EE198 – Robotics System Project

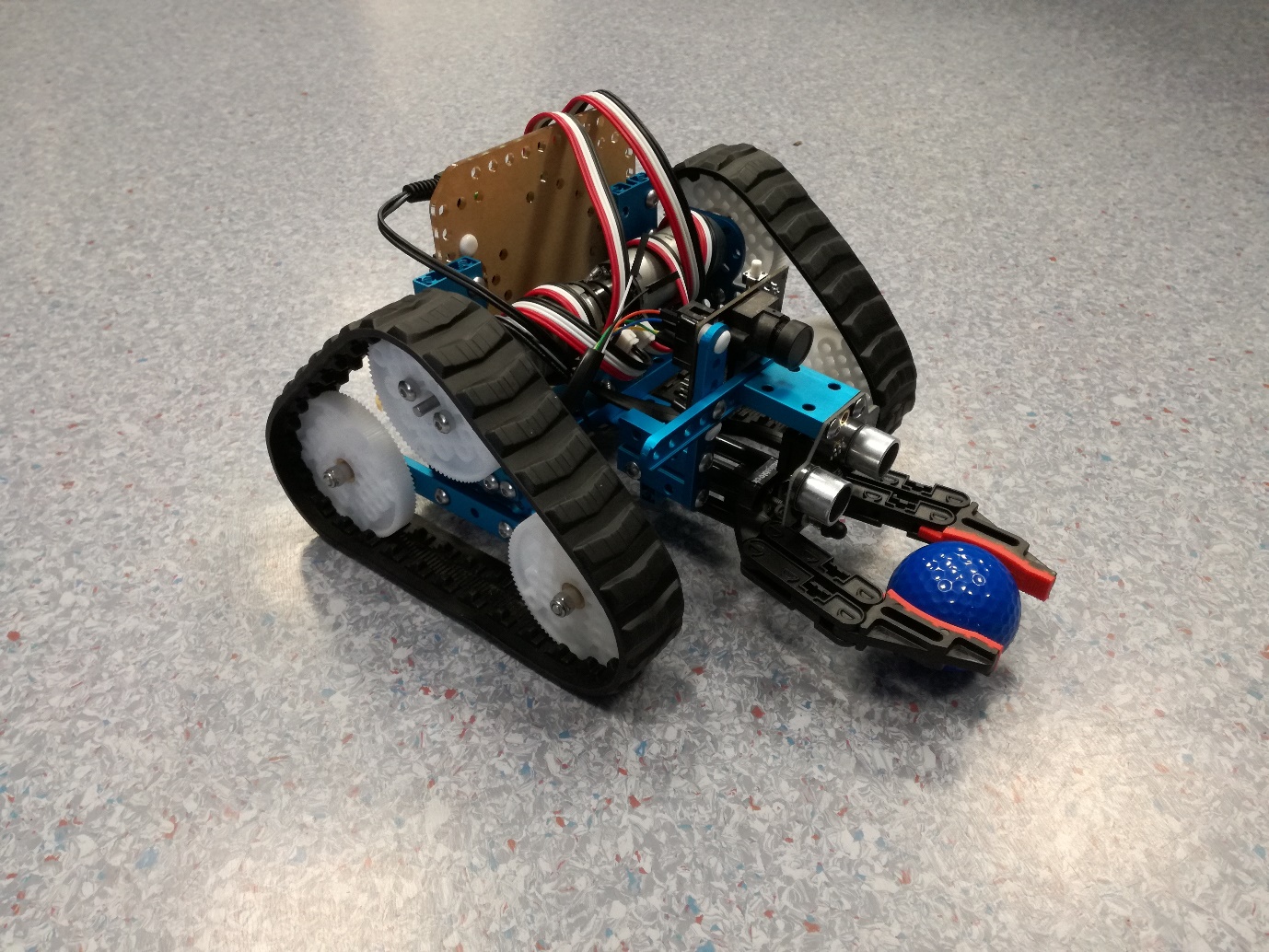
Team 5

Adam Duke – 17718781

Ronan Carroll – 17704351

Michael Codd - 17398106

**Final Report**



# Abstract

The following document is a report on the project K3V1N. The report aims to give an in depth and complete overview of this project. The report will cover the background of the project to lay a foundation for its purposes as well as explain the construction process and reasoning behind the robot design chosen. The software implementation for the motors used and the PixyMon camera are discussed and analysed. The testing process is then described, and results are demonstrated, explained and analysed.

# Acknowledgements

We want to thank John Dooley, project supervisor, for his help in guiding and encouraging us along the way. Violetta McLoone for her help with the coding process and her advice on report structure. John Maloco for his input when we met challenges in coding our robot. Andrew Meehan for being safety officer and for taking part in the interviews.

Table of Contents

[Abstract 2](#_Toc513198947)

[Acknowledgements 3](#_Toc513198948)

[1. Introduction 5](#_Toc513198949)

[2. Task 5](#_Toc513198950)

[3. Background 6](#_Toc513198951)

[3.1. Encoder Motor 6](#_Toc513198952)

[3.2. DC Motor 7](#_Toc513198953)

[3.3. PixyMon 8](#_Toc513198954)

[3.4. Organisation 9](#_Toc513198955)

[4. Construction 10](#_Toc513198956)

[3.4. Pros and Cons 10](#_Toc513198957)

[3.5. Final Design 12](#_Toc513198958)

[4. Algorithm 13](#_Toc513198959)

[5. Software Implementation 15](#_Toc513198960)

[5.1. Encoder Motors 15](#_Toc513198961)

[5.2. DC Motors 15](#_Toc513198962)

[5.3. PixyMon 16](#_Toc513198963)

[6. Testing 17](#_Toc513198964)

[6.1. Initial Testing 17](#_Toc513198965)

[6.2. Final Testing 18](#_Toc513198966)

[7. Conclusion and Future Work 19](#_Toc513198967)

[8. References 21](#_Toc513198968)

[Appendices 22](#_Toc513198969)

# Introduction

Participants in the first year Robotic System and Intelligent devices course in Maynooth University must complete the module EE198. This module is a robotics project where the students are tasked with designing and constructing a robot from a specified kit, that must accomplish a set goal by the end of the module. The purpose of this of this course is to familiarise the students with group projects, analytical and problem-solving skills, engineering production, and technical report writing. Other learning outcomes include individual and collaborative research as well as a familiarisation with work presentation and communication. The project-based learning would also familiarise the students with the tools, systems, and components of robotics throughout the process.

The goal set for this project was to design, construct and programme a robot with the Ultimate 2.0 kit that would accomplish distinguishing, separating and relocating targets. This was to be performed within a determined area and to be done with as much efficiency as possible in terms of speed, time taken and programming. The robot would then perform this task and be graded on its efficacy accordingly. Throughout the course of the module, status updates were required in the forms of reflective journals, interim reports, and presentations.

