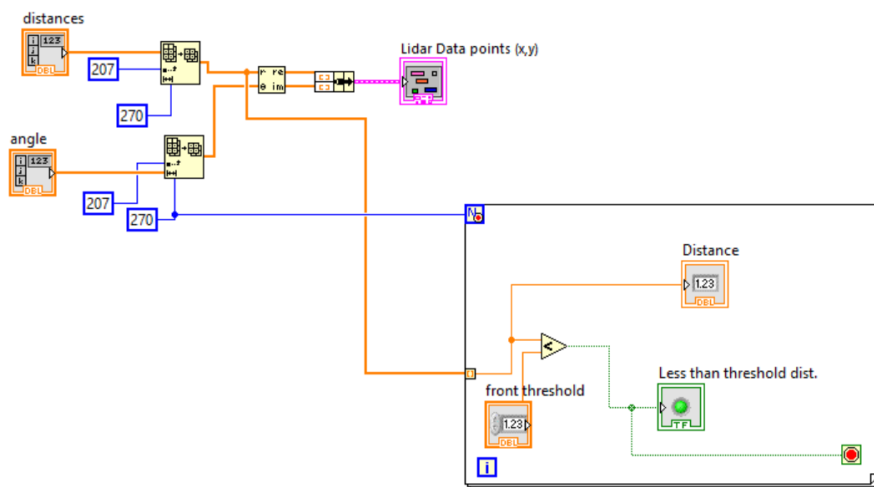


Figure 31. LabVIEW subVI for front detection

5.4 Wall Following

A wall following algorithm was used to help easily navigate the robot around certain areas of the track. As the locations of obstacles in the track would be unknown, wall following would also allow the robot to avoid them by remaining in close vicinity to the walls.

The basis for the wall following algorithm was to ensure the robot travelled parallel to the wall while remaining between a certain range from it. The methodology for the wall following algorithm is as follows; if the robot is greater than distance d_{upper} , the robot would move towards the wall, and if the distance was less than d_{lower} , the robot would move away from the wall.

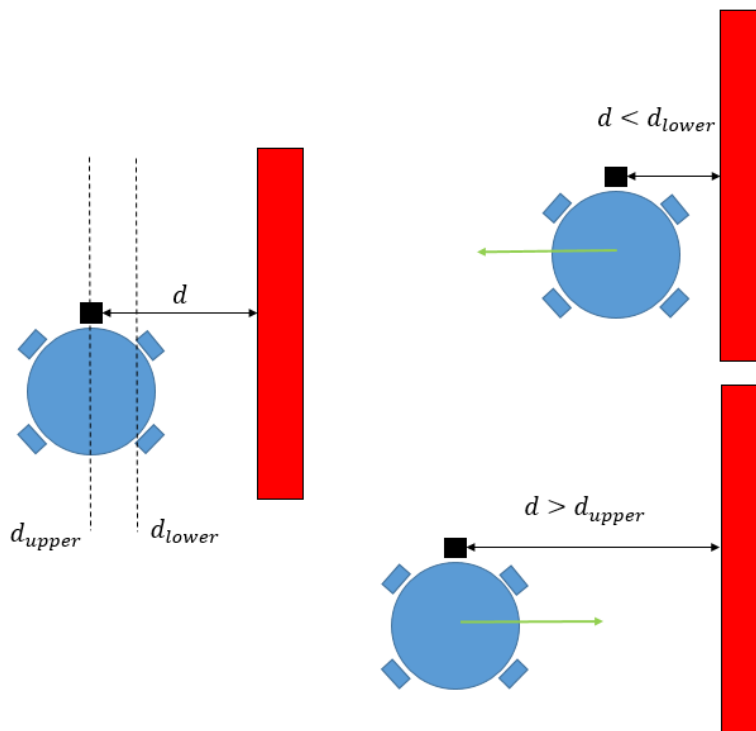


Figure 32. Wall following methodology

Another condition to the wall following algorithm was that if the LIDAR returned a distance greater than 1m, this would indicate the robot had reached an open area indicating the end of the wall. This condition was included as the wall following technique was only required for two walls of the track which had open ends. An opening distance of 1m could be used as one of the rules of the competition was that all obstacles would be placed at least 1m from any walls and other obstacles.

In the initial implementation stages, the robot was made to correct its distance via veering left or right as appropriate. Using a single point on the LIDAR proved to be effective, however, if the robot's heading was slightly angled towards the wall, the robot would continue to follow the wall at an angle.

This would occasionally cause problems where the robot would crash into the wall due to the robot believing it was further away from the wall than it actually was.

An initial attempt to amend this problem was by increasing the lower and upper threshold boundaries. This was successful in preventing the robot from hitting the wall, however, it did not eliminate the issue of the robot following the wall at an angle. As a consequence, when the robot reached the end of the wall, it would become lost when entering the Main Area of the track.

The angular movement was then replaced with lateral movement to correct the robot's distance from the wall. Lateral movement proved to be more favourable due to a number of reasons. Firstly, in the competition, the robot can easily navigate its way to the wall from the starting position. Secondly, lateral movement would also allow the robot to traverse through the course faster, as lateral movement could be utilised where the robot would otherwise be required to turn by 90°. Thirdly, whilst the robot is wall following, it could also correct itself (angularly) to ensure it remained parallel to the wall, hence minimising the risk of the robot becoming lost when entering various regions of the track (this method is later discussed in Chapter 5.5). The implementation of the wall following algorithm is identified in Figure 33.

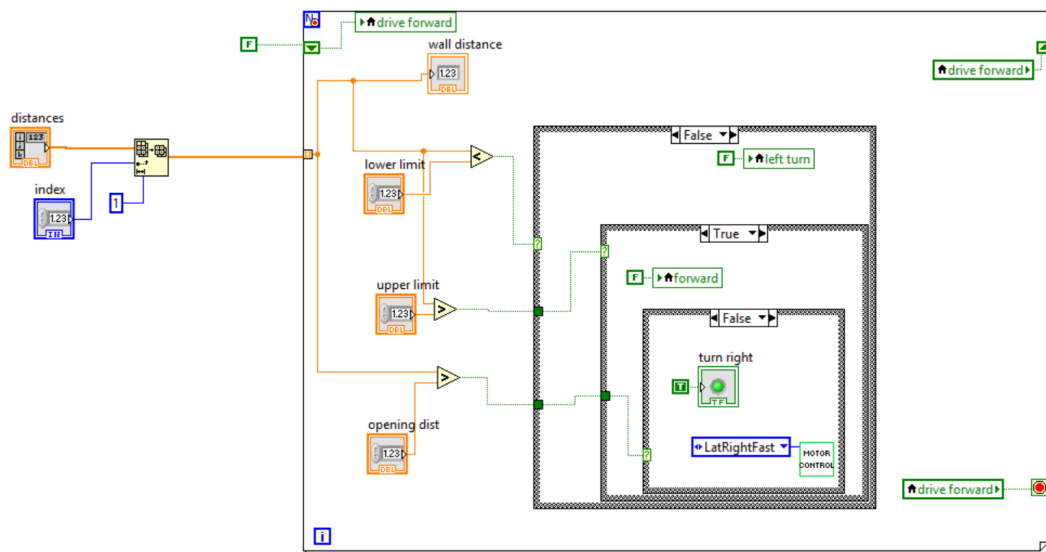


Figure 33. LabVIEW wall following subVI

5.5 Wall Alignment

One of the main challenges encountered was ensuring the robot would drive in its intended direction, to avoid it becoming lost. Although correcting the voltages supplied to the motors aided in keeping the robot from straying too far from its intended direction of travel, additional measures were still required due to wheel slippage.

