ECS514 Design & Build Project for EE_Group 7

Syed Uzair Maghrabi

Zackaria Azam Kazmi

Raakin Musa Ali

Engin Hayrullah Ates

Zain Y Y M I Malallah

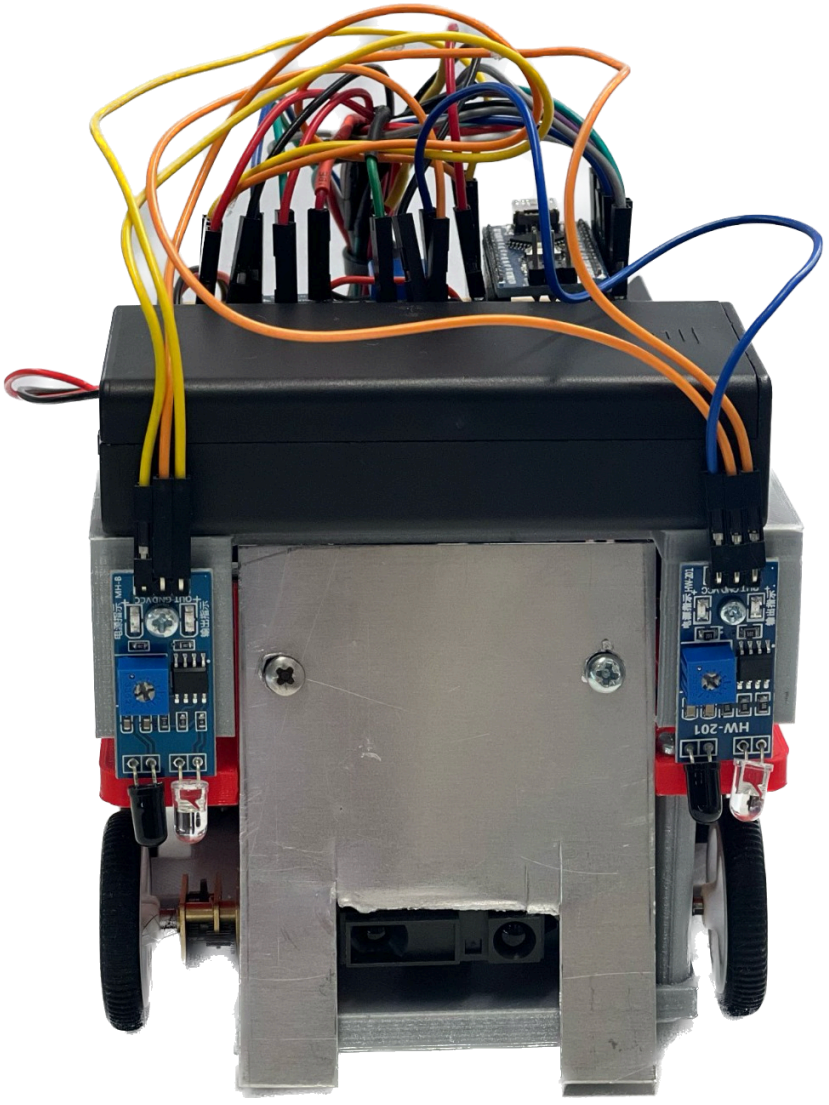

Table of contents

1. INTRODUCTION

The project aims to build an autonomous sumo robot to be developed for use in combat against other sumo robots.

The sumo robot will need to be able to:

- Manoeuvre around the dohyo autonomously.
- Be able to attack and defend in appropriate scenarios.
- Recognise and make manoeuvres against the white boundary line.
- Have a delay of 5 seconds before starting.

The robot will need to be autonomous, so a programmable microcontroller is required to make the calculations necessary, therefore a programming IDE and programming language need to be chosen. Various components will be utilised as inputs and send information to the microcontroller; depending on the information received the microcontroller will need to determine whether the robot is in an attack or defend scenario and appropriate instructions to the outputs will be sent to manoeuvre the robot. The 5-second delay will need to be added to the code manually.