The teams project name is K3V1N, and its members are Michael Codd, Ronan Carroll, and Adam Duke. This report details the efforts that were undertaken to complete this project. The strategies that were incorporated in meeting the requirements of the project and the methods used throughout the process from beginning until completion.

# Task

Using the Ultimate 2.0 kit, each team had to design, construct and code a robot. This robot must then complete its objective and perform the task as efficiently as possible. The area in which each robot would perform the objective was a two metre by two metre walled arena, each wall being 30cm high with a designated 30cm square goal area in one corner. Within this arena nine golf balls were placed at random. The golf balls were painted in three separate colours red, blue, and green. The objective was for the robot to distinguish the red coloured golf balls from the other two colours, and to transport the red target balls to the goal area. The robots would be graded by the time and speed at which this task was completed. There was also points deducted from the grade for any non-red golf-balls remained in the target area upon completion of the objective.

The robot used had to be constructed from components found in the Ultimate 2.0 kit. This kit had multiple components both mechanical and electrical for creating small robotic designs, the kit has a MegaPi control board for storing and running code and a large AA battery pack for power. There are several mechanical parts including tyres, wheels, and gears. There are also a range of electronic components; Bluetooth module, ultrasonic sensor, line following sensor, triple axis accelerometer and gyroscope, a motorized gripping claw and 3 encoder motors. The kit also included a wide range of support struts and lattices, and various other construction pieces for the structure of the robot. The only other item allowed to be used in the design was the PixyCam camera. The PixyCam is a small PCB camera with a lens and optical sensor. It has a wide range of internal capabilities such as colour recognition and Arduino integration. The only other parameter of the project was that in the coding of the robot the programming software “mBlock” was forbidden. Any coding or programming would have to be done through the alternative languages readable by the MegaPi, namely, Arduino.

# Background

## Encoder Motor

Encoder motors are based around the concept of always knowing in what position the shaft is rotating, the speed of rotation and the number of total rotations. This is often achieved by placing a disk with slots in it on the shaft of the motor and using a photodiode and a photodetector on the opposite side, it is possible to count the number of rotations that the shaft makes.

There are 2 different types of encoder motors; Incremental and Absolute [1]. Incremental Encoders can be further broken down into non-quadrature and quadrature encoders. Incremental non-quadrature encoders provide a single positive pulse to the controlling computer each time the shaft rotates. This makes it possible to calculate speed and acceleration. A quadrature encoder provides two pulses, out of phase, per rotation, which can also be used to calculate the direction of rotation of the encoder. An Absolute encoder can provide all this and more, such as the angular change or the shaft and the incremental changes for when the encoder makes small moves that otherwise wouldn’t trigger the Incremental encoder.

Possible uses for an encoder motor would be in a robot that is battery powered, if there was a low battery, the robot could be coded to run for a certain number of degrees of rotation, compared to being coded to run for a certain amount of time, as is more common in DC motors. This therefore provides a much higher level of accuracy and control over the robot’s motions and distance covered

## DC Motor

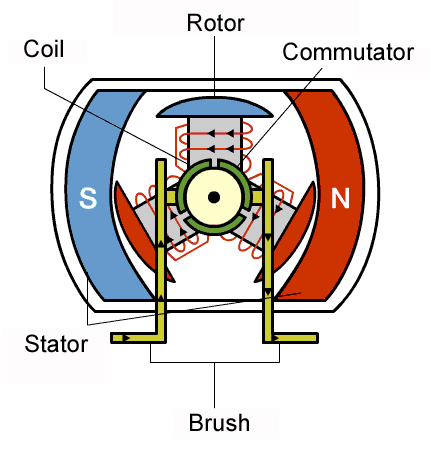
DC motors are divided into two groups; Brushed motors and Brushless motors [2]. Brushed motors are based on the principle of using electrical current flowing through coils that are arranged in a fixed magnetic field (See Fig. 1). As the current flows, it causes the coil to rotate away from the like pole and towards the unlike pole, but as it approaches the unlike pole, the current is flipped, therefore causing the coil to have to continue rotating to ‘chase’ the unlike pole. Power is supplied to the coils through brushes which contact a rotating commutator which is attached to the shaft that the coils are mounted to, therefore rotating with them.

Fig1. Brushed DC Motor [3]

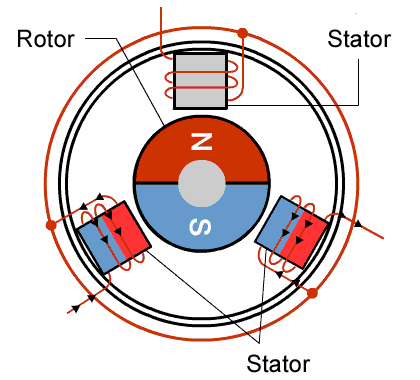
Brushless motors have a constant magnet located in the centre of the shaft that rotates. The coils do not rotate, instead they are fixed on the stator around the perimeter of the shaft (See Fig. 2). Rotation is achieved by controlling the magnetic field generated by the coils on the rotor, while the magnetic field generated by the magnets remains fixed. To modify the rotation speed, you simply change the voltage for the coils.

Fig. 2 Brushless DC Motor [3]

The advantage brushless motors have over brushed are that they are low maintenance, highly efficient with high power to size ratio and a higher speed range than an equivalent brushed motor. Their main disadvantages are a higher cost of construction and their control systems are more expensive than brushed motors, which only require a steady voltage input

## PixyMon

PixyCam (See Fig. 3) created as a kickstarter [4] by charmed labs is a vision sensor for DIY robotics or other similar applications. PixyCam can be taught to highlight and track objects very easily. PixyCam is capable of recognising seven different colour signatures and hundreds of objects simultaneously. It connects to Arduino with an included cable and is C/C++ supported.

Fig. 3 PixyCam board [5]

PixyCam is the result of much research to address two of the main drawbacks of image sensors; they output lots of data and the processing of said data can prove overwhelming for processors. If a processor is unable to keep up with the data rate, the processing power of the robot won’t be available for other tasks and will slow the robot/application down.

Pixy addressed these drawbacks by including a powerful dedicated processor with the image sensor. Instead of sending megabytes of data to the main processor of the application pixy only sends useful information, for example "colour signature 1 is at X=34 Y=76 at a rate of 50Hz". This ensured that the main processor would not be overwhelmed with excessive data and only need to process small amounts of information freeing up resources for other components of the application.