Ensuring the heading of the robot remained as desired was a key factor in ensuring proper navigation for the robot. As an example, the wall following algorithm discussed in Chapter 5.4 required a correctional measure to ensure the robot remained parallel to the wall. In addition, one of the main challenges in the competition was ensuring the robot was able to enter and exit the wards without hitting, or crashing into the Perspex walls which divided them. Due to the size of the robot, there was not a great amount of leeway for error for the robot inside the wards. This meant the robot must position itself as closely to the centre of each ward. In addition, the heading of the robot must be as accurate as possible such that the robot could enter the wards head on.

A wall alignment subVI was thus written to correct the heading of the robot, by aligning the robot to walls when possible. To implement the wall alignment, a small section of the LIDAR data which would correspond to a wall was utilised. The first and last data points of that section would then be used to calculate the slope of the robot in relation to the wall via equation (27).

$$\text{Slope} = \frac{y_2 - y_1}{x_2 - x_1} \quad (27)$$

If a slope of 0 is calculated, this indicates the robot is parallel to the wall and the robot can continue driving straight. If the resultant slope is negative, then the robot would correct itself by veering right, and if the resulting slope was positive, the robot would veer to the left.

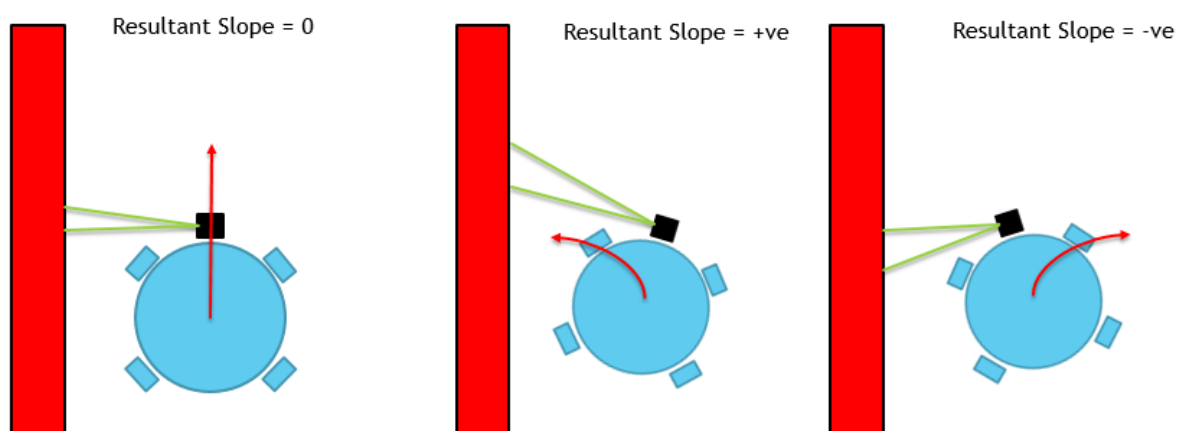


Figure 34. Wall alignment methodology

The code implemented for the wall alignment algorithm is depicted in Figure 35. The x, y co-ordinates required for the calculation were calculated by converting the angles and distances to their real and imaginary counterparts. The resultant slope could then be monitored through a plot.

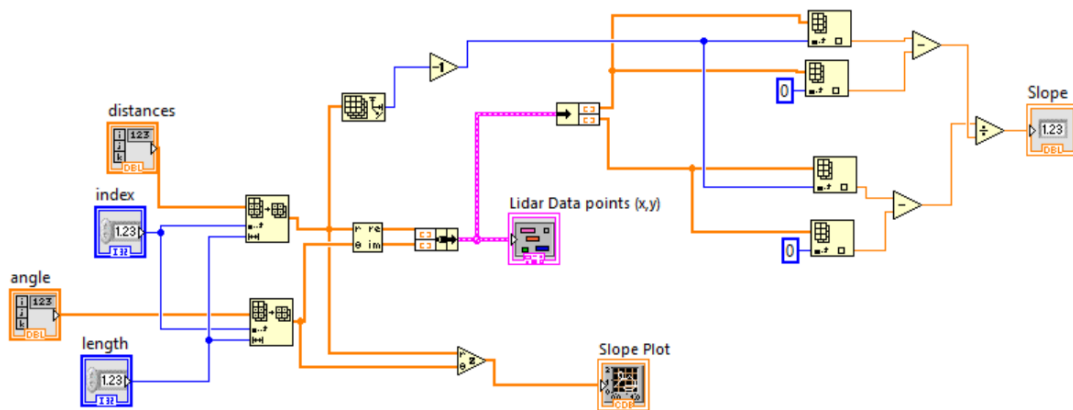


Figure 35. LabVIEW wall alignment subVI

Due to hardware limitations where the distances returned from the LIDAR would continuously fluctuate, the ideal slope of 0 could not be practically achieved. Hence, a tolerance range was required which would correspond to a slope of 0. This range was determined via trial and error, and by examining the slope values calculated whilst the robot remained stationary and aligned next to a wall. If the range was not large enough, the robot would continuously try to align itself to the wall, resulting with the robot constantly turning left and right, or it would overcompensate when it was already correctly aligned, causing it to then be misaligned. Consequently, if the range was too large, the robot would falsely believe it was correctly aligned.

This program proved to be successful when utilised for aligning the robot to left or right side walls, however was not successful when using walls in front of it. When examining the results for the front wall alignment, the slope values being calculated had ridiculous fluctuations and values between 200 to -100 although the robot was positioned perpendicular to the wall.

A useful tool in LabVIEW is the 'probe' tool which allows variables within a VI to be monitored. With the use of this tool and by examining the slope plot, the reason for these unreasonable values being calculated was found. When monitoring the values of y_2, y_1, x_2 and x_1 , the values of x_1 and x_2 were noticed to be similar to each other, whilst there was a large difference between y_1 and y_2 . When cross checking these results with the graph, it became apparent that due to how the LIDAR scans were orientated on the plot, the calculation for the front slope required to be switched to $\frac{x_2 - x_1}{y_2 - y_1}$.

