SMART TRANSPORTATION

Ryan Safour

Bachelor of Engineering
Mechatronic Engineering Major

Department of Engineering
Macquarie University

November 6, 2017

Supervisor: David Inglis

ACKNOWLEDGMENTS

I would like to acknowledge National Instruments for their funding of the myRIO, LabVIEW software and extensive tutorials this will surely be an excellent tool to have in the workforce. I would like thank my Supervisors for all their assistance and guidance. Thank you to last years NIARC team have provided us with an excellent base for success in the competition. Also a final shout out to Team MQMechatronics it was a pleasure working with everyone and has been a memorable experience.

STATEMENT OF CANDIDATE

I, Ryan Safour, declare that this report, submitted as part of the requirement for the award of Bachelor of Engineering in the Department of Electronic Engineering, Macquarie University, is entirely my own work unless otherwise referenced or acknowledged. This document has not been submitted for qualification or assessment an any academic institution.

Student's Name: Ryan Safour

Student's Signature: R. Safour

Date: 06/11/2017

ABSTRACT

The National Instruments Autonomous Robotics Competition is a challenging experience where students are pushed to the limit to create an autonomous vehicle that performs various tasks. Using National Instruments state of the art technology and software, myRIO and LabVIEW. Challenges faced are localization, path planning and obstacle avoidance all while trying to complete the tasks at hand.

Our design utilizes techniques using a mecanum drive system, quadrature encoders and ultrasonic sensors in order to successfully navigate a changing environment accurately. The purpose of my thesis will be to be a guide and mentor the future NIARC teams to give them an insight of what to expect and problems they may face. Our research will also serve those looking to find a simple way to successfully navigate a changing environment.

Contents

List of Figures

Abbreviations

ECHO	Echo Location
ENC	Encoder
FPGA	Field Programmable Gate Array
NI	National Instruments
NIARC	National Instruments Autonomous Robotics Competition
PWM	Pulse Width Modulation
RIO	Re-configurable In/Out
TRIG	Trigger

Chapter 1

Introduction

The National Instruments Autonomous Robotics Competition (NIARC) is an Australia and New Zealand wide competition with a new team coming all the way from Singapore to compete. National Instruments (NI) are the sponsors of this event and provide every team with a myRIO microcontroller, LabVIEW software as well as training for LabVIEW. The myRIO device provided by National Instruments requires LabVIEW to function. It is a real time imbedded evaluation board which utilizes its on board FPGA and microprocessor. The FPGA created by NI is unlike any other as it can be easily programmed by anyone who has a familiar background of basic coding. This is explained in comprehensive training modules provided to students. There is an opportunity for teams to complete LabVIEW training ranging from the basics to more advanced topics. The

NIARC 2017 was the 7th consecutive year the competition has been run and this year was hosted at UTS in the Great Hall. Every year a new theme is developed relating to a real-world problem. This year the theme being smart transportation and the problem being navigation, path planning and obstacle avoidance. This year we were challenged to come up with the fastest way of navigating the track whilst trying to complete the set tasks. For a full list of the competitions rules and tasks please refer to the National Instruments website [13].

Chapter 2

Literature Review and Background

2.1 Smart Transportation - The Track

Figure 1 is a standard scenario with no obstacles. Follow the Purple line as you read starting in the green square in the corner and returning to the same location. Please refer to the competition task and rules document for more information [13].

The track represents a real city with the robot representing a car. The objective of the robot begins in the green square in the bottom left of the track. The robot must then travel through the residential zone, stopping in the yellow squares on the left-hand side as necessary according to the task adjudicator. Once stopped in the squares, a team member will load a maximum of 2 orange cubes onto the robot which will represent a passenger entering the vehicle. Once the passengers are loaded the robot must travel into the city grid as indicated by the purple line.

The robot will receive a data packet via WiFi informing the robot of various road blocks located between intersections in the city grid. The robot must then travel through the city grid whilst abiding by the school speed zone surrounding the school and avoiding the various road blocks. Next the robot will exit the city grid to the drop off zone, the vehicle must come to a stop within the yellow square. A member must then remove the orange cubes.

After that it will proceed to the highway where it will encounter a moving robot in which it must detect and avoid. The highway zone is a strict one way only if the robot moves backwards for any reason points will be deducted. At the end of the highway it will then proceed back into the city grid once again, avoiding road blocks and abiding by school zone speed limits. Finally the robot will enter the residential zone and finally come to a complete stop in the green square.

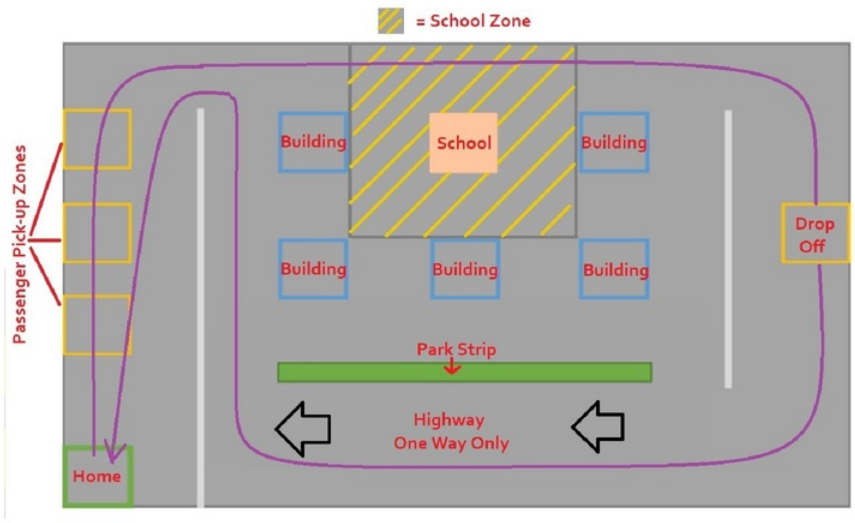

Figure 2.1: 2017 Smart transportation track, best case scenario path plan

2.2 Drive Systems

A drive system is a base for a robot. It is important to breakdown the problem at hand and figure out what could be the best drive system for our scenario. Different types of drive systems being considered are 3 wheeled swivel tank drive, swerve drive, holonomic drive and mecanum drive.