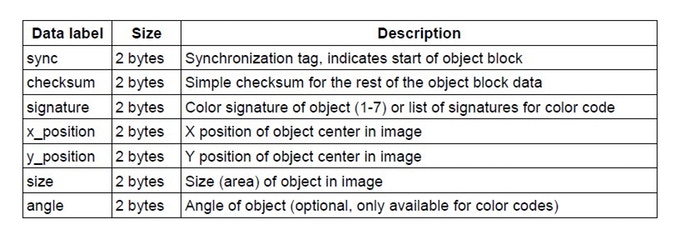
PixyCam uses a hue-based colour filtering algorithm to detect objects. Most people are familiar with RGB (Red Green Blue). Pixy calculates the saturation and hue of each RGB pixel from the image sensor and uses these as primary filtering parameters. It was found that lighting and exposure differences can have a frustrating effect on the filtering algorithm, causing them to not pick up previously assigned signatures or the signatures being poorly recognised. PixyCam processes a 640 X 400 image at 50Hz which outputs the position of detected objects every 20ms.

PixyMon is the open source QT framework companion application for PixyCam, it allows the user to see what Pixy sees either as raw or processed video and provides the tools to configure and calibrate the camera for the required task (See Fig. 4)



Fig. 4 PixyMon colour signature recognition.

Pixy communicates through UART, SPI, I2C, digital or analogue output. The serial interfaces (UART, SPI, I2C) use a simple binary protocol. Every 20ms Pixy outputs a list of objects it has detected with each object represented by an “object Block” (See Fig. 5)

Fig. 5 Data returned by PixyMon camera [4]

## Organisation

The time between the first Laboratory session of EE198 on Tuesday February 6th, when the project was announced and the hand in date for the final report of the project of Friday May 4th was twelve weeks. Given what had to be achieved in this limited time, there was a strong incentive to be as organised and efficient with this time.

To best facilitate communication of ideas, discussions, and coordination the team established a private messenger group on Facebook messenger. This was only accessible by team members and the content within pertained only to the project. For data involving the project such as clips for the vlogs and documentation for presentations a private shared folder was created on Microsoft OneDrive where any content uploaded by a member could be viewed and edited by other members.

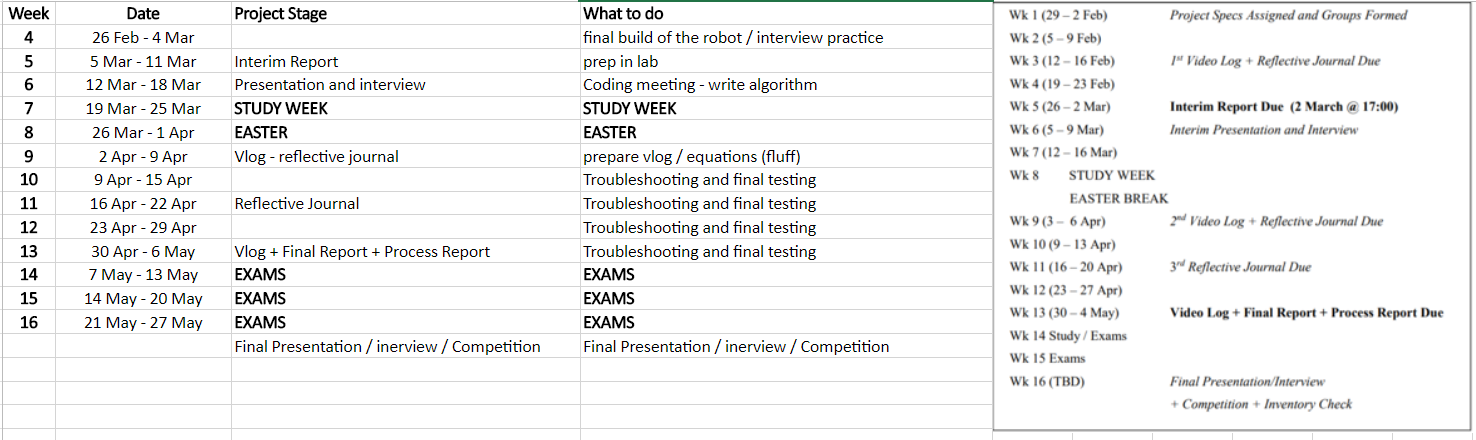
As there were two lab sessions of three hours each week for the project, an additional one-hour meeting at 1pm on Wednesday was decided upon. This peripheral meeting was used to discuss what would be the best use of time in the upcoming lab sessions and sometimes devoted to one topic that required discussion. Topics discussed at these meetings included; writing a weekly schedule (See Fig. 6) to meet the specific interim deadlines on presentations and defining the algorithm necessary for the robot. The additional time devoted for the Wednesday meetings was at its most beneficial when it came to prepare for reports and presentations. The meetings helped to rehearse the interim presentation, and to edit and record the scripted content for the vlog entries for the project.

Fig. 6 Weekly Planner

The course content for EE198 on Moodle contained a weekly planner which acted as a schedule for the deadlines that were required by the project. The team decided to recreate this document but in greater detail and outlined goals and personal deadlines that were felt to be necessary. The teams personalised weekly planner also detailed what each week should be focused. This proved very beneficial to the team in the earlier stages of the project when the bulk of the content was yet to be done.

# Construction

## Pros and Cons

Of the time allocated for each section of the project, construction was given the least. This was determined by the team’s experience with the assembly of the predesigned robot included in the Ultimate 2.0 kit at the onset of the project. The MegaPi board and battery pack were deemed to be potential balance concerns and therefore were configured centrally. Then the PixyCam camera and line sensor were components were connected. With these components in place the skeletal frame of the robot was constructed around these components to the general shape of our design. Once the framework was completed the gripper claw was fastened using a bracket to the front of the robot. The front being determined by the direction of the line follower and PixyCam’s field of view. Once these components were in place the tracked wheels were added to each side perpendicular to the face of the robot. These were added last as they were easily the most manageable of all the components. They could easily be adjusted to allow for the 21mm clearance between the claw gripper and the ground surface. Then additional support structures were attached to ensure that the motors were firmly attached to the frame.