5.6 Encoders

The encoders embedded within the motor controllers were utilised for cases when the robot was required to move a certain distance, or turn by a specific amount. For example, when using the wall following algorithm, when the robot reached the end of the wall, it required a method to clear the wall before attempting to enter the Main Area.

The wheel circumference of the robot is calculated using $c = \pi d$, where d is the diameter of the wheel. This is equal to the distance travelled by the wheel after one revolution. From this, the number of encoder counts required to drive the robot a certain distance can be calculated using equation (28).

$$\text{Encoder counts} = \frac{\text{distance to travel} \times \text{counts per revolution}}{\text{wheel circumference}} \quad (28)$$

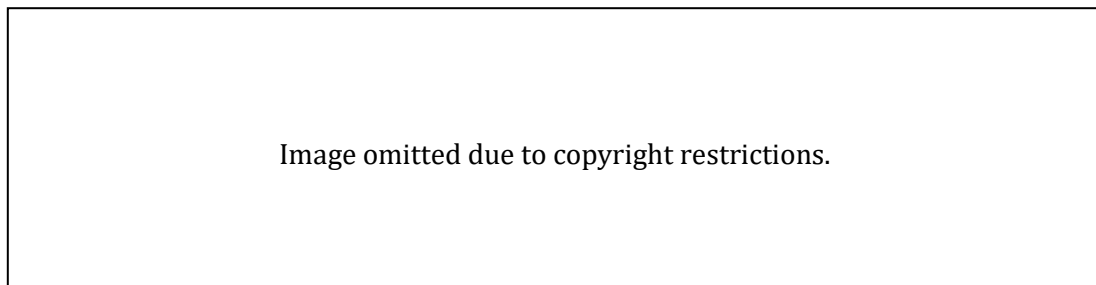


Figure 36. Distance travelled by one revolution of a wheel [16]

When this formula was adapted to drive the robot forward a set distance, the encoder counts calculated for this distance did not correspond to the distance travelled by the robot. The reason for this was due to the kinematics of the robot's wheel placement, where, the distance travelled by the robot is not equivalent to the linear distance travelled by the wheels.

To work around this issue, a general relationship between the encoder counts and distance travelled by the robot (in the forwards direction) was determined. A set of encoder counts were tested on one encoder, and the distance travelled between the starting and end positions of the corresponding wheel was measured. From this experiment, it was found that the robot travelled forward approximately 1cm per 100 encoder counts. As it was not crucial for the robot to accurately travel the defined distance, this method was still effective for ensuring the robot cleared the open walls of the track.

Although equation (25) could not be used to drive the robot forwards a specific distance, it however, could be applied when rotating the robot on the spot. This is likely due to the angular rotation of the robot closely relating to the linear distance travelled by the wheels. For the robot to have rotated 360° , it must turn a distance of πD , where D

is the diameter of the robot footprint. Thus, to rotate the robot 90°, the robot must travel a distance of $\frac{\pi D}{4}$. This could be applied to any one of the encoders as they are all required to travel the same distance to turn a certain angle. The number calculated was used as a basis, and the count was slightly adjusted through trial and error.

The code implemented to set the distance the robot must travel is seen in Figure 37. To ensure the program worked as required, it was important to clear the encoder count beforehand. If the current encoder count had already surpassed the specified count, the program would be caught in the loop, but if the count had not been surpassed, the offset of the current count would cause the specified distance to be cut short.

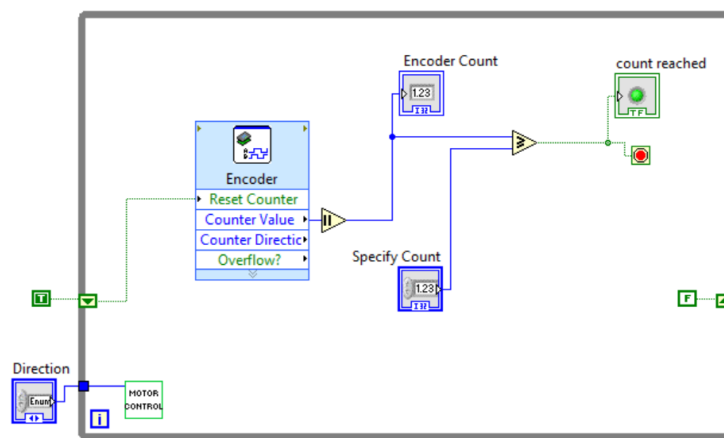


Figure 37. LabVIEW subVI to control the distance travelled by the robot

5.7 Placement of Medicine Units

A simple and repetitive program for the placement of the medicine units was opted for to reduce the amount of programming required. This goal was easily achieved with the pick and place mechanism created by Joel.

The servo motors and stepper motors could be controlled by connecting them to the PWM lines of the myRIO, then sending them a duty cycle and frequency. The duty cycle and frequency combinations to open and close the gripper by the appropriate amount, along with the linear movement of the robotic arm was determined by Melissa. Using these values, the movement of the robotic arm to place the medicine units were able to be achieved sequentially by using a 'flat sequence'. The routine is as follows: close gripper to obtain block, lower the arm down to the elevated carrier, open gripper to place block, bring arm back up.

Although the stepper motor was selected to allow for precise incremental control of the top rotating plate, it was found that using a timer to control how far the plate rotated was accurate enough. A timer value was found such that each rotation positioned the

block directly in front of the robotic arm. This then allowed the movement of the rotating plate to be easily integrated into the ‘flat sequence’ depicted in Figure 38. The plate would rotate whilst the current block is being placed, so that the next block would be readily positioned in front of the grippers for the next delivery.

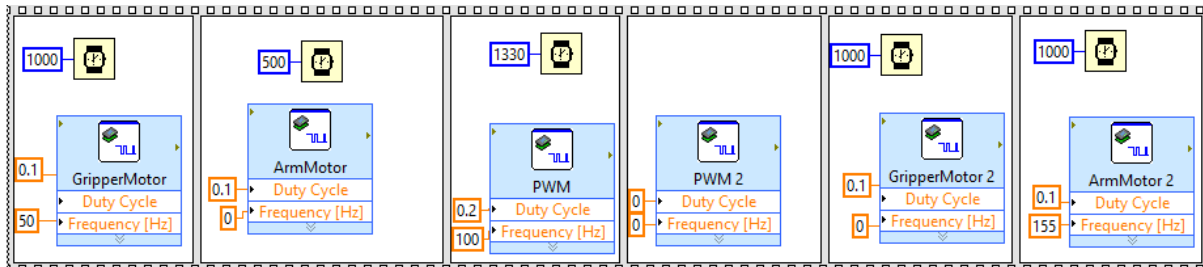


Figure 38. LabVIEW flat sequence for medicine unit placement

5.8 Split and Merge

From Chapter 3.3 it was proposed that Split and Merge line extraction was a good option for feature-extraction due to its fast response time and accuracy. As the course would consist entirely of straight lines (apart from the dynamic obstacle), this technique would be useful for extracting the surrounding walls and static obstacles.