2. SPECIFICATION

The Sumo robot will be competing in the mini sumo robot class, the rules for this class are specified below:

- Maximum mass of 0.5 kg
- Dimensions of 10 cm by 10 cm
- Must move within the dimensions of the dojo
- Must be able to attack and defend
- Must not cross white ring around the dojo
- Budget of £100
- Sumo circular ring will be for the mini category; diameter of 77 cm,
- Black background with 2.5 cm white border lines
- Starting (Shikiri) parallel lines of 10 cm in length and separation of 10 cm

The robot has a maximum weight of 500g and dimensions of 10cm x 10xcm. Therefore, the components we chose have to adhere to the specification. Various components will need to be considered such as the size of the battery, how many motors/wheels and how powerful and heavy they need to be. We want the robot to ultimately have a weight nearing the max weight of 500g as heavier objects have more inertia and therefore more force will be required to push us off the dohyo ($F = M \times A$).

The robot will need to be able to detect the outer white line of the dohyo and make defensive manoeuvres to guarantee the robot does not cross the white line. The colour white does not contain any pigment that absorbs light so the majority of light is reflected, we can use this scientific principle to determine whether the robot has crossed a white line or not. One component that uses this phenomenon is an IR sensor. The speed of the robot will need to be adjusted according to the clock speed of the microcontroller to make sure the robot senses the white line and conducts the necessary manoeuvres before falling off the dohyo.

The budget of the robot has a limit of £100. As a consequence, we need to decide which of our components can benefit the most from being more expensive than others. Components such as the microcontroller, motors and battery can benefit the most from being more expensive. We also must look into alternative methods of constructing the chassis of our robot and take into consideration the materials we should use to fit this quota.

The two robots only start 10 cm away from each other, so our robot will need to be able to accelerate rather quickly to ensure maximum pushing force is applied. This means the motors need to have a high rpm. Additionally, the motors need to have a high enough torque so if our robot and an enemy robot are pushing against each other, our robot will have the upper hand and be able to overcome the pushing force of the enemy robot. When the motors are at maximum torque the motors will be at a phase called “stall”. The battery will need to be able to provide the necessary “stall current” which is the maximum amount of current the motors can take, so we need to select a battery that can be able to provide the necessary current as well as keep our other components operational during this “stall” period. As a motor falls into the category of components of inductors, a magnetic field is induced when current flows through the motor, when the current stops or changes the magnetic field around the motor will collapse and an induced EMF will be created, this EMF is called the “back emf”, to protect our microcontroller from this back emf motor controllers or flyback diodes will be required.