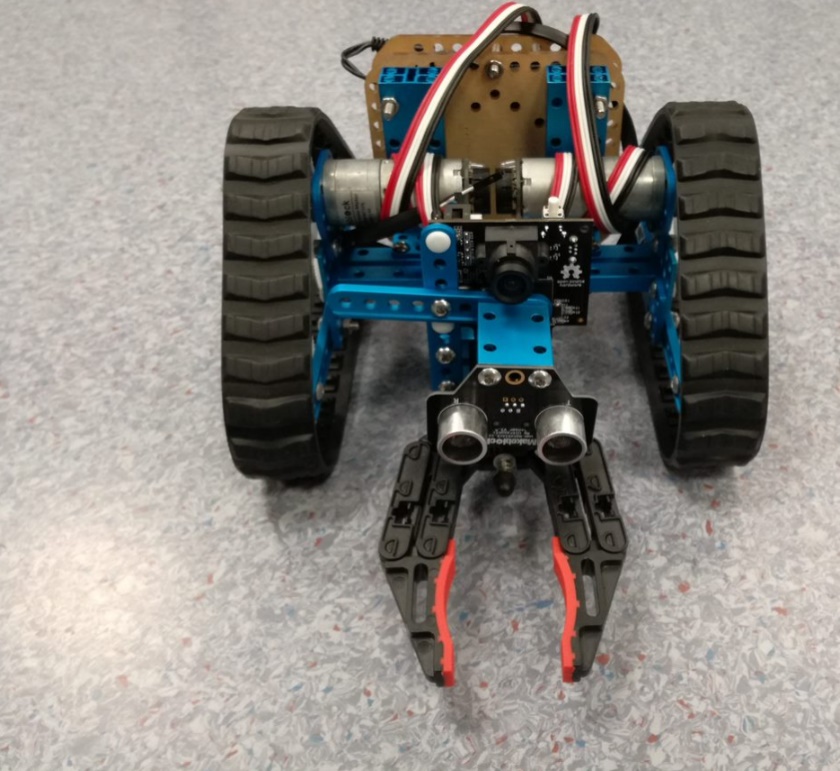
The primary design of the robot initially incorporated two inputs, the PixyMon camera for position and the line following sensor for determining the drop goal zone of the targets. The line sensor was initially placed directly below the gripping claw and the PixyCam camera. The gripping claws position could not be altered as it would fundamentally change the entire design and have an impacting effect on the algorithm. The PixyCam camera was the most relevant input for the entire robot. Its central position was crucial as it required both a view of the pincer mechanism and a wide field of view for ascertaining targets. An attempt was made at positioning the line sensor beside the gripping claw but was rendered pointless as its field of view would have been obstructed once the gripping claw had acquired a target. With the position of both the gripping claw and the PixyCam camera of the highest locational priority, the team concluded that it was more efficient to remove one component entirely compared to reposition two. The removal of the line follower sensor would also have far less impact on the final algorithm, compared to the relocation of the camera with respect to the gripper. Once these changes had been put in place, the final design was realised physically, and the construction section was complete. (See Fig. 7)

Fig. 7 Final Design

## Final Design

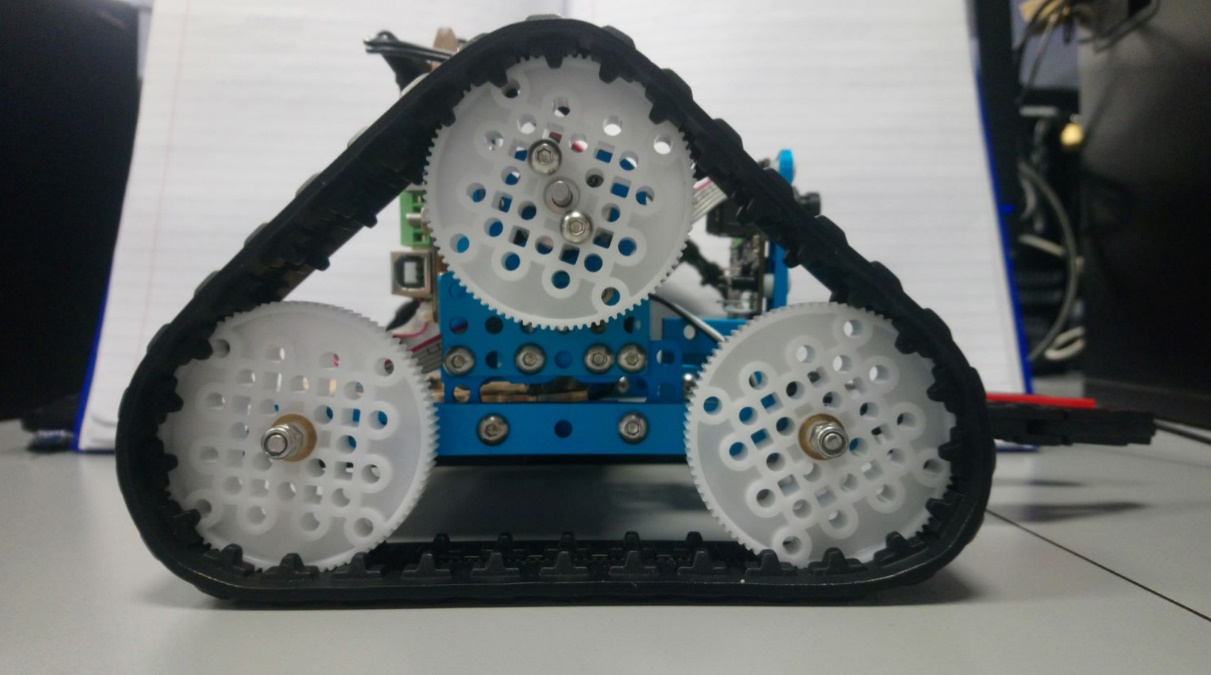
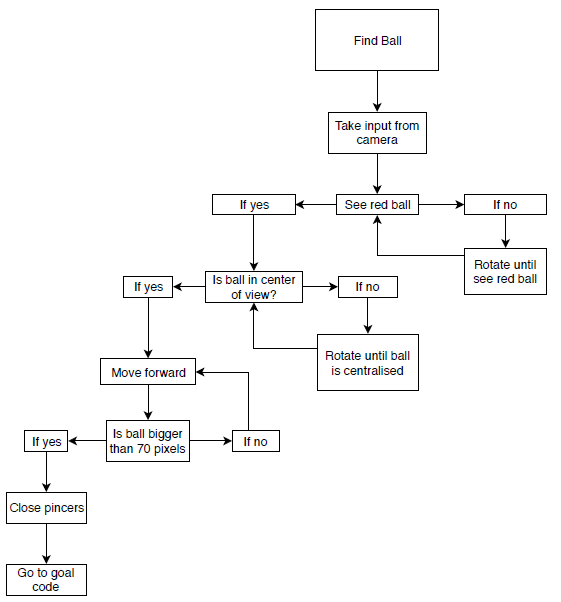
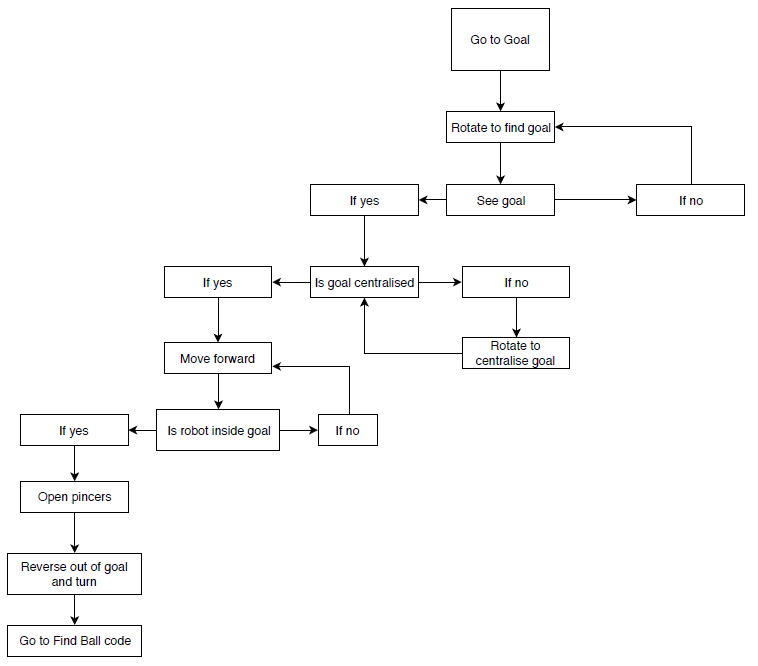
Initially the concepts designed by each member were quite different, with the only common theme being simplicity. The simple designs were reflective of the awareness of each team member that the more components per design increased the amount of programming necessary. The teams final design for the robot was determined through cultivating the aspects of each individual members concept designs and amalgamating the features best suited for purpose. The design is a low-profile design, having the PixyCam camera and claw grabber facing forward, with the battery pack underneath and the MegaPi board at the rear. It is propelled by two separate tracks each containing three rotational wheels at each side. The grabber protrudes from the face of the robot perpendicular to the ground, with a height of 21mm from the surface of the ground as this was measured to be the centre point of a golf ball resting on the ground. This feature was ensuring that the target golf ball will neither roll beneath the gripper nor slip above the pincers when they are engaged. The gripper is rigidly attached to the frame of the robot to ensure that its position remains fixed with the movement of the entire robot. This approach was chosen to reduce the amount of programming with respect to movement, as all coding relevant to position was controlled by the two encoder motors. The tracked wheel system was chosen primarily as it allowed a far smaller turning distance than that of a wheeled system. The wheel-based system on the structural framework would have had a far greater turning radius. The track-based system has a turning radius of only 8.9cm and due to this tighter turning radius, the track-based system was favoured. Another benefit of the track-based system was the total amount of ground surface friction. Surface contact given by the wheel-based system was 1.3cm x 4 wheels which is significantly less than the 3cm x 16cm x 2 tracks provided from the track-based system. (See Fig. 8)