The algorithm can be separated into two separate entities the ‘Split’ and the ‘Merge’. The Split function works by recursively splitting the LIDAR data until no more splits can be made. A baseline equation is found by using the first and last elements of the LIDAR data. The perpendicular distance of each point in the array to the baseline are then calculated, and the data is split into two separate arrays at the index of the point with the greatest distance from the line if the distance is greater than the threshold value. This process then occurs recursively for each split, until no more splits can be made. That is, the furthest distance from the baseline is less than the threshold [10].

One of the difficulties encountered when implementing the split function was calculating the baseline equation. The distance from a point (x_p, y_p) to a line $Ax + By + C = 0$ is given equation (29) [17].

$$d = \frac{|Am + Bm + C|}{\sqrt{A^2 + B^2}} \quad (29)$$

Where the line $Ax + By + C = 0$, has a slope of $-\frac{A}{B}$. The slope $-\frac{A}{B}$ could easily be calculated by calculating the slope between the first and last elements of the LIDAR data. However, the complication occurred when trying to solve C . Re-arranging the equation of a line, C could be determined using $C = -Ax - By$. When this was implemented, the Split VI caused LabVIEW to continuously disconnect to the myRIO.

The problem was known to be caused due to an error in how C was being calculated, but the location of the calculation error could not be identified.

As a result, another equation was found which allowed the perpendicular distance of a point (x_p, y_p) to be calculated from a line defined by two points (x_1, y_1) and (x_2, y_2) , which is described in equation (30) [18].

$$d = \frac{|(y_2 - y_1)x_0 - (x_2 - x_1)y_0 + x_2y_1 - y_2x_1|}{\sqrt{(y_2 - y_1)^2 + (x_2 - x_1)^2}} \quad (30)$$

This equation is equivalent to equation (26), however eliminated the need to calculate C as the equation was in terms of known points which would be extracted from the LIDAR. The code to implement the Split function is depicted in Figure 39.

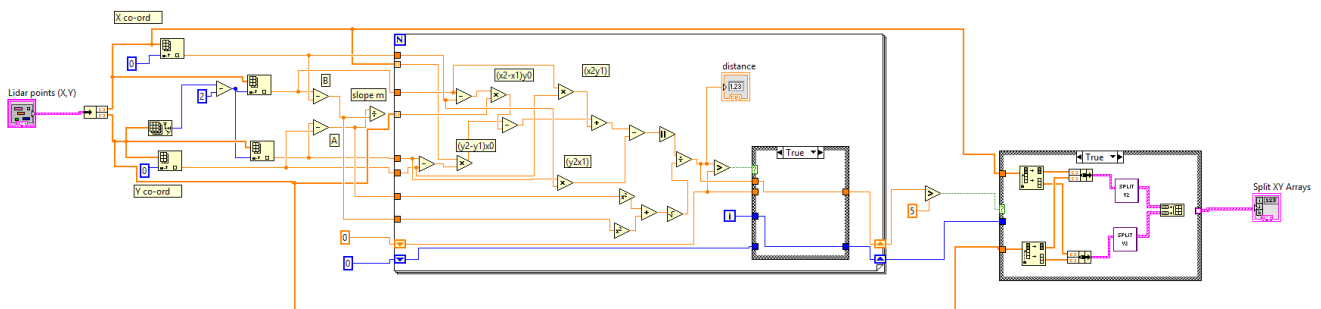


Figure 39. LabVIEW subVI for Split function

The 'Merge' section of the algorithm traverses through the set of split arrays (where each array represents a 'line' defined by a series of points), and determines which arrays can be joined together. This is achieved by first finding the baseline equation between the first two lines. The perpendicular distance to the baseline from the point where the two lines join is then compared to a threshold value. If the distance is less than the threshold value, the two lines are merged, else they remain as two separate lines. An example of how Split and Merge works is depicted in Figure 40.

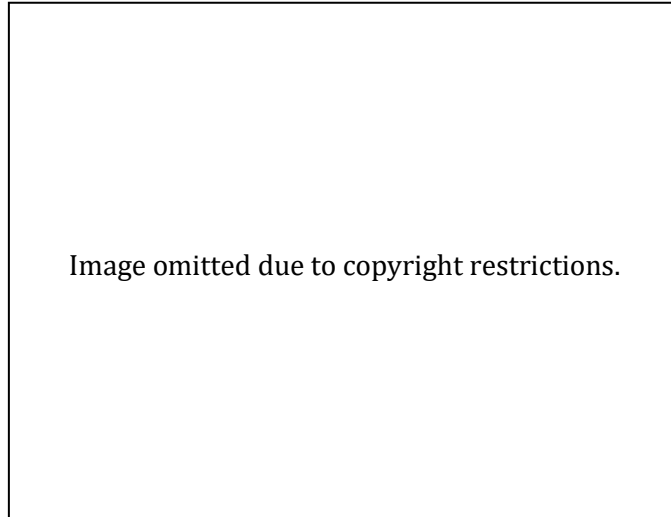


Figure 40. Theory of Split and Merge line segmentation [10]

The code implemented for the Merge function is shown in Figure 41 where Equation (27) was also used to calculate the perpendicular distance of a point to the baseline.

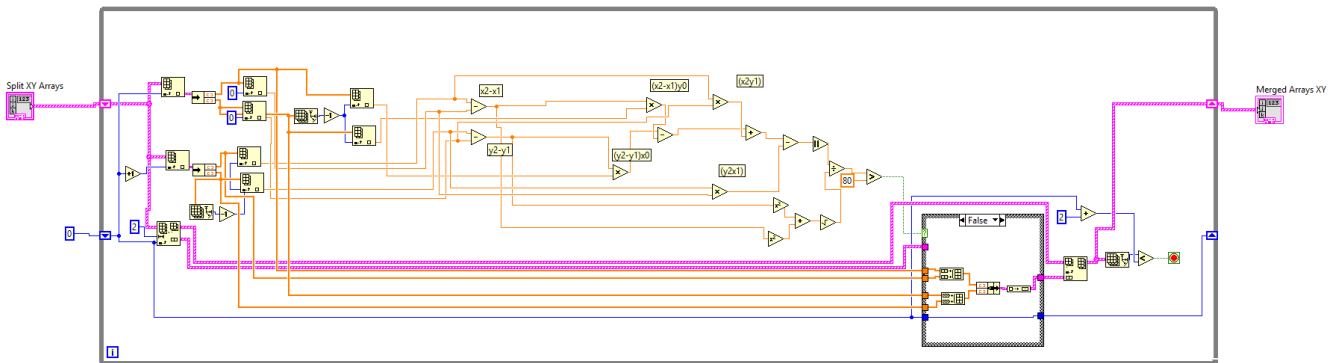


Figure 41. LabVIEW subVI for Merge function

Figure 42 is the plot of the original LIDAR scan, and Figure 43 is the resulting extracted lines from the Split and Merge. As the threshold selected for the Split function is decreased, the data is split into more sections. This is portrayed in Figure 44 where a threshold of 10 was used. The Merge thresholds were adjusted through testing where the threshold required to be increased when lines which were expected to be merged were not merged.