2.2.1 Swivel Drive (3 Wheel)

The swivel drive is a tank like drive with a front free turning wheel at the front. The benefits of this design are it only requires two motors (inexpensive), simple wheels (inexpensive), simple build, simple conceptually [14]. Consequences are it is imbalanced around corners cannot have a large load on the swivel, proportions could be difficult to achieve for corner movements can only hold low weight.

The motors are available for this design however we have four available motors so this is not an issue. If motors and wheels were not available this would be a definite choice.

2.2.2 The Akerman Drive

This Akerman drive design is two independently controlled motors with two wheels which require a motor to turn. This simple conceptually with independently steered motors, it is inexpensive with simple wheels and only two large motors and one small motors which allow for maximum pushing force. The design is difficult to build and program.

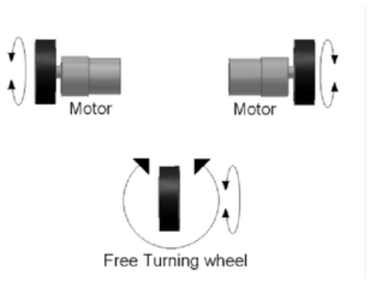

Figure 2.2: Swivel Drive with a front free turning wheel [1]

The small motor is not available for the steering system the steering system has moving parts which may prove difficult to maintain.

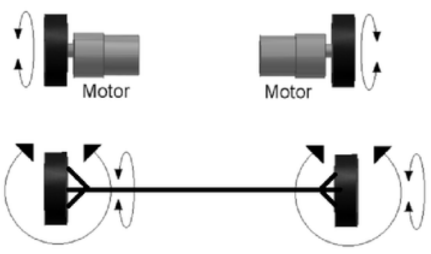

Figure 2.3: Akerman Drive front wheels turn freely [1]

2.2.3 Holonomic Drive

Holonomic drive uses 3 or 4 holonomic wheels. The drive is able to travel in any direction, it is simple to build. Consequences are the wheels are expensive, complex conceptually, maximum pushing force is lost when travelling in certain directions depending on the orientation. When wheels are positioned as shown in Figure 2.4 the kinematics are exactly the same as the mecanum wheel drive.

Overall the 3 wheel would be the cheaper option and just as effective as the 4 wheel drive with less pushing force. However the wheels are expensive and dont have enough benefits to overturn other options.

2.2.4 Mecanum Drive

Mecanum drive is wheels made up of rollers on a 45-degree axis. The drive is able to travel in any direction, it is simple to build. Consequences are the wheels are expensive, complex

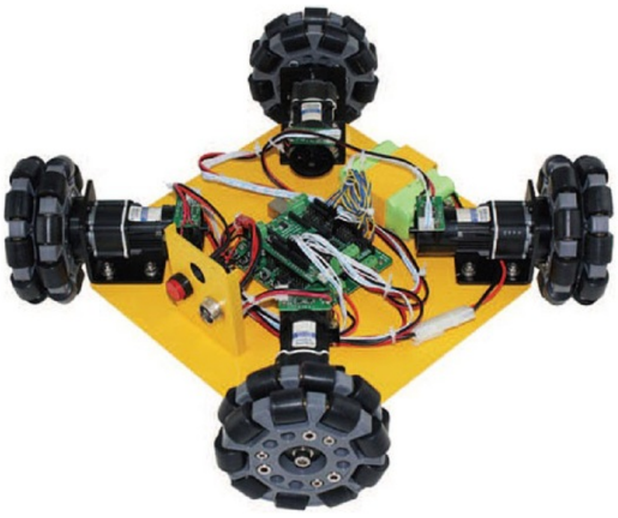

Figure 2.4: Holonomic Drive, wheels with straight rollers [2]

conceptually, maximum pushing force is lost when travelling in 45-degree direction and loss of traction on rough terrain.

Overall this design is favourable as we have access to the wheels and motors. A powerful battery will ensure the drive will still travel quickly. The design is slowest in the 45 degree direction though the vehicle will usually be travelling forwards, backwards or sideways. Also the course flooring is smooth so there shouldn't be loss of traction.

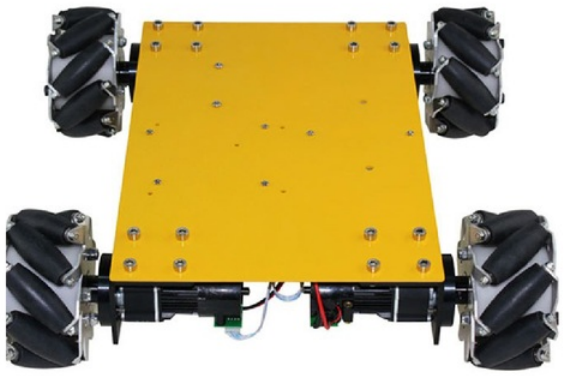

Figure 2.5: Mecanum Drive, wheels with straight rollers [3]

Considering the drive available is the mecanum drive it minimises the consequences of using a omnidirectional drive (Ability to move in any direction). The shape best fitting a mecanum drive is a square or rectangular chassis. In the article Simplistic control of Mecanum Drive the author has proposed an elegant solution for the problem as shown in Figure 2.6 [4]. Using these equations a voltage multiplier can be calculated for each motor to travel the desired speed and angle.

$$V_1 = V_d \sin(\theta_d + \frac{\pi}{4}) + V_\theta$$

$$V_2 = V_d \cos(\theta_d + \frac{\pi}{4}) - V_\theta$$

$$V_3 = V_d \cos(\theta_d + \frac{\pi}{4}) + V_\theta$$

$$V_4 = V_d \sin(\theta_d + \frac{\pi}{4}) - V_\theta$$

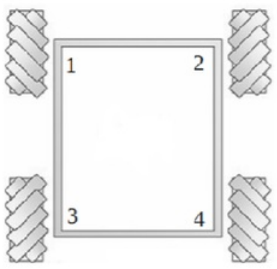

Figure 2.6: Calculating wheel speed Vd is the desired wheel speed, d is the desired angle, V is the desired speed for changing direction and Vn is the voltage multiplier being applied to the motor [4]

2.3 Path Planning Algorithms

2.3.1 Dijkstras Algorithm

Path planning is an important function if we wish to complete the course with a competitive time. It is important to think ahead and find the quickest path. There are algorithms out there in which use logic to travel though a changing environment. One of the simplest algorithms is Dijkstras algorithm [5]. As our map is simple enough to be represented by an array of switches this is the perfect algorithm.

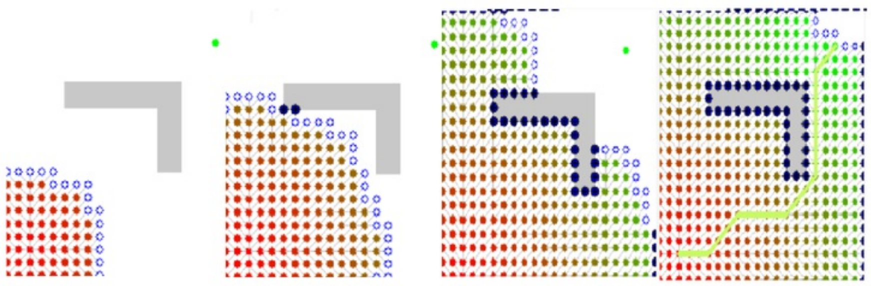

Figure 2.7: Dijkstras algorithm animation by Subhrajit Bhattachary [5]

The algorithm works beginning at the starting x y position then checking the indexes touching it. The indexes being touched become new arrays until the final point is reached. Once this point is reached the array which began this path then becomes the start of the path to be taken. This algorithm works until speed is a factor then the weighting of each position becomes different.

2.3.2 Physical Design Choices and Hardware Specifications

myRIO - 1900

The myRIO is a portable Reconfigurable In/Out (RIO) device that allow students to design control, robotics and mechatronics systems. This has been provided by National Instruments and must be the main controller of the robot. There are 37 digital IN/OUT pins available on either side of the myRIO microcontroller many of these pins have secondary function. Pulse width modulation (PWM) and Encoder (ENC) secondary functions are useful in the design of our robot. If you would like to see more specifications please refer to the national instruments myRIO manual [6].

Relevant Specifications

- Voltage input : 6-16V
- Size: 136.6mm x 88.6mm x 24.7mm

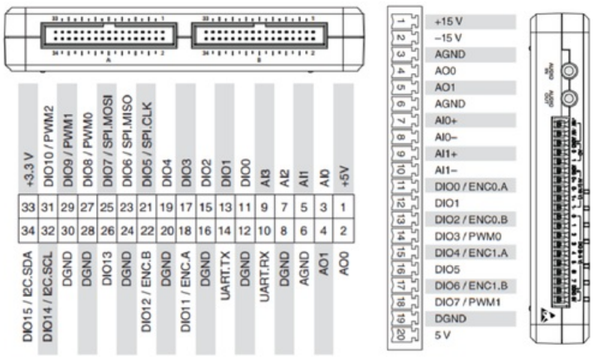

Figure 2.8: myRIO in/out pins, connector ports A,B and C [6]

Mecanum Wheels

Have obtained the mecanum wheels courtesy of the NIARC 2016 team. The set of mecanum wheels were purchased from Seed Studios [3]. The drive system with the given algorithms depicted in our research shown in Figure 2.6 gives us the ability to enter an angle and speed for the robot to travel upon. The advantage of mecanum drives as depicted in research of drive systems is that the mecanum wheels allows us to maneuverer the course without having to rotate the vehicle allowing the sensors to continuously scan a wall it is moving towards where a robot which turns would not be able to achieve this as efficiently.

Relevant Specifications

- Diameter: 60mm
- Width: 31mm
- Circumference: 188.5mm
- Maximum Weight: 10kg
- Number of wheels: 4
- Total Cost: $99 (USD)

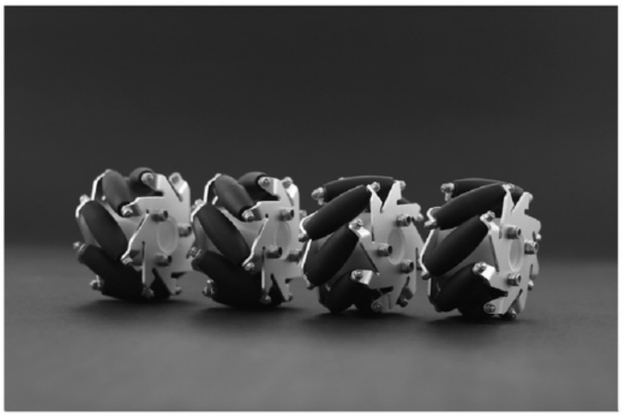

Figure 2.9: Mecanum wheels purchased from seed studios [3]

NeveRest 20 Geared motors and Encoders

Courtesy of the NIARC 2016 team. The NeveRest 20 gear motors were purchased from Andy Mark [8]. They feature a built-in encoder am-3102 attached to the rear of the motor. The encoder has 7 contact zones which stimulate 7 pulses per revolution (ppr). The motor is geared 20:1 and single rotation will yield 140 pulses. Since the encoder has 2 contacts 90 degrees out of phase we are able to determine which direction the encoder is moving from remembering the previous value [7]. Embedded in the myRIO is a secondary

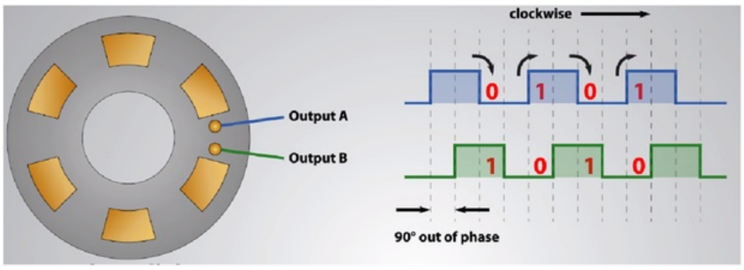

Figure 2.10: Inner workings of quadrature encoder A and B phases [7]

function in the DIO pins called ENC which accommodates for encoders to ensure a simple setup and a reduced amount of time coding in LabVIEW [6].

This motor is susceptible to failure under shock loads [8] which is highest when the robot is quickly taking off and stopping suddenly as it demands a large current to achieve. However from the NIARC 2016 teams experience they have found it to be able to go from 0 to full throttle without issues. PWM is a secondary function on the myRIO. It is used to control speed of the motor and is done by altering the duty cycle between 0% and 100%. The duty cycle is the percentage that that wave is activated for each cycle manipulating the average voltage and changing the overall speed of the motor.