Fig. 8 Side view of K3V1N showing the tracks

# Algorithm

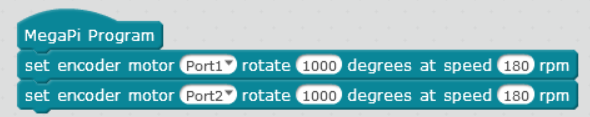




# Software Implementation

## Encoder Motors

The software implementation for an encoder motor is complicated. Since the nature of the motor is to be very precise, the code that runs it is equally precise and complex. Initially a mixture of sample codes was examined, but it was discovered that for a variety of reasons, they wouldn’t work. MBlock was then used to create a sample run code for the Encoder Motors (See Fig. 9) from where the syntax for operating codes would be visible and hopefully replicable.

Fig. 9 Sample mBlock code.

This code was then opened in the Arduino IDE and it was discovered that the code was extremely long and complex. (See Appendix 1) A few hours were spent trying to understand and manipulate the code to try and simplify it down to an easier level wherein it would be possible to implement it for use, but by removing or modifying any lines of the code, it stopped the whole program running. It was decided by the team that a simpler way had to be found for controlling the motors.

## DC Motors

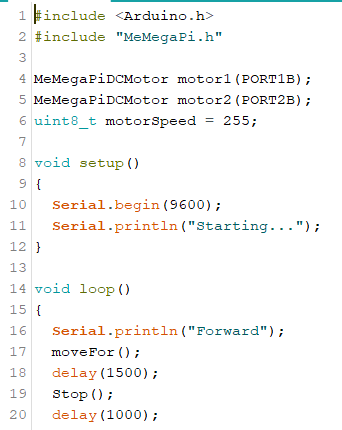
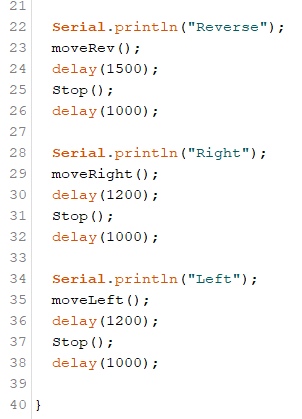
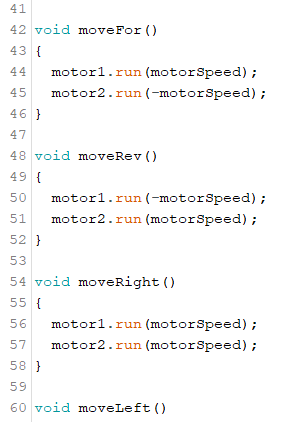
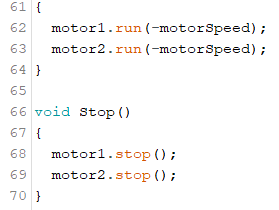
While searching for a simpler way of controlling the motors, the sample programs were examined in greater detail in the hope of finding a simpler method. Eventually a DC motor code (See Fig. 10) was discovered that would be able to run the motors, albeit with less control than the encoder motor could would have provided. After discussing it as a team, it was decided to go ahead with this simpler version of motor code, and therefore it was implemented.