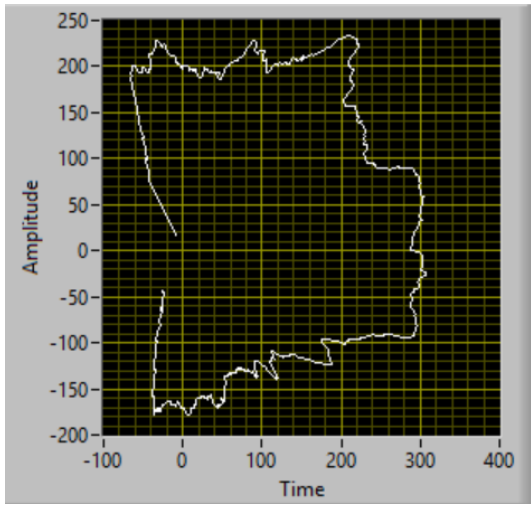


Figure 42. Original LIDAR data plot

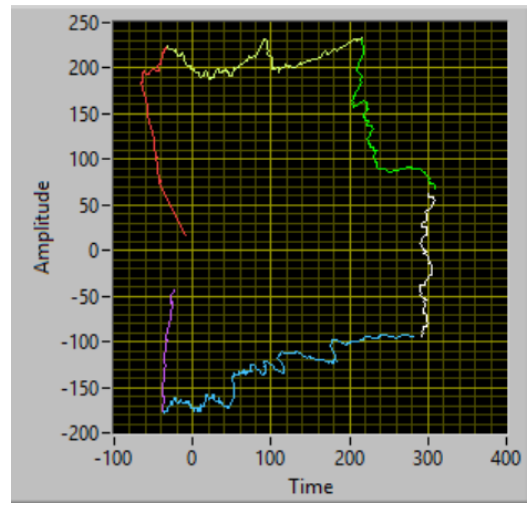


Figure 43. LIDAR data after Split and Merge

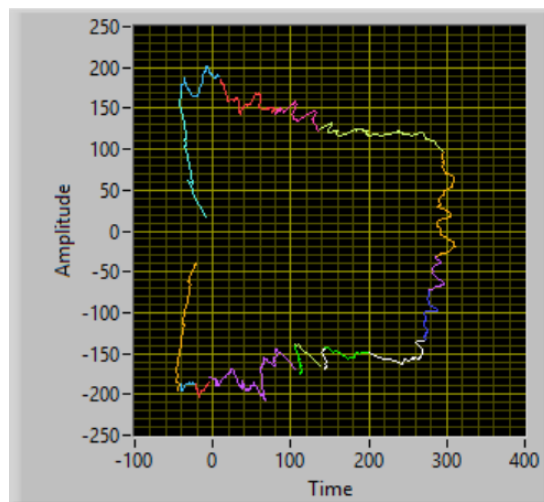


Figure 44. Split with threshold value of 10

5.9 Corner Detection

By applying the Split and Merge functions, features in the course such as corners could be extracted. A corner could be identified if the slope of two lines were perpendicular to one another, such that the product of the two slopes is equal to 1.

The corner detection algorithm was implemented by iterating through the merged line arrays, and calculating the product between the slope of the two lines. Due to hardware limitations and inaccuracies, the chances of the product between two perpendicular lines equalling exactly 1 is not likely. Therefore, a range was determined to compensate for this. This program was initially tested by utilising a section of the LIDAR and positioning the LIDAR in front of corners and observing if the 'Corner Detected' indicator would turn on. A second method of testing this was to graph the locations of the detected corners and see if they matched with the corners in the LIDAR plot. Results from both methods indicated the algorithm successfully detected corners.

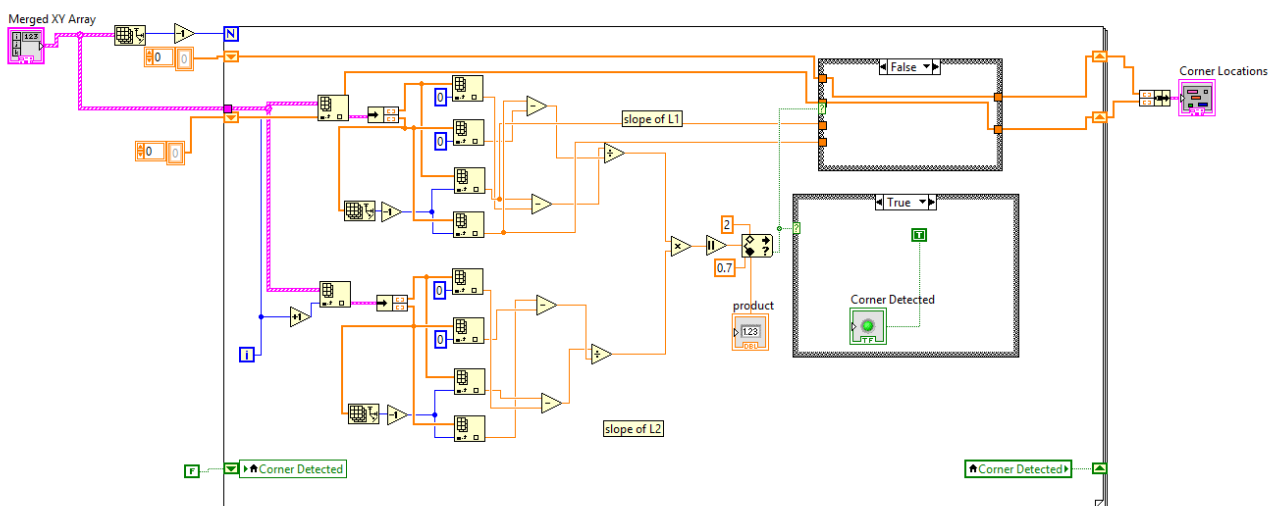


Figure 45. LabVIEW subVI for corner detection

5.10 Wireless Communication

To obtain bonus points in the competition, wireless communication to the server was required. A string was to be sent the server after the robot had completed its task of delivering all of the medicine units to The Wards and Operations Theatre. A program to achieve this was written by Melissa and was successfully tested on the server demo application. The basis of the program is to open a connection to the server, send the length of the string message, followed by the string message, then close the connection.