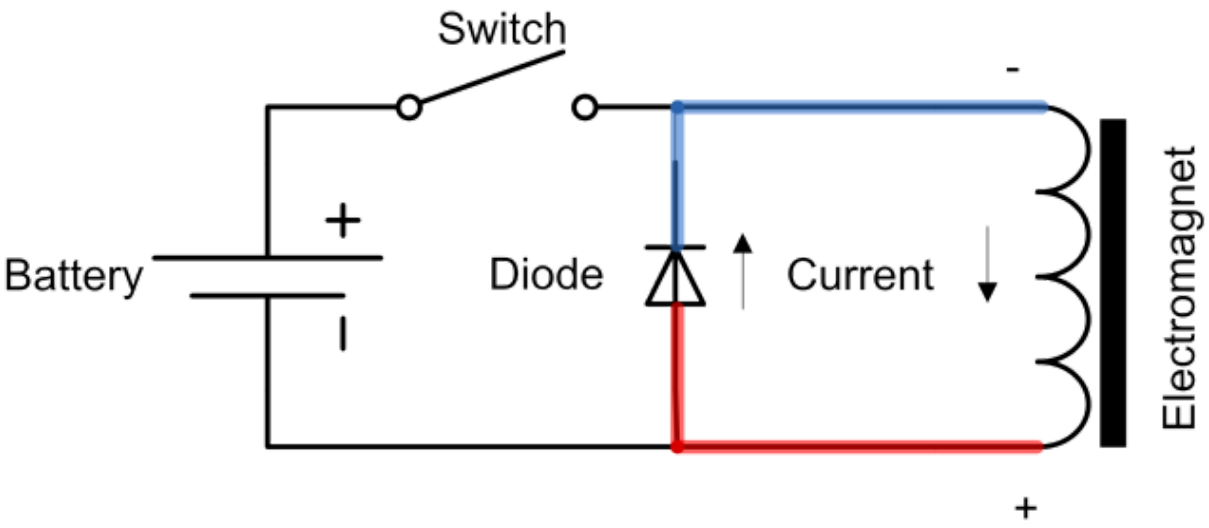

Figure 1: Example of back EMF prevention using flyback diode

The specification depicts the size of the dohyo as having a diameter of 77cm if we want to identify our opponent from every point of the dohyo the detection range needs to have a maximum range of 77cm. There is also a chance that the robot may be able to detect a spectator as an enemy when turning at the edge of the dohyo and falling off the dohyo, so the robot will either need to filter out the detection range or simply ignore the detector component entirely.

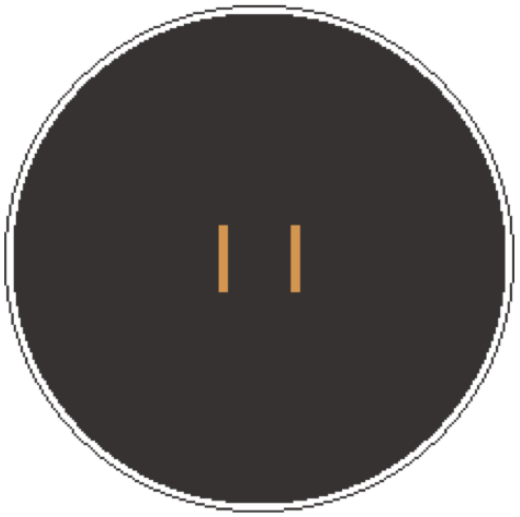

Figure 2: Example of a dohyo

3. DESIGN

Microcontroller

The obvious choice for microcontroller boards was the Arduino family. The Arduino is an open-source microcontroller capable of interfacing with multiple other components with multiple I/O ports. Due to its small compact size, we will be choosing the Arduino Nano for our project. Fortunately, due to the software being open source, many cheaper replicas are available on the market for much cheaper than their official price.

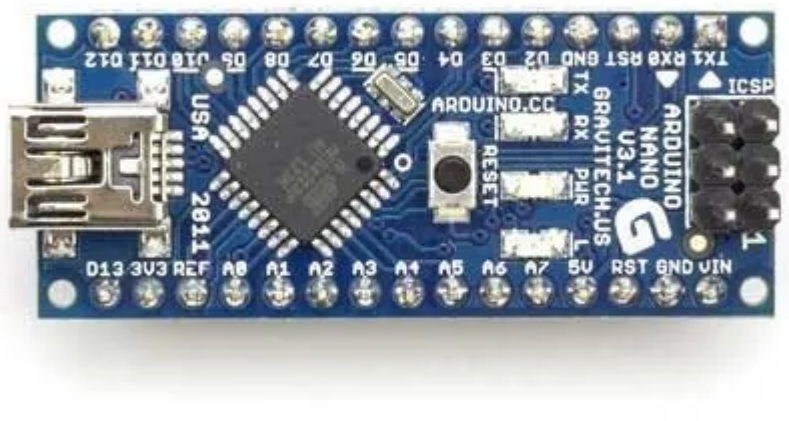

Figure 3: Arduino nano board

Despite its small size, the Arduino Nano offers the same connectivity and specifications as its larger counterpart the Arduino UNO. The Arduino Nano can be powered in a multitude of ways, one way is through the micro-USB port which is also used for communication with other devices. Another way of powering the Arduino Nano is by supplying a regulated 5V output into the 5V pin header on the Arduino Nano. The downside of this method is that either we have to limit ourselves to a 5V battery, which would considerably limit the performance of our motors, or regulate the external power supply ourselves which would reduce the efficiency of our device. The more conventional way of powering the board is by using an unregulated 6-20V external power through the Vin pin which is the method we would be choosing..