Fig. 10 Simple Motor run code made to practice and troubleshoot

## PixyMon

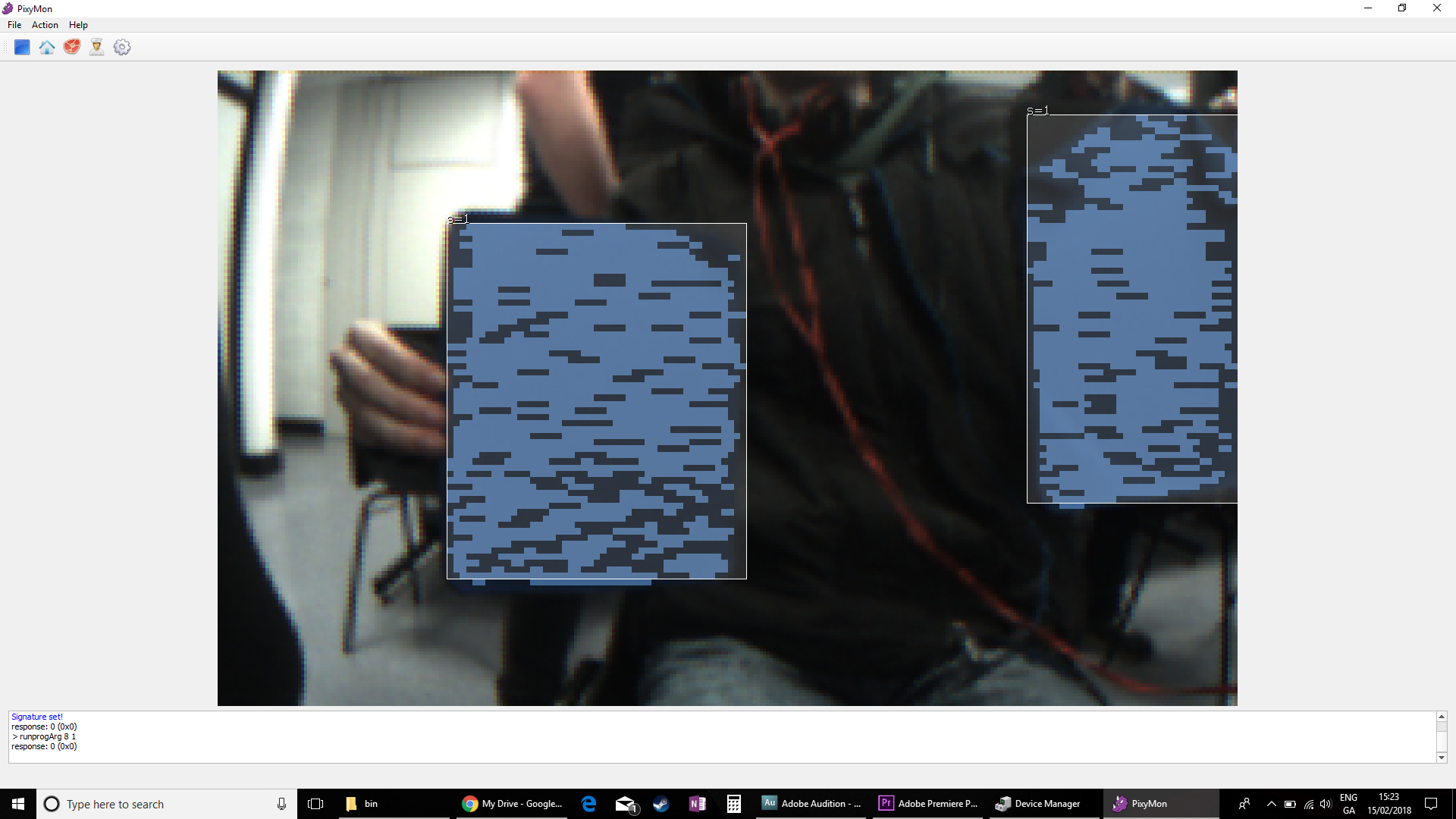
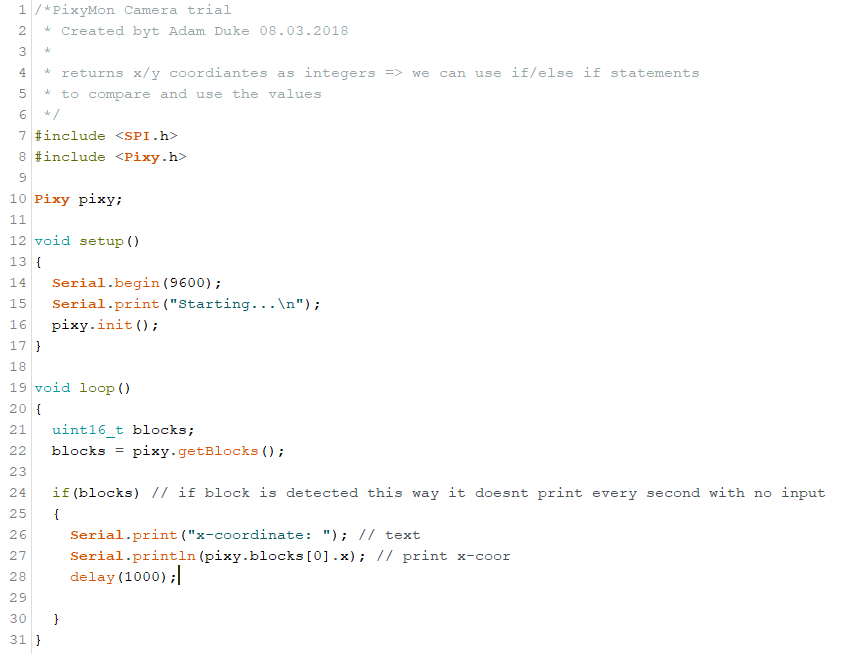
Using the PixyMon software, the camera was connected to a computer via USB. The camera was set up to recognise certain colours by selecting raw video mode then holding an object of a certain colour in front of the camera. The image would freeze when the “Set colour code” command was detected, and the object could then have highlighted using the mouse. From this point on, the camera would now recognise this as a signature and highlighted it as such (See Fig. 11)

Fig. 11 PixyMon colour code signatures

At this point the PixyCam was set up to recognise certain coloured objects but was not sending this information anywhere. The PixyCam was then connected to the MegaPi board using the provided cable. After much research, a website was found ([www.cmucam.org](http://www.cmucam.org)) which contained the relevant commands to connect Pixy to Arduino, and using this information, a sample code was written which was used to examine and learn how to correctly connect the components together

With this code (See Fig. 12), the PixyCam provided all the information required to continue the coding of the robot. The PixyCam, when connected to the MegaPi, sends the x-coordinate of the detected object in integer form, therefore making it easy to compare and use if/else statements to control the robot.

Fig. 12 Arduino code for printing out x-coordinates of object detected with PixyMon

# Testing

## Initial Testing

Using all that was learnt throughout the software implementation and research, the algorithm was implemented into Arduino. Problems were initially discovered wherein the code was unable to run through a mixture of while loops and if statements. This was remedied by making a further series of if statements which would be used to catch any outliers in the output from the PixyCam. This extra catch was implemented due to the camera regularly returning an output of 1367 for the x coordinate, where the expected range was between 0 and 320. The suspected cause of this error was that Arduino was reading in values too quickly and this was therefore causing a bottleneck in the system and causing two values to be connected into one.

Once this error was resolved, another error was discovered wherein the system couldn’t read in the x-coordinate quickly enough and therefore would decide based on an old input, which would therefore not allow the robot to correctly turn and move. A solution was eventually found by restructuring a few of the if statements such that it allowed the robot to read in new numbers more efficiently, as well as removing delay commands throughout the program and simply putting a single delay (See Fig. 13) in the main loop that would cause the program to wait a second before executing the code each time.