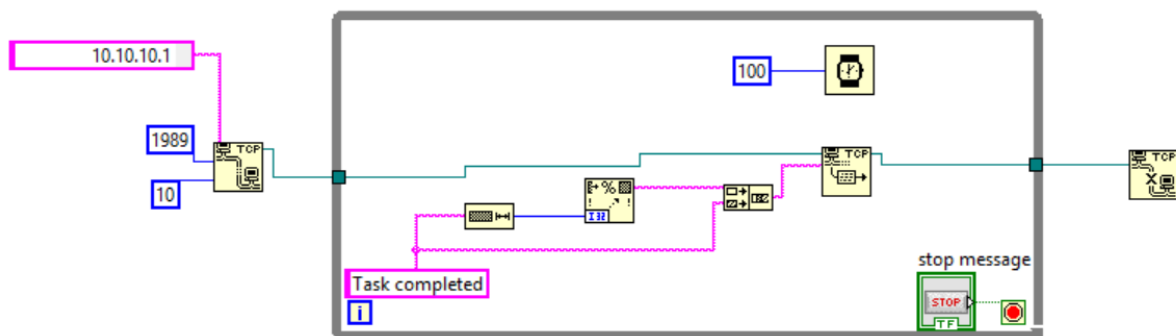


Figure 46. LabVIEW subVI for wireless communication

5.11 Final Program Overview

For the robot to succeed in the competition, the robot must exhibit a form of navigation. This was achieved by implementing a state machine system. The track was broken down into discrete states to navigate the robot around the course. The robot would know which state to transition to depending on the requirements met in the current state.

As indicated in Table 1, the tasks for Qualifiers, and Round of 16 are identical, where the robot is required to deliver a total of two medicine units; the first to one of the wards, and the second to the Operations Theatre. This allowed for the same program to be utilised for both rounds. The path taken by the robot is identified in Figure 47 below.

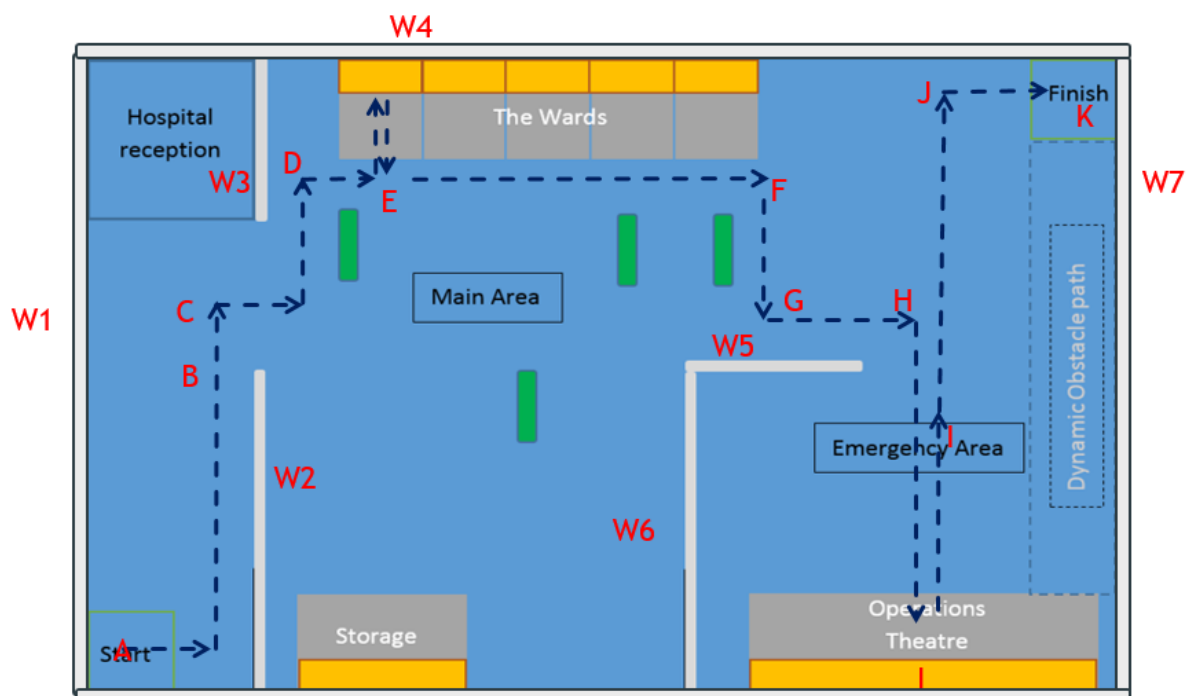


Figure 47. Navigation using a state machine

To begin, the robot uses the wall following technique to navigate from A to B. When the opening of the wall was sensed, the robot then travels a certain distance to ensure clearance of the first wall identified by C. At C, the robot ensures it is aligned to W1, before navigating to D.

To avoid any obstacles which may be in the main area, the robot positions itself next to W3 shown by D. The robot accomplishes this by moving from C such that it is a certain distance away from W1, it then moves forward until it reaches a certain range from the wall in front of it identified by W4. The robot utilises W3 to align itself and also as a landmark to determine how far it has to travel to centre itself in front of the first ward.

At D, the robot aligns itself to W3, then moves laterally to E. The robot knows it is at E by detecting its distance from W3. The robot was required to move relatively slowly when positioning itself in front of the wards as to avoid the LIDAR missing the defined range for the robot to stop.

The robot enters the first ward and delivers the medicine unit and then reverses from the ward back to E. The robot recognises when to stop reversing from the ward by sensing its distance from the elevated carrier within the ward.

At E, the robot again aligns itself to L3, then turns 90° right. By first aligning itself to L3, this ensures the robot turns the required 90°. After turning, the robot reaches location F by determining its distance from W7, then moves to G and H by wall following. While following W5 the robot also checks it remains aligned to the wall. When the opening of L5 has been reached, the robot uses the same technique as it did to clear W2, to reach H. From H the robot turns right 90°. By ensuring the robot remained aligned to L5, this guaranteed the robot would complete the 90° turn with the correct heading.

After turning, the robot places the second medicine unit onto the elevated carrier in the Operations Theatre. The robot again recognises it is in front of the carrier by detecting its distance from it. After placing the second medicine unit, the robot turns 180°, and drives a certain distance to I. At I, the robot ensures it is aligned to W6 before moving to J. Due to the glass Perspex encasing the operations theatre, it was essential the robot cleared the Perspex before attempting to align itself to W6, else the Perspex would cause errors in the LIDAR data. After aligning itself, the robot also ensures it is a certain distance from W6, to remain a safe distance from the Dynamic Obstacle Path.