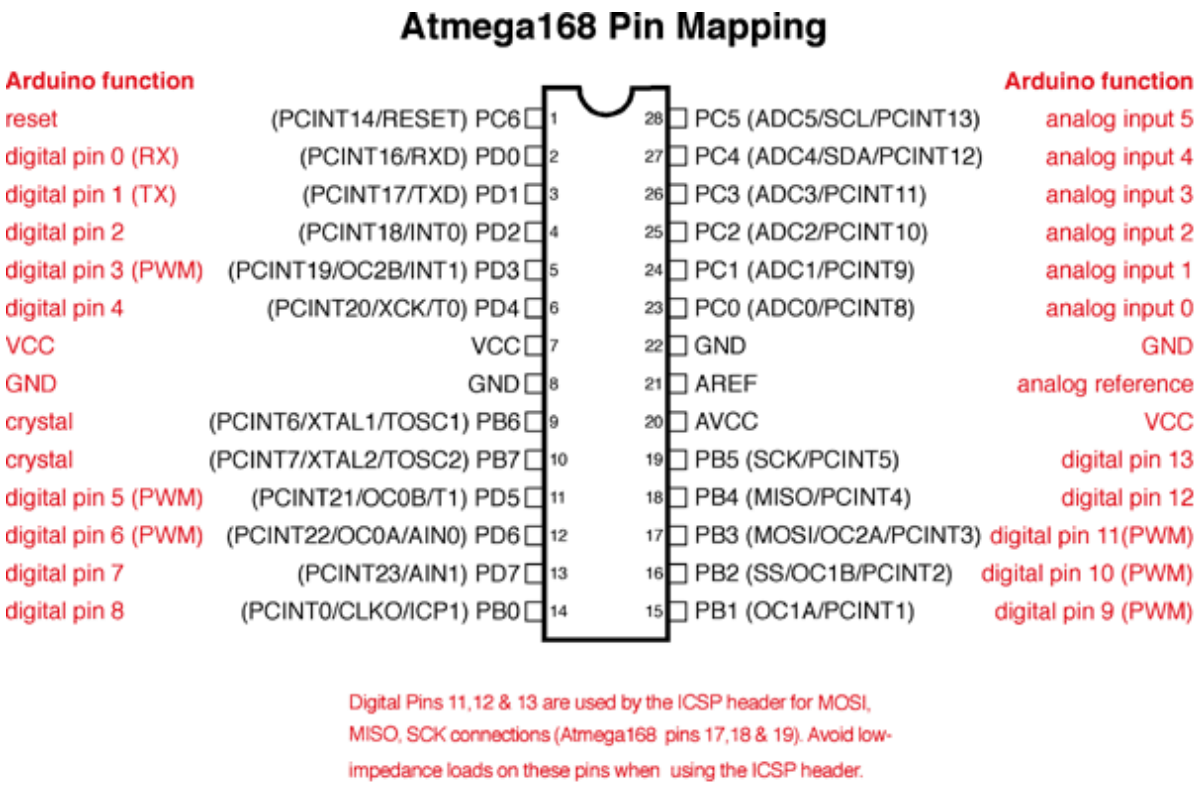

Figure 4: Mapping of Arduino pins to Atmega168

The ATmega168 contains 16KB of flash memory which is used for the bootloader and the memory for storing code, this should be enough to store our sumo robot script.

To communicate with another computer the ATmega168 can provide UART (Universal Asynchronous Receiver/Transmitter) TTL (5V) communication which uses the digital pins 0 (RX) and 1 (TX) and is as simple to connect as connecting the opposing transmit pin to receive pin and vice-cera, this is done through the micro-USB cable. UART sends and receives data by converting the data into bits and then splits them into packets for transmission. Each packet contains a bit to start, 5 to 9 bits for data, a parity bit and 1 to 2 stop bits. After preparing the packet it is sent out via the TX pin and received through the RX pin.