Relevant Specifications

- Voltage: 12V DC
- No Load Speed at Shaft: 275-315 RPM
- Stall Torque: 14.2 kg-cm
- Stall Current: 11.5A
- Weight: 345.6g
- Gearbox Reduction: 20:1
- Number of motors: 4
- Total Cost: $112

DC Motor Driver Lite

Courtesy of the NIARC 2016 team. The 2 motor drivers have been purchased from DFRobot [9]. The drivers use 4 high performance and high current driver chips (BTS7960) which have an in-built protection circuit board (PCB) for current short, temperature and voltage protection. Each driver is able to control 2 motors and each motor requires PWM

Figure 2.11: NeveRest Geared Motor with built in am-301 encoder from Andy Mark [8]

control for the speed and an enable pin to control the direction. The motors drivers have a mutual 5V and GND.

Relevant Specifications

- Input Voltage: 4.8 - 35 V
- Maximum output current: 15 A at 13.8 V
- Peak output current: 20 A @13.8 V
- PWM capability: 25 kHz
- Size: 73 mm x 68 mm x 14 mm

Ultrasonic Sensor HC-SR04

The ultrasonic sensor uses a high frequency sound wave which is undetectable to the human ear. The sound wave that is produced is sent out to the front of the sensor when it is triggered by the ECHO pin [10]. The sound wave is deflected off any objects in front of the sensor and is received by the same sensor which triggers the TRIG pin. Note: Using multiple sensors may interfere with the readings given by the sensors, particularly over larger distances.

Relevant Specifications

- Input Voltage: 5V
- Minimum Range: 20mm
- Maximum Range: 4000mm

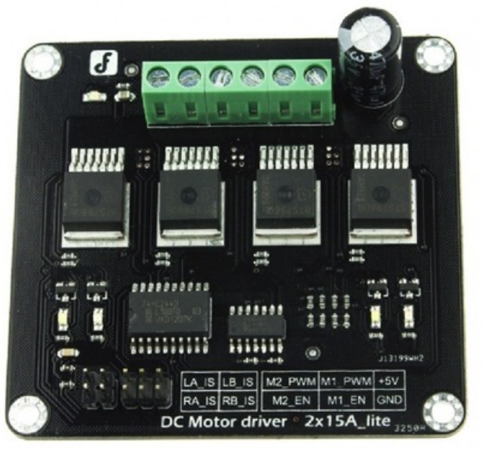

Figure 2.12: DC Motor Driver Lite from DFRobot [9]

- Measuring Angle: 15 degrees
- Size: 45mm x 20mm x 15mm

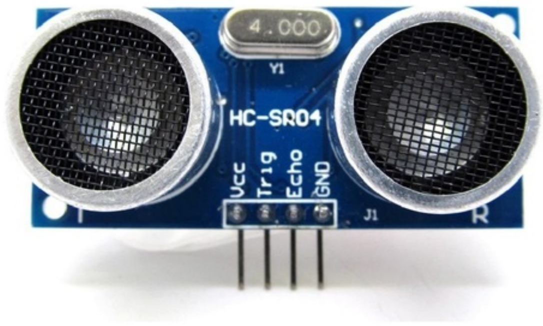

Figure 2.13: HC-SR04 Ultrasonic sensor [10]

Zippy Lithium Polymer Battery

Purchasing this battery to reuse another which had already been purchased, in series to save money rather than just buying a battery with more cells. Lithium polymer have a high discharge chemistry without losing efficiency for capacity. The battery must be charged with a balancing charger [11]. Placing 2 batteries in series will double the voltage being supplied to the motors and therefore will double the maximum speed of the robot.

Relevant Specifications

- Capacity: 5000mAh
- Voltage: 3 cells 11.1V
- Discharge: 30C Constant
- Weight: 408g
- Dimensions: 144mm x 51mm x 27mm

Figure 2.14: Zippy 3S1P 11.1V 30C discharge [11]

Chassis and Security

As shown in the appendix, reimplementing the design used in NIARC 2016 but making a few changes to the length of the vehicle. 3D printed parts are used to fit tightly around the motors and hold them in place when they are fastened to the base of the robot using large bolts. The Base is a clear polycarbonate 300mm x 150mm x 6mm.

In the centre of the chassis there is enough room for the myRIO and 2 motor drivers. They are supported with double sided tape to allow for easy detachment without tools. The top layer of the robot is then applied it is made up of another sheet of clear polycarbonate 200mm x 150mm x 6mm. On this there are 4 handmade folded pieces of clear polycarbonate with holes for the ultrasonic sensors supported with 2 small screws each. The top layer is supported by a bolt screwed into the pre-drilled hole on the 3D printed fastener. The batteries fit smug between the polycarbonate top piece and tall bolts. The front and back ultrasonics are secured with double sided tape to the 3D printed fastener.

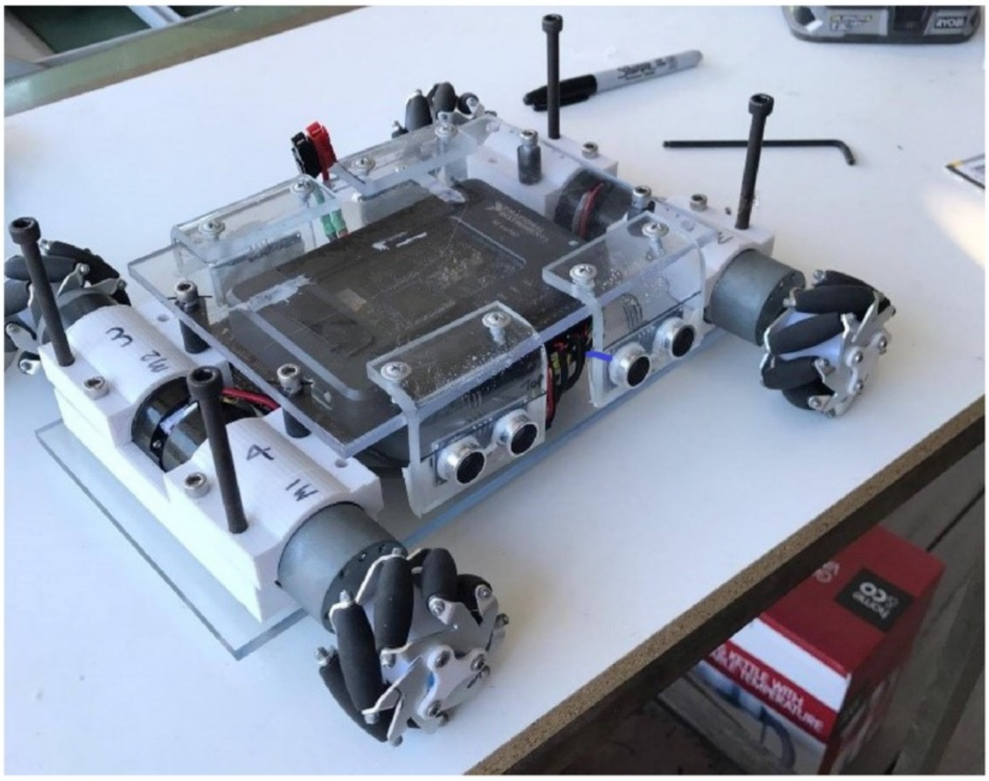

Figure 2.15: Robot Mid Build

Circuit Schematic

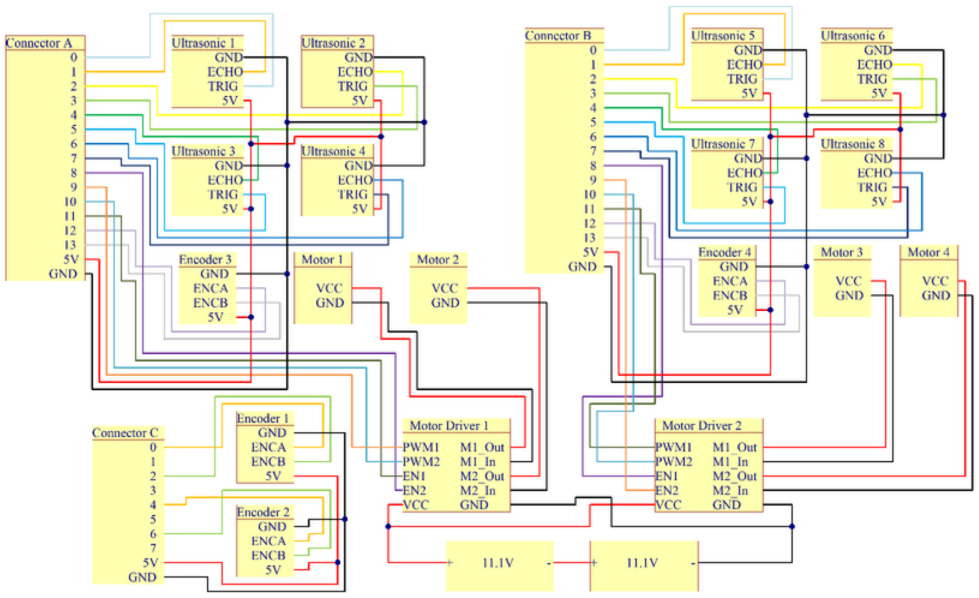

Figure 2.16: Complete circuit diagram created using circuit maker

Chapter 3

Strategy and Techniques to Navigate Environment

3.1 Encoder Based Coordinate system using Mecanum Drive Calculation

The encoder based coordinate systems takes the encoder values and is able to translate this information into a x y coordinate. Each turn of a mecanum wheel contributes 50 percent to forward movement and 50 percent to sideways movement as depicted in Figure 3.1. Equation 3.1 and 3.2 represents this calculation, e1,2,3,4 represent the encoder values and Lx and Ly represent the current x and y location

$$\text{Lx} = \frac{-e1 + e2 + e3 - e4}{2} * \frac{2\pi r}{\text{Ticks per rotation}} \tag{3.1}$$

$$\text{Ly} = \frac{e1 + e2 + e3 + e4}{2} * \frac{2\pi r}{\text{Ticks per rotation}} \tag{3.2}$$

Using this position, we are then able to determine the direction towards our intended position. Taking the intended coordinates and subtracting them from the current location it becomes simple Pythagoras to solve for distance. With this distance sine rule can be used to calculate our angle which can then be substituted into our Simple Mecanum Drive Calculations2.6

3.2 Angle and position corrections

Angle correction utilizes the ultrasonic sensors and ensures they have not veered off path. At strategic points on the track the robot will check its front, side or rear sensors. The sensors are placed 10cm apart and if the robot is square with the wall the sensors should read the same value. However, if the robot does not read the same values for those sensors it will rotate slowly until it finds itself square with the wall.

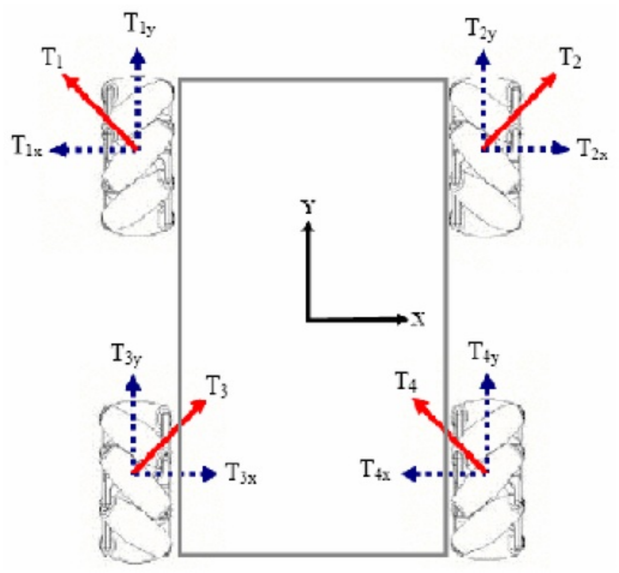

Figure 3.1: Force vectors of Mecanum Wheels [12]

Position correction then occurs once it is straightened up. It uses a single ultrasonic from the front/rear and one of the sides and compares it to its encoder calculated position. If it matches it carries on with the course otherwise it adds an error value to its current position to correct itself.

3.3 Path Planning Algorithm

One of the main objectives of the robot was to travel through a city grid whilst avoiding road blocks and abiding by the school zone speeds. As shown in the figure below the robot must start in position (1,1) and complete the zone at position (7,1). Taking inspiration from [5] we derived our own similar algorithm for navigating a changing environment. At the start the robot receives a data packet consisting of a string of locations where the road blocks could be located. The possible locations are given and are represented on the map by a Letter from A through I and 1 through 8. As the map is relatively simplistic we are able to represent it using an array of switches consisting of 0 or 1 to represent a clear path or obstacle. The map below you can see all dark blue squares and dark green square are pre-considered to be an obstacle and therefore 1 and all others pre considered to be 0.

The map has now been initialized and is ready to receive a data packet. The packet received will be translatable to the road block locations. For instance, if Road Block A was received the location (2,1) would switch from 0 to 1 to indicate an obstacle is present. Now all obstacles are known in the grid and a safe passage can now be determined using the algorithm. It takes the (x,y) position and checks one space left, right, up and down. If the position is clear is fills the n position of the string if the n position is already filled it then creates a new string. When a position is checked it is then marked as an obstacle to avoid overlapping of strings. If a string cannot move in any direction that string path is terminated. The algorithm continues until the position (7,1) is reached. The path which arrived with the smallest n is then considered the least distance to travel. However, reaching the destination in the smallest amount of time is the goal. To fix this if the vehicle travels through 3,5,D or E the array will read the same position twice to account for the delay of slowing to the school zone speed.

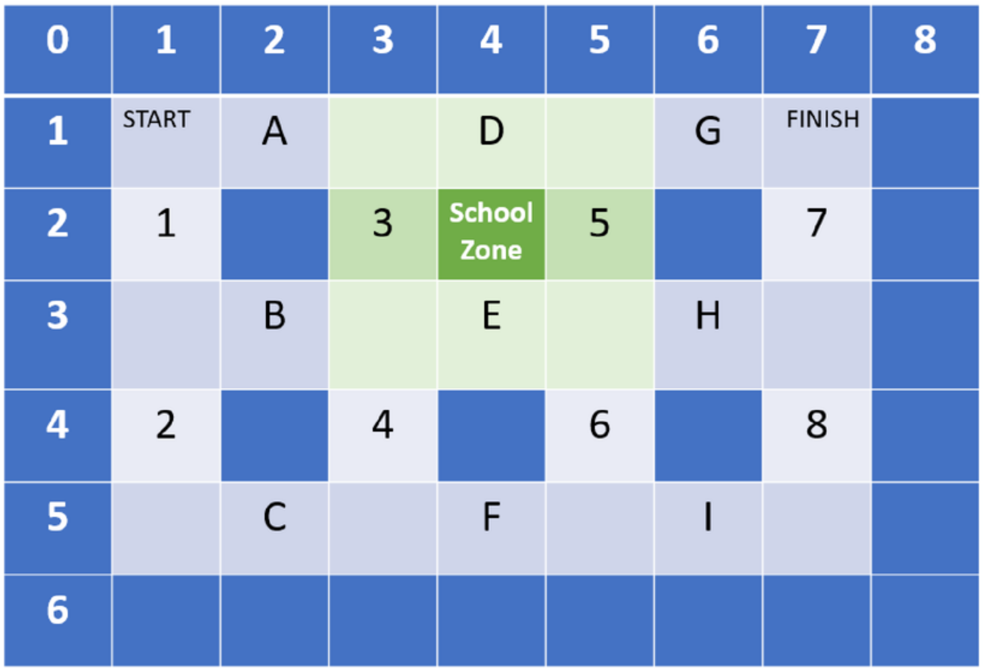

Figure 3.2: City grid obstacle guide for path planning

Chapter 4

LabVIEW Architecture

4.1 FPGA Inputs and Outputs

FPGAs most attractive feature is the ability to rapidly process data from an input and almost instantaneously produce an output. FPGAs are used to monitor the real-time data produced from the 8 ultrasonic sensors and 4 rotary encoders. This data is used to control the speed and direction of the 4 motors which ultimately enable it to navigate its environment.

The following code in 4.1 is difficult to portray LabVIEW being a highly visual coding language. This is the FPGA for the motors. In order to control each motor they require a zero or one for the direction and also a PWM signal. The top left and right have 4 switches which control the direction the motor travels these are not controlled here and are only outputted. Next we have 4 sequence structures. Each structure is identicle to the next other than the PWM pin in which is is outputting towards. The PWM has a duty cycle which can be altered between 0-100 to control the speed of the motor. The start of the sequence in this case intializes the pin to be set to another value, the FPGA runs extremely fast and had difficulty switching simultaneously with real time motors. Adding a short 10us delay allowed a smoother journey for the robot.

Working coinside with motors is the encoders which also utilize the onboard FPGA chips. The following is the code 4.2 for a single encoder out of a total 4. The encoder has 2 inputs which are the represented by the 2 purple rectangles to the left. These inputs are known as A and B and are able to determine which direction the motor is spinning based on their values. The FPGA will quickly process this information and will add or subtract 1 depending on which which way the encoder ticks. The blue and gray box to the far right is the output of the encoder which represents the angular displacement of the wheel.

The final type of sensor which we use the FPGA to receive data from is the ultrasonic sensors. The ultrasonics sensors work using a transmit and receive pin. The received data

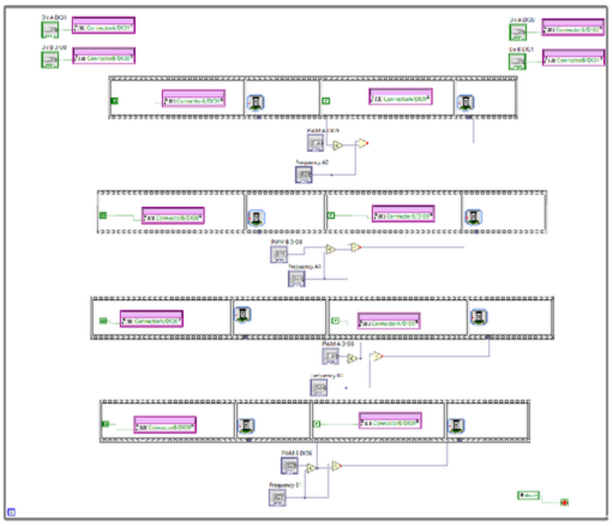

Figure 4.1: FPGA code for motors to control direction and PWM output

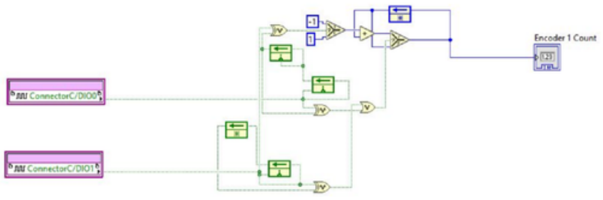

Figure 4.2: FPGA code for quadrature encoders

captured from the sensor will initially process the time of flight which is how long the sound wave has travelled from the transmitter to an obstacle in front of it and back to the receiver. The purple rectangle is the receiving pin and the time of flight is calculated.4.3

4.2 Motor Calculations

Using the calculations derived from Simplistic Control of Mecanum Drive [4] each motor has a unique equation for determining the speed and direction it must travel in order to successfully reach the designated location. This equation is found within the blue rectangle for PWM control. Given an X,Y coordinate the calculations are able to obtain an angle and throttle plan to achieve this destination. The output from each motor calculation to determine the speed and direction is fed into the motor FPGA which is shown on the right 4.4. At the left of the same image there is a proportional control for

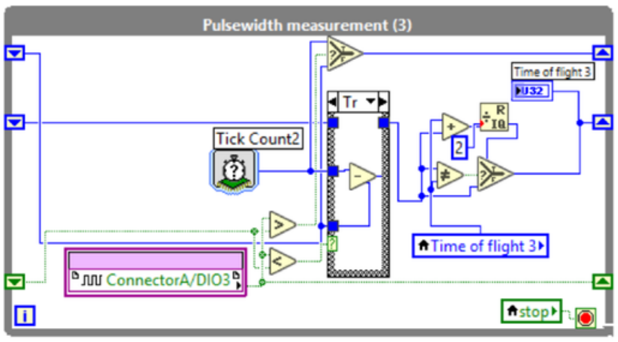

Figure 4.3: FPGA code for Ultrasonic sensors

when the robot begins to near the set point it will slow down so it may accurately stop.

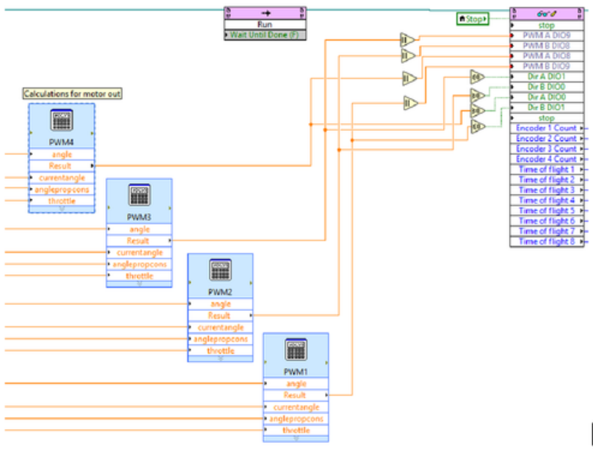

Figure 4.4: Motor calculations for proportional control and angle control

4.3 Encoder Position and Angle Calculations

Each mecanum wheel travels the same distance in the x and y direction as it works along a 45 degree angle. The calculations take the amount of ticks from each encoder add them respectively (depending on their orientation) and then multiply them to achieve a distance which can be used as an X and Y coordinate. The centre formula 4.5for angle error is a calculation for rotating the robot when an angle error is detected either via ultrasonics or encoders.

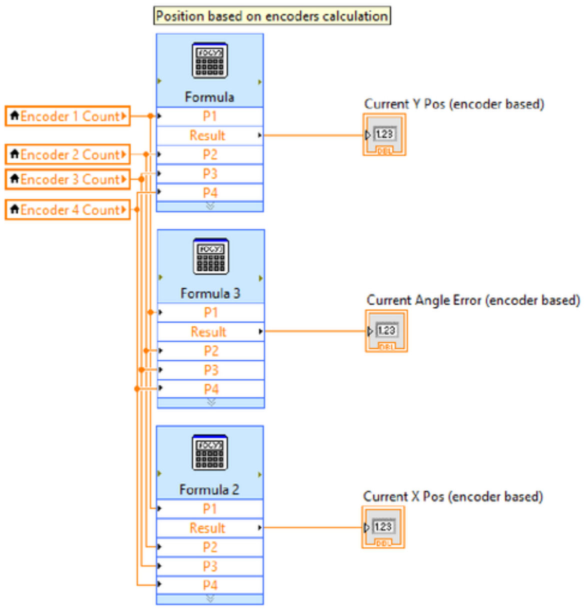
Position based on encoders calculation
Encoder 1 Count
Encoder 2 Count
Encoder 3 Count
Encoder 4 Count
Formula
P1
Result
P2
P3
P4
Formula 3
Formula 2
Current Y Pos (encoder based)
Current Angle Error (encoder based)
Current X Pos (encoder based)

Figure 4.5: Encoder Position and Angle Calculations

4.4 Coordinate Position Control

Using an array of x y coordinates the motors and encoders calculate the distance and angle from the current x y coordinate to the designated x y coordinate. The robot progresses through the coordinates in real time with the help of case structures. The case will call the nth value in each string of x y values and travel to the set position before processing the next set position.

4.5 Path Selection Algorithm

As stated earlier [5] the path selection algorithm takes it current position and checks one space left, right, up and down. If the space is open it will add the x y coordinate to the nth position of the array if however that nth position is already filled a copy of that array will be made and the nth position will be replaced. This x y position will then be marked as an obstacle because if it isnt it will cause overlapping in other paths. The process will continue until the final position has been achieved. The array which achieves this in the smallest amount of steps will become the route in which the robot will take.

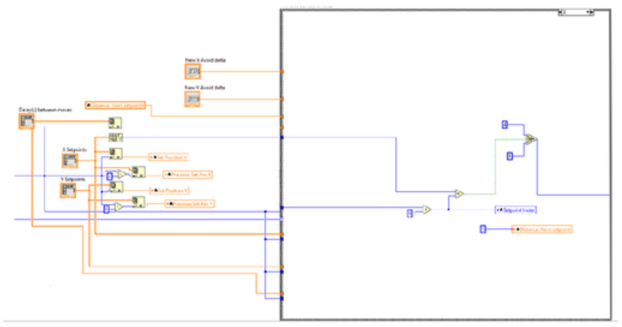
Figure 4.6: Coordinate Position Control Sequence

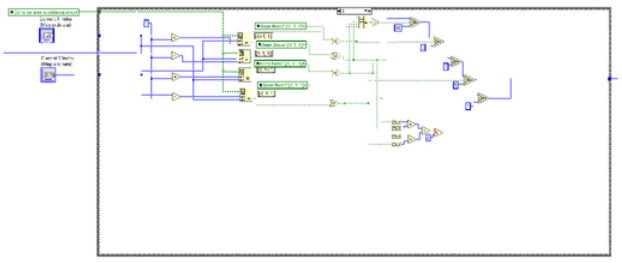
Figure 4.7: Path Planning Sequence and Algorithm

4.6 Ultrasonics Time of Flight Conversion

The ultrasonics time of flight value is converted into meters so it may be easily interpreted as shown in equation 4.1. To convert the time of flight to meters the following equation may be used. Some of the ultrasonic sensors read a slightly larger value, this may have been because of the manufacturing of the chassis caused it to be positioned out of alignment.

$$\text{meters} = \frac{\frac{\text{Time of Flight}}{10^6} * \text{speed of sound}}{2} \tag{4.1}$$

4.7 Ultrasonics Position and Angle correction

It is important to note that the ultrasonic sensors sometimes have outliers which may need to be rejected. An average of 10 readings from each ultrasonic is taken, this average will then be used to determine if an action is required.

The use of ultrasonics is a very important tool in improving the accuracy of our encoders. When the ultrasonics can detect a wall, they are able to correct the encoders by updating its x y coordinate. For example, if the robot senses a wall 20cm to its left when it is intended to be 30cm the robot will change its x position to compensate for the 10cm error. This localization removes all error created from inaccuracies in the encoders. There may also be inaccuracies in the angle of the robot. The angle of the robot is calculated using the 2 ultrasonic sensors on each side. If there is a difference in reading between the 2, the robot will know that there is an error in angle in which needs to be corrected by the motors.

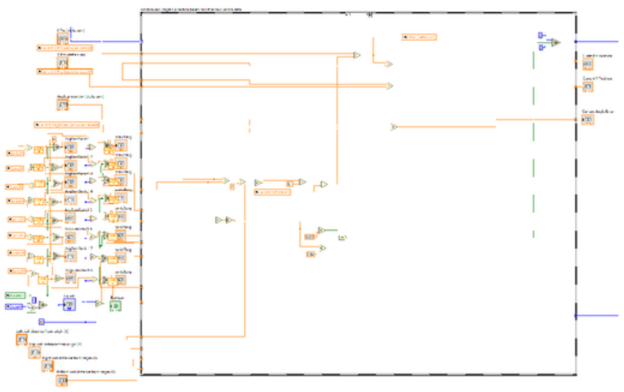

Figure 4.8: Position and Angle Correction Calculations

4.8 Live Final

On the morning of the competition team MQMechatronics was first to run the track. The robot had to be recompiled when connecting to the wifi server. Taking us by surprise the FPGA takes 10 minutes to recompile a new FPGA and forced us to forfeit the first two heats of the competition. The final round we were able to compete, however, the wifi server was having difficulties and forced us to miss the wifi packet unfortunately crash into a road block which was not registered in our path planning algorithm. This was not a planned circumstance and ultimately terminated our position in the next round. It would have paid off to have a failsafe installed in the robots logic in which it would detect there is a wall and move around the obstacle.

There was many issues with our wiring that had to be debugged and rewired as rigorous testing the day before had loosened the wires. It would have been much easier to debug if the chassis was easily dismountable and the wires were organised with zip ties and colour coordinated. Our final design was simple yet effective and I dont think this was a mistake, however our time management and motivation was far more relaxed than required and cost us the qualifying rounds.

Chapter 5

Conclusion

Overall team MQMechatronics was able to make it to live competition, create a path planning algorithm, have a localisation system and an obstacle avoidance system. In the future if I have the opportunity to participate in another robotics competition I should make sure our team are setting deadlines and keeping to them. The milestones set by National Instruments gave us a false sense of hope thinking we were ahead of the competition when the fact was we were behind of schedule.

Bibliography

[1] 2017. [Online]. Available: https://en.wikibooks.org/wiki/Robotics/Types_of_Robots/Wheeled

[2] 2017. [Online]. Available: http://www.robotshop.com/en/4wd-omni-directional-arduino-compatible-mobile-robot.html

[3] 2017. [Online]. Available: https://www.seeedstudio.com/Mecanum-Wheel-Kit-%282-Left%2C--2-Right%29-p-2171.html

[4] I. McInerney, *Simplistic Control of Mecanum Drive*, 2017. [Online]. Available: http://thinktank.wpi.edu/resources/346/ControllingMecanumDrive.pdf

[5] N. Correll, "Introduction to robotics 4: Path-planning," 2017. [Online]. Available: http://correll.cs.colorado.edu/?p=965

[6] National Instruments, 2017. [Online]. Available: http://www.ni.com/pdf/manuals/376047c.pdf

[7] 2017. [Online]. Available: http://howtomechatronics.com/tutorials/arduino/rotary-encoder-works-use-arduino/

[8] N. (am 3102), "Neverest classic 20 gearmotor (am-3102)," 2017. [Online]. Available: https://www.andymark.com/NeveRest-20-12V-Gearmotor-p/am-3102.htm

[9] 2017. [Online]. Available: https://www.dfrobot.com/product-796.html

[10] 2017. [Online]. Available: https://www.jaycar.com.au/arduino-compatible-dual-ultrasonic-sensor-module/p/XC4442

[11] 2017. [Online]. Available: https://hobbyking.com/en_us/zippy-flightmax-5000mah-2s1p-20c.html?countrycode=AU&gclid=EAIaIQobChMI9L-k57On1wIVkR0rCh3ItQO9EAQYASABEgJXf_D_BwE&gclsrc=aw.ds&___store=en_us

[12] 2017. [Online]. Available: https://www.researchgate.net/figure/268326364_fig2_Figure-2-Force-vectors-created-by-Mecanum-wheel

[13] Australia NI, 2017. [Online]. Available: http://australia.ni.com/sites/default/files/Task%20and%20Rules%20NI%20ARC%202017_Revised%2011%20Aug.pdf

[14] R. Roboticists, H. Abbass, T. Conversation, F. Tobe, O. Mitchell, B. Smith, R. Editors, M. News, J. Rasiel, and B. e. a. Smith, "Pros and cons for different types of drive selection — robohub," 2017. [Online]. Available: http://robohub.org/pros-and-cons-for-different-types-of-drive-selection/