The robot then positions itself at J and aligns itself to the wall in front of it. The robot identifies it is at J by determining its distance from W4, then moves laterally to the finish point K.

For the remaining rounds of the competition, the main differences were the number of wards which were required to be delivered to (where the maximum was three), and the quantity of medicine units to be delivered to the Operations Theatre. To accommodate for the additional wards, the same logic as entering the first ward and delivering the block was implemented, where W3 was used as a landmark to centre and align itself in front of each ward. This was integrated by using one 'wall alignment' state within the state machine. After aligning itself, the robot would determine whether another ward required a delivery or not, depending on the conditions of the case statements within the state. An excerpt of this is depicted in Figure 48. The robot would then navigate its way to the Operations Theatre using the same method discussed for the Qualifiers and Round of 16 program.

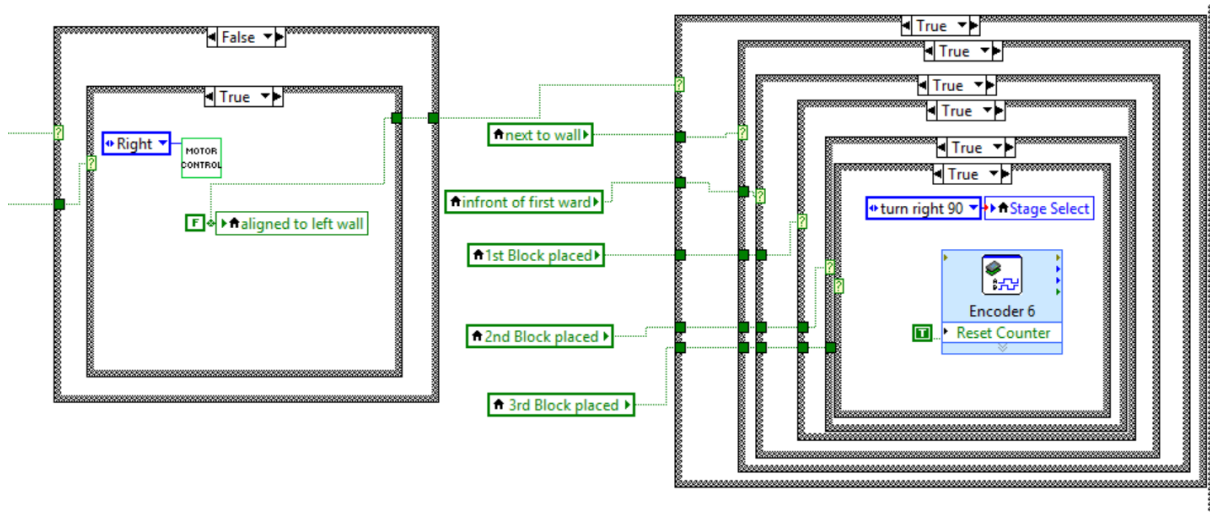


Figure 48. Excerpt of 'wall alignment' state

The delivery system of the robot only allowed for one delivery at a time. When more than one block was required to be delivered to the Operations Theatre, the robot was required to move slightly to the left or right to place the next unit. To avoid the robot entering the Dynamic Obstacle Path, the robot was made to move in the direction towards W7 to place the subsequent units. The robot would then navigate itself to the finish position K via the same process discussed for the Qualifiers and Round of 16.

The robot was never made to deliver to Storage as it was not a requirement and contributed to the least amount of points. In addition, there was a high likelihood of encountering an obstacle along the way to Storage, such that it was not worthwhile making a delivery there.

Although a program for wireless communication was written, it was not implemented in the final program for the competition. The main reason being due to not being able to test the wireless communication with the server prior to the competition. At the time the wireless communication was going to be tested, there were issues with the server.

Chapter 6

Results

6.1 Pre-competition testing

All members of the team were present at the live final competition which was held at the University of Technology Sydney (UTS). The day before the competition, every team was allocated a period of time to test their robots on the two identical tracks used for the competition. This allowed teams to tweak their programs according to the robot's performance on the competition track. As the replica track in our lab at Flinders University closely resembled the competition track, only minor changes were required. These changes in particular were the stopping ranges for entering the wards and the stopping distances in front of the elevated carriers. The only other major change required was removing the alignment state used before parking the robot in the finish position. Although this alignment state proved to improve the alignment of the robot when tested on our replica track, this was not the case when tested on the competition track. Rather than correcting the alignment of the robot, the robot became more misaligned causing the robot to enter the Dynamic Obstacle Path. However, it was found that alignment at position I in Figure 47 was sufficient enough to keep the robot in the correct orientation when moving to the finish area.

6.1 Competition Results

On the first run of the group knockout stage, the robot did not complete the course as it stopped too far away from W4 (Figure 47) and caused the robot to miss W3. From here, the robot did not have the correct landmark to navigate to the first ward. This error was quickly corrected for the second run, where the robot successfully completed the course without any collisions. However, the robot was not as successful in the final run, as the medicine unit was not correctly placed onto the carrier of the ward, and fell to the ground. Nevertheless, the team still finished in 1st place in their heat for the Qualifiers round; scoring the highest points with the second run. Unfortunately, the team was immediately knocked out in the Round of 16, as this time, the robot stopped too close to W4, such that when it moved towards the first ward, the robot hit the Perspex and caused the LIDAR to clip the Perspex wall. The robot remained stuck and also lost its way as it began to rotate while it was clipped. As a result, the team had to call time, and were eliminated from the competition. This could have been avoided if the distance range from W4 was increased. However, since this did not seem to be an issue for the previous runs, this result was not expected.

Chapter 7

Future Work and Conclusion

7.1 Future Work

After the completion of the competition, there are a number of improvements which can be made to improve the performance, reliability and efficiency of the robot.

Firstly, a motor control algorithm could be implemented to control the movements of the robot. In the competition, the robot was controlled by manually setting the voltages sent to the motors to control its speed and direction of motion. Although this worked fine, a more advanced way to control the motors could be implemented to better utilise the manoeuvrability and motion of the robot. In LabVIEW there is a steering frame VI which is able to control the motion of the robot. By providing the lateral, forward and angular velocities, the VI is able to determine the required output voltages of each motor to achieve the overall steering frame velocity desired. If this VI was utilised the robot could be made to drive more accurately and with better motion. Additionally, a controller such as a proportional integral derivative (PID) controller could be integrated to smoothly control the acceleration and deceleration of the robot, such that the movement of the robot is not so 'jerky'.