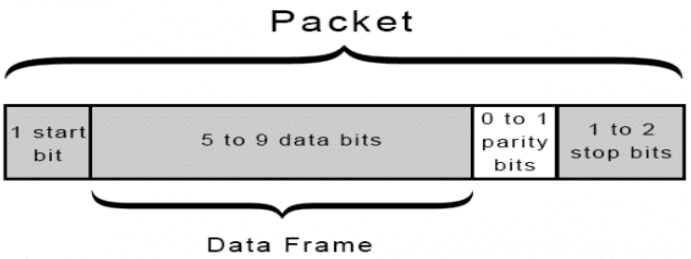

Figure 5: UART communication packet

To program our Arduino we need to use the Arduino IDE. The Arduino IDE is where Arduino programs are written and then uploaded to the Arduino board, the Arduino IDE uses a language similar to C++ and uses a variety of libraries which is used to interface with components and communication methods such as I2C, which we will use later.

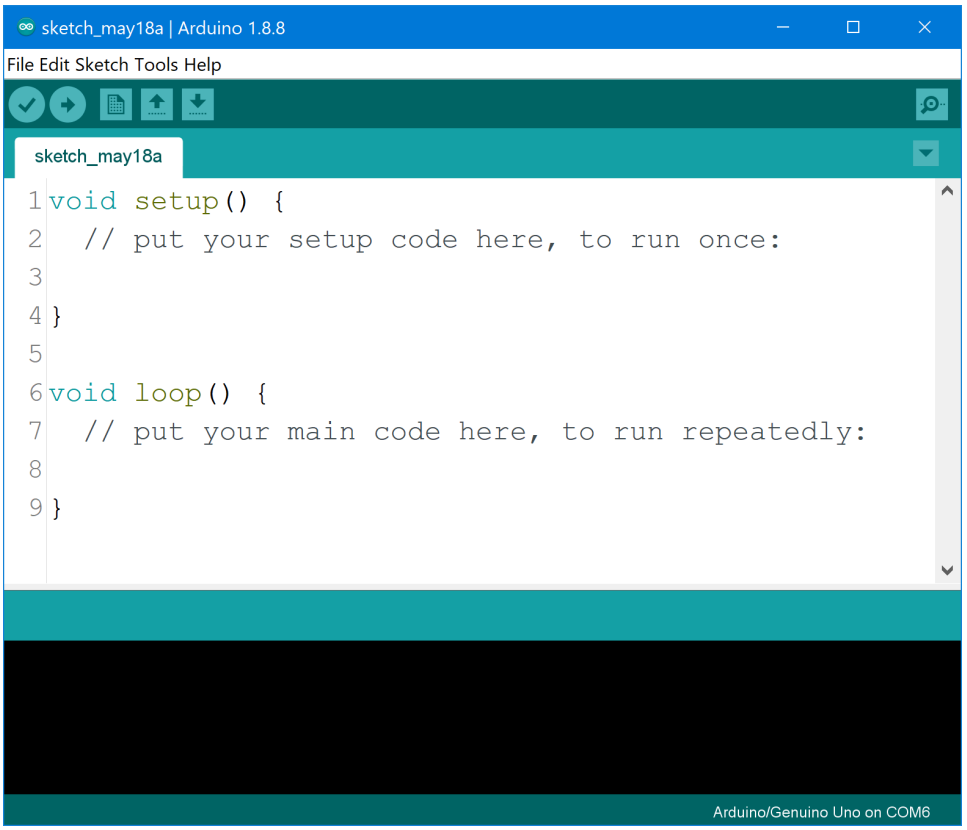

Figure 6: the Arduino IDE

In the Arduino IDE (located above) we can see there are two main functions: *“void setup()”* and void loop(). The setup() function only runs once and is used to initialise your components and set up serial communication with the Arduino and the serial monitor. The void loop() is an area of code which runs indefinitely and where the majority of our robot sumo code is entered. To add a library you must write the correct #include statement above the setup() function. Global variables are variables that don't change like the pin bindings are also usually located above the setup() function.

14 digital I/O pins on the Nano can be used for input and output which operate at 5V. As they are digital pins they can either write a HIGH or LOW signal usingdigitalWrite() or read a HIGH or LOW signal using digitalRead(). Whether they are input or output pins can be initialised using the pinMode() function. 6 of these digital IO pins (3, 5, 6, 9, 10 and 11) have PWM output capabilities.

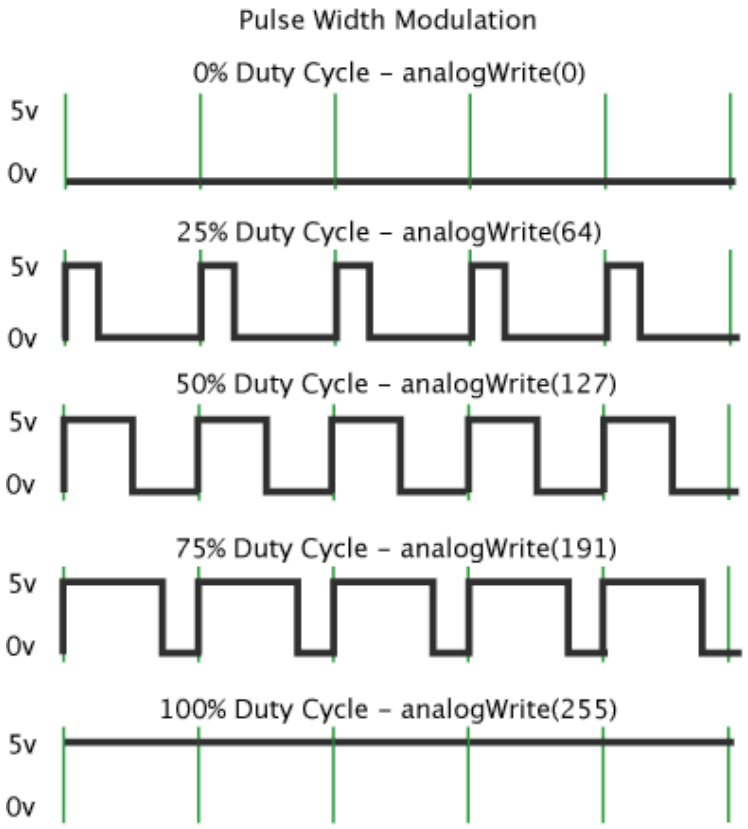

Figure 7: Example of PWM output using analogWrite()

Pulse width modulation is a technique for getting an analogue output using digital square waves. By switching on and off a digital signal we can replicate voltages between the max Vcc of the board (for the Nano 5V) and 0V by changing the ratio of time the signal stays on to the time it stays off. The duration of time the signal stays on is called the pulse width. To change the pulse width you use the function analogWrite(). analogWrite() uses a scale of 0 - 255, where analogWrite(255) creates a signal with a 100% duty cycle and analogWrite(127) creates a 50% duty cycle.

The ATmega168 also has 6 analog pins which are capable of reading analog signals which will be useful for interfacing with more precise components allowing us to take more accurate measurements. The analog pins A4 and A5 are used in IC2 communication which we will later use to communicate with our components.

Motors

For motors, we decided on the **Pimoroni 50:1 Micro Metal Gear Motor**

Figure 8: Pimoroni 50:1 Micro Metal Gear Motor

The motors' small size and weight (9.6g) allow us to use 4 motors in our sumo robot. The motor has a 50:1 gear ratio which means for every rotation of the motor's input shaft, the output shaft will rotate 50 times. This gear ratio is significant because it allows the motor to produce more torque but at the expense of speed.

The voltage mentioned in the technical specification is 6V and the max torque is 3.92Ncm this means that one motor can produce enough rotational force of 3.92N acting over a distance of 1 cm, as we are planning to use 4 of them then the total torque of 15.68Ncm. At max torque, the RMP is at a maximum of 215rmp. Therefore at max torque, the total horsepower of my sumo robot is:

$$\text{Horsepower} = \frac{Torque\ x\ RPM}{5252}$$

$$= \frac{0.1568\ x\ 215}{5252} = 0.0064\text{hp}$$

At max torque (stall) the maximum current draw is 770mA so for all 4 motors the maximum current out battery should supply is 3080mA. The stall torque is 0.5kg cm. Stall torque is the maximum torque applied to stop the motor's movement. Combined stall torque is 2 kg cm.