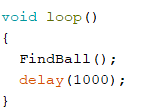


Fig. 13 – Single delay block

## Final Testing

With the code now working and running correctly in theory, it became time to finally test the robot in the arena. The robot was placed in the corner furthest from the goal, with 1 ball of each colour in the arena. The robot was turned on and when the code began running, the robot rotated 30 degrees until the red ball was centralized then it moved forward to the ball. Unfortunately, the robot didn’t stop when it reached the ball and instead continued until it reached the wall, at which point it paused and turned. The robot was retrieved and modified the values for the width of the ball necessary to stop the robot. The test was rerun, and it was able to move up the ball and stop, but just a bit too late, therefore the grabber had knocked the ball off its stand and caused it to roll. To correct this, the delay was reduced between each time the code runs from one second to once every half second. This allowed the robot to react much more efficiently to changes in its environment and therefore be much more responsive to its own actions. When this newest update was run (See Appendix 2), the robot successfully maneuverer to the ball, stopped, and closed the grabber around the ball. It then turned and found the goal before driving over and depositing the ball. (See Fig. 14)

Fig. 14 Robot in arena after completing a test

After this initial success, it was decided to scale up for a full test and ran the test again, but with the full nine balls this time; three of each colour. The robot started in the same position again and when the robot was turned on, it seemed initially confused by there being two red balls visible within its view, but after it moved forward, one of the balls was no longer visible, and therefore it was able to centralise on the remaining red ball and drive towards it correctly. The robot approached the ball and picked it up correctly before navigating its way to the end zone and only knocked one blue ball over as it drove by. It dropped the red ball in the goal, reversed, and turned before identifying the second ball which it recovered without problems. The robot seemed to have a few difficulties in finding the third and final ball as it was partially hidden behind a green ball but after rotating twice, it was able to pick up enough of a colour signature to be able to identify the final ball, which it then went and picked up before bringing to the goal.

# Conclusion and Future Work

The purpose of undertaking this project was to teach problem solving skills, group-work, presentation skills and team co-ordination. These outcomes were all achieved during the project and the team feels pleased with the work and effort that was put in.

The project incorporated independent research on many topics, specifically the PixyCam camera as there was almost no source material given. As a result, all data or information had to be obtained through research. From its fundamental basic operation, to its complex functions necessary for the project, all aspects had to be researched, studied, and learnt. The same can be said for the research and understanding of the Arduino code and the MakeBlock system, although a basic knowledge of coding was already achieved through previous courses in JAVA. The use of direct input form sensors as parameters in code, the use of programming libraries and the delay function were all new concepts that the team had no experience of prior to the project.

Collaboration amongst the team was well exercised in all aspects of the project especially in terms of design and work presentation. For the purposes of creating the best design, but also to ensure that each member's concept and ideas were used, the team chose an amalgamation of each individual members designs. This both allowed for the most efficient design, as it took the major benefits ad features for each design, but it also allowed each member of the team to witness their input in the final design. Which created a sense of equality and investment for all members in the final design. The project also incorporated team work for each of the presentations of the project as they required preparation in layout, content, and the execution of the actual presentation. For this the team delegated specific areas amongst each member for the presentation then combined each individual section for the total project. This was especially evident in the preparation for the interim presentation as there was a specified time limit on the length of the presentation. Each member practised their section of the speech with the PowerPoint slides they had prepared while the other two members critiqued and edited their presentation. This preparation and rehearsal allowed for a very cohesive final project that met the designated time limit and prevented an overlap in terms of content presented.

The removal of the line follower component is an example of how the team dealt with a problem in the project. The design originally contained a line follower sensor for identification of the goal area. Once construction was underway however it became clear that this would not have been possible in the current design. The team debated and discussed this issue with each member suggesting different solutions and the remainder of the team pointing out flaws and benefits for each decision. The team was quite satisfied with the final product and very happy overall with the process of the project. Each member agrees that the methods that were used in the process would be brought forwards to future projects. The team agreed that the organisational approach used was imperative in the success of the project and agreed to replicate this again in future projects. The methodology for communication such as the messenger and OneDrive applications were unanimously praised also. However, it was noticed that in future projects, a more rigid editing structure could be implemented, as a few issues occurred when two members were editing the same document from the OneDrive at the same time. A suggestion was that in future members would timestamp each document with a title following the format of: “FileName/Date/Time”. This would prevent the issue of two members editing the same document at the exact same time unaware of the alternate changes to the document.

Each member noted that taking the amount of code required for their designs into account when initially drawing out their designs was very beneficial as this prevented them from including unnecessary components that would be superfluous to the core function. While this foresight was beneficial, it affected the final design as it is very conservative in terms of components. An alternate approach would be to not consider the code until it came to the stage of coding. This may lead to a finally more complex robot, with more sensors for input, but this it could have been more efficient in terms of speed, energy consumption and overall durability. The team approached the process from a ground up approach, i.e.: design-> construction-> programming-> testing. This was the most intuitive approach which gave each member who maybe lacked experience, the time necessary to gain the relevant experience in the process.

In summary the team felt that the overall project was a success, that the purpose of the module had been achieved in its learning outcomes. The objective of the robot had been achieved to the best of the team's capabilities and that it was able to successfully carry out the task at hand.

# References

|  |  |
| --- | --- |
| [1] | SteveO, “Stack Exchange - Robotics Beta,” 06 11 2016. [Online]. Available: https://robotics.stackexchange.com/questions/11108/what-is-the-difference-between-dc-motor-with-encoder-and-dc-with-out-encoder. [Accessed 27 04 2018]. |
| [2] | RENESAS, “BLDC Motor Basics - Understanding high efficiency motors,” 2018. [Online]. Available: https://www.renesas.com/en-sg/support/technical-resources/engineer-school/brushless-dc-motor-01-overview.html. [Accessed 27 04 2018]. |
| [3] | RENERAS, “BLDC Motor Basics,” 2018. [Online]. Available: https://www.renesas.com/en-sg/support/technical-resources/engineer-school/brushless-dc-motor-01-overview.html. [Accessed 27 04 2018]. |
| [4] | CMUcam5, “Pixy (CMUcam5): a fast, easy-to-use vision sensor,” 15 08 2013. [Online]. Available: https://www.kickstarter.com/projects/254449872/pixy-cmucam5-a-fast-easy-to-use-vision-sensor/updates. [Accessed 01 05 2018]. |
| [5] | Conrad, “Conrad,” 2018. [Online]. Available: https://www.conrad.pl/p/pixycam-cmucam-5-pixy-cam-pixy-cam-1364834. [Accessed 02 05 2018]. |

# Appendices

Appendix 1.  MBlock sample move code

#include <Arduino.h>

#include <Wire.h>

#include< SoftwareSerial.h>

#include <MeMegaPi.h>

//Encoder Motor

MeEncoderOnBoard Encoder\_1(SLOT1);

MeEncoderOnBoard Encoder\_2(SLOT2);

MeEncoderOnBoard Encoder\_3(SLOT3);

MeEncoderOnBoard Encoder\_4(SLOT4);

voidi sr\_process\_encoder1(void)

{

if(digitalRead(Encoder\_1.getPortB()) == 0){

Encoder\_1.pulsePosMinus();

}else{

Encoder\_1.pulsePosPlus();

}

}

voidi sr\_process\_encoder2(void)

{

if(digitalRead(Encoder\_2.getPortB()) == 0){

Encoder\_2.pulsePosMinus();

}else{

Encoder\_2.pulsePosPlus();

}

}

voidi sr\_process\_encoder3(void)