The state machine navigation algorithm allowed the robot to efficiently navigate the robot through the course and complete its required tasks. However, this was highly dependent on the robot successfully reaching each pre-set/expected location to progress correctly to the next state. If this was not the case, the robot would immediately become lost and had no method to recover its location. For this reason, a localisation algorithm such as the EKF could be implemented. This would allow the robot to know its location on the course and recover its location.

At times, the wall alignment technique would be a hit or miss depending on when the function was applied. To account for this, the sensor fusion technique discussed in Chapter 3.1 could be used where a gyroscope could be used in conjunction with the EKF to provide more accurate information on the robots heading along with providing more precise localisation.

Although the Split and Merge function along with corner detection was implemented, they were not utilised. One of the main reasons as to why, was because the current implementation of the state machine worked well at the time, where relying solely on distance measurements obtained from the LIDAR data proved to be effective. Rather than over complicating the program, it was decided it was more important to implement a complete program which will be successful for the competition. However, in future, these functions could be integrated with an EKF to achieve a more accurate localisation process. In addition, using Split and Merge to identify walls of the track

could eliminate the robot from becoming lost, where sometimes using distance data alone is not reliable enough.

Lastly, the use of optical mouse sensors could be further looked into for localisation. As seen in Chapter 3.2, optical mouse sensors can provide more accurate odometry measurements in comparison to encoder odometry when used for localisation.

7.2 Conclusion

Overall, the team worked well together and were able to complete all the milestones on time, and compete in the live finals. Although the team did not progress very far into the competition, it was pleasing to know the robot was able to compete against the top universities from Australia and New Zealand. The team received many compliments on the robot's object handling mechanism, where the team's mechanism resulted in being one of the most reliable.

From the competition, it was learnt that the simpler robots are usually more successful. Although having the robot move with greater speed is more impressive it often caused the robot to drop units along the way and severely crash into obstacles. It was found that sometimes moving slowly and accurately will bring out better results.

In addition, the results and experience from the competition showed that the programming of the robot did not require the most complex algorithms to achieve the tasks of the competition. A simpler program allowed quick adjustments to be made on the spot which was an important factor in succeeding in the competition. Nevertheless, if implemented correctly, more advance algorithms would prove to be more efficient and robust for localising the robot, and there are still many improvements for localisation which could be made to the team's robot in the future.

References

- [1] National Instruments 2016, *NI Autonomous Robotics Competition 2016*, viewed 1st October 2016, <<http://australia.ni.com/ni-arc>>
- [2] National Instruments 2016, *NI ARC 2016 Competition Task and Rules*, viewed, 1st October 2016, <<http://australia.ni.com/sites/default/files/NI%20ARC%202016%20Task%20and%20Rules%20Documentation.pdf>>
- [3] National Instruments 2013, *NI myRIO-1900: User Guide and Specifications*, viewed 1st October 2015, <<http://www.ni.com/pdf/manuals/376047a.pdf>>
- [4] National Instruments 2013, *NI myRIO*, viewed 1st October 2015, <<http://sine.ni.com/nips/cds/view/p/lang/en/nid/211694>>
- [5] National Instruments 2016, *LabVIEW Environment Basics*, viewed 1st October 2015, <<http://www.ni.com/getting-started/labview-basics/environment>>
- [6] Negenborn, R 2003, 'Robot Localisation and Kalman Filters', Masters Thesis, Utrecht University, viewed 20th July 2016, <http://www.negenborn.net/kal_loc/thesis.pdf>
- [7] Sangale, V & Shendre, A 2013, 'Localization of a Mobile Autonomous Robot using Extended Kalman Filter', *Advances in Computing and Communications*, Maharashtra Institute of Technology Pune, India, pp. 274-7, viewed 17th June 2016, <<http://ieeexplore.ieee.org.ezproxy.flinders.edu.au/document/6686387/>>
- [8] Cho *et al.* 2011, 'A dead reckoning localization system for mobile robots using inertial sensors and wheel revolution encoding', *Journal of Mechanical Science and Technology*, Vol.25, Issue 11, pp. 2907-2917, viewed 18th June 2016, <<https://www.scribd.com/document/279383696/Full-Text>>
- [9] Sekimori, D & Miyazaki, F 2007, 'Precise dead-reckoning for mobile robots using multiple optical mouse sensors', *Informatics in Control, Automation and Robotics II*, pp. 145-151, viewed 17th July 2016, <https://people.ece.cornell.edu/land/courses/ece4760/FinalProjects/s2009/ncr6_wjw27/ncr6_wjw27/docs/dead_reckoning_with_mouse_sensors.pdf>
- [10] Nguyen *et al.* 2005, 'A comparison of line extraction algorithms using 2D laser rangefinder for indoor mobile robotics', International Conference, Edmonton, Canada 2005, viewed 17th June 2016, <<http://ieeexplore.ieee.org.ezproxy.flinders.edu.au/xpls/icp.jsp?arnumber=1545234>>
- [11] Hokuyo 2009, *Scanning range finder (SOKUIKI sensor)*, viewed 10th April 2016, <https://www.hokuyo-aut.jp/02sensor/07scanner/urg_04lx_ug01.html>

- [12] Hokuyo Automatic 2009, *Scanning Laser Range Finder URG-04LX-UG01 (Simple-URG) Specifications*, viewed 10th October 2016, <http://www.hokuyo-aut.jp/02sensor/07scanner/download/pdf/URG-04LX_UG01_spec_en.pdf>
- [13] Pololu Robotics and Electronics 2016, *50:1 Metal Gearmotor 37Dx54L mm with 64 CPR Encoder (No End Cap)*, viewed 10th April 2016, <<https://www.pololu.com/product/1444>>
- [14] Dimension Engineering 2007, *Sabertooth 2x5 User's Guide*, viewed 10th October 2016, <www.dimensionengineering.com/datasheets/Sabertooth2x5.pdf >
- [15] RLX 2016, *Servo - Hitec HS-422 Standard Size (Sparkfun ROB-11884)*, viewed 10th October 2016, <<http://rlx.sk/en/servo-motor/3758-servo-hitec-hs-422-standard-size-sparkfun-rob-11884.html>>
- [16] ROBOTC 2012, *Using encoders to drive some distance*, viewed 10th October, <http://www.robotc.net/wikiarchive/Tutorials/Arduino_Projects/Mobile_Robots/VEX/Using_encoders_to_drive_some_distance>
- [17] Interactive Mathematics 2014, *Perpendicular Distance from a Point to a Line*, viewed 1st September 2016, <<http://www.intmath.com/plane-analytic-geometry/perpendicular-distance-point-line.php>>
- [18] WIKIPEDIA 2016, *Distance from a point to a line*, viewed 5th September 2016, <https://en.wikipedia.org/wiki/Distance_from_a_point_to_a_line>