Power supply

There are a multitude of different ways to power our robot but we have to select the option that fits within our specifications.

Option #1 9V battery

Figure 9: 9V DC battery

9V DC batteries are very common in Arduino projects, as it's 9V we use high-powered motors as well as the unregulated 6V-20V Vin pin on the Arduino board. However 9V batteries generally are rated at 1 ampere hour meaning the battery can supply 1 Amp of current before running out and even less if our motors. As previously mentioned earlier at max load the current load at a minimum is approximately 3080mA therefore the battery life of the robot will be:

Battery Life (Hours) = $\frac{Battery\ Capacity\ (Ah)}{Current\ Draw\ (A)}$

= $\frac{1}{3.08}$ = 0.325 hours

0.325 hours is approximately 20 minutes, for testing purposes, 20 minutes is not a lot of time before the battery of our robot is depleted so the battery must be replaced constantly which isn't very efficient or sustainable.

Option #2 AA batteries

Figure 10: 1.5V AA battery

AA batteries are also another option to power our robot. One advantage of using AA batteries is if you need more voltage you can wire them in series and if you want more current then you can wire them in parallel. Each AA battery has a voltage of 1.5V so if we decide to power our Arduino using our Vin pin we need at least 4 AA batteries wired in series to generate a voltage of 6V. When wiring AA batteries in series the charge remains the same as if it was one battery, the typical charge in an AA battery is 2Ah therefore the estimated battery life of our robot will be.

Battery Life (Hours) = $\frac{Battery\ Capacity\ (Ah)}{Current\ Draw\ (A)}$

= $\frac{2}{3.08}$ = 0.65 hours

If you wanted to double the battery life you would need 8 AA batteries 4 in series and 4 in parallel. The downsides of the AA battery are similar to the downsides of the 9V DC battery. Having to change a minimum of 4 batteries each time the battery depletes is also not very efficient as well as not being very sustainable. In addition, the batteries will take up a large majority of space on the robot, contributing to the overall bulkiness of the design.

Option #3 Lithium ion batteries

Figure 11: Li-ion Battery

A lithium-ion battery is a type of rechargeable battery that uses the reversible intercalation of positive lithium ions into electrically conducting solids to store energy. In comparison with other types of rechargeable batteries, lithium batteries are characterised by higher specific energy, higher energy density, efficiency and a longer cycle life.

Unlike the other two options listed above, lithium-ion batteries can be recharged after their charge is depleted. During the charging process an external power supply, like a laptop or computer, applies an over-voltage to the cell (a voltage greater than the cell's voltage so if you're charging a 3.7V Li-OH you can use a 5V power supply as an over-voltage). This forces electrons to flow from the positive to the negative electron. The lithium ions travel through the electrolyte from the positive to the negative electrode where they embed themselves into the porous electrode material in a process called intercalation. A charger for the lithium-ion batteries will be bought separately

Figure 12: 18650 charger

For our purposes we will be using the 18650 Lithium ion batteries, they carry a voltage of 3.7V so we will use 2 in series to maximise the potential of our motors as well as use the Vin port on the microcontroller. The capacity of each 18650 battery can vary and depends on the manufacturer, in our instance we will be using a capacity of 6800mAh.

Battery Life (hours) = $\frac{Battery\ Capacity\ (Ah)}{Current\ Draw\ (A)}$

= $\frac{6.8}{3.08}$ 3.08 = 2.21 hours

Out of all the battery choices the 18650 has the longest battery life and unlike the AA batteries won't be as bulky, so for our robot, we will be using the 18650 batteries.

To store our batteries, we will also purchase a 2 x 18650 battery holder with a switch.

Figure 13: 2x 18650 Battery Holder

IR sensor