{

if(digitalRead(Encoder\_3.getPortB()) == 0){

Encoder\_3.pulsePosMinus();

}else{

Encoder\_3.pulsePosPlus();

}

}

voidi sr\_process\_encoder4(void)

{

if(digitalRead(Encoder\_4.getPortB()) == 0){

Encoder\_4.pulsePosMinus();

}else{

Encoder\_4.pulsePosPlus();

}

}

void move(int direction, int speed)

{

int leftSpeed = 0;

int rightSpeed = 0;

if(direction == 1){

leftSpeed = -speed;

rightSpeed = speed;

}else if(direction == 2){

leftSpeed = speed;

rightSpeed = -speed;

}else if(direction == 3){

leftSpeed = speed;

rightSpeed = speed;

}else if(direction == 4){

leftSpeed = -speed;

rightSpeed = -speed;

}

Encoder\_1.setTarPWM(rightSpeed);

Encoder\_2.setTarPWM(leftSpeed);

}

void moveDegrees(int direction,long degrees, int speed\_temp)

{

speed\_temp = abs(speed\_temp);

if(direction == 1)

{

Encoder\_1.move(degrees,(float)speed\_temp);

Encoder\_2.move(-degrees,(float)speed\_temp);

}

else if(direction == 2)

{

Encoder\_1.move(-degrees,(float)speed\_temp);

Encoder\_2.move(degrees,(float)speed\_temp);

}

else if(direction == 3)

{

Encoder\_1.move(degrees,(float)speed\_temp);

Encoder\_2.move(degrees,(float)speed\_temp);

}

else if(direction == 4)

{

Encoder\_1.move(-degrees,(float)speed\_temp);

Encoder\_2.move(-degrees,(float)speed\_temp);

}

}

double angle\_rad = PI/180.0;

double angle\_deg = 180.0/PI;

void setup(){

//Set Pwm 8KHz

TCCR1A = \_BV(WGM10);

TCCR1B = \_BV(CS11) | \_BV(WGM12);

TCCR2A = \_BV(WGM21) | \_BV(WGM20);

TCCR2B = \_BV(CS21);

attachInterrupt(Encoder\_1.getIntNum(), isr\_process\_encoder1,

RISING);

Encoder\_1.setPulse(8);

Encoder\_1.setRatio(46.67);

Encoder\_1.setPosPid(1.8,0,1.2);

Encoder\_1.setSpeedPid(0.18,0,0);

attachInterrupt(Encoder\_2.getIntNum(), isr\_process\_encoder2,

RISING);

Encoder\_2.setPulse(8);

Encoder\_2.setRatio(46.67);

Encoder\_2.setPosPid(1.8,0,1.2);

Encoder\_2.setSpeedPid(0.18,0,0);

Encoder\_1.move(1000,abs(180));

Encoder\_2.move(1000,abs(180));

}

void loop(){

\_loop();

}

void \_delay(float seconds){

long endTime = millis() + seconds \* 1000;

while(millis() < endTime)\_loop();

}

void \_loop(){

Encoder\_1.loop();

Encoder\_2.loop();

}

Appendix 2. Final code

#include <Arduino.h>

#include "MeMegaPi.h"

#include <Wire.h>

#include <SoftwareSerial.h>

#include <SPI.h>

#include <Pixy.h>

Pixy pixy; //create pixy global variable

MeMegaPiDCMotor motor1(PORT1B); //Left Motor

MeMegaPiDCMotor motor2(PORT2B); //Right Motor

MeMegaPiDCMotor motor3(PORT4B); //Grabber

uint8\_t motorSpeed = 100;

void setup()

{

Serial.begin(9600);

Serial.println("Starting...");

pixy.init(); //initialise pixymon

}

void loop()

{

openGrabber();

FindBall();

delay(1000);

}

void FindBall()

{

if ( Sig() == 3)

{

if ( xCoor() > 400)

{

Serial.print("ERROR: ");

Serial.println(xCoor());

}

if ( xCoor() > 165 && xCoor() < 175 )

{

Serial.println("Centralised");

Serial.println("Move Forward\_1");

moveFor();

//delay(500);

if ( width() > 70)

{

Serial.println("Close Grabber");

Stop();

closeGrabber();

goToGoal();

}

}

else if (xCoor() < 165)

{

moveRight();

Serial.println("Move Right");

//delay(500);

}

else if (xCoor() > 175 && xCoor() < 400)

{

moveLeft();

Serial.println("Move Left");

// delay(500);

}

}

else

{

Serial.println("Rotate Right");

moveRight();

//delay(500);

}

}

void goToGoal()

{

Serial.println("Go To Goal Code");

if ( Sig() == 4) //see black goal

{

if ( xCoor() > 165 && xCoor() < 175 )

{

Serial.println("Centralised");

Serial.println("Move Forward\_1");

moveFor();

if ( width() > 180)

{

Stop();

}

Serial.println("Open Grabber");

openGrabber();

Serial.println("Reverse and turn");

moveRev();

delay(500);

Stop();

moveLeft();

delay(500);

Stop();

Serial.println("Go to Find Ball code");

FindBall();

}

}

else if (xCoor() < 165)

{

moveRight();

Serial.println("Move Right");

//delay(500);

}

else if (xCoor() > 175 && xCoor() < 400)

{

moveLeft();

Serial.println("Move Left");

}

}

else

{

Serial.println("Rotate Right");

moveRight();

}

////////////////////////////////////////////////////

//////////////////////

//PixyMon setup

int xCoor()

{

delay(50);

uint16\_t blocks = pixy.getBlocks();

if (blocks)

{

return pixy.blocks[0].x;

}

}

int Sig()

{

delay(50);

uint16\_t blocks = pixy.getBlocks();

if (blocks)

{

return pixy.blocks[0].signature;

}

}

int width()

{

uint16\_t blocks = pixy.getBlocks();

if (blocks)

{

return pixy.blocks[0].width;

}

}

////////////////////////////////////////////////////

////////////////////////

//Motor setup

void moveFor()

{

motor1.run(motorSpeed);

motor2.run(-motorSpeed);

delay(1000);

motor1.stop();

motor2.stop();

}

void moveRev()

{

motor1.run(-motorSpeed);

motor2.run(motorSpeed);

delay(1000);

motor1.stop();

motor2.stop();

}

void moveRight()

{

motor1.run(motorSpeed);

motor2.run(motorSpeed);

delay(1000);

motor1.stop();

motor2.stop();

}

void moveLeft()

{

motor1.run(-motorSpeed);

motor2.run(-motorSpeed);

delay(1000);

motor1.stop();

motor2.stop();

}

void Stop()

{

motor1.stop();

motor2.stop();

}

void openGrabber()

{

motor3.run(motorSpeed);

delay(1000);

motor3.stop();

}

void closeGrabber()

{

motor3.run(-motorSpeed);

delay(1000);

motor3.stop();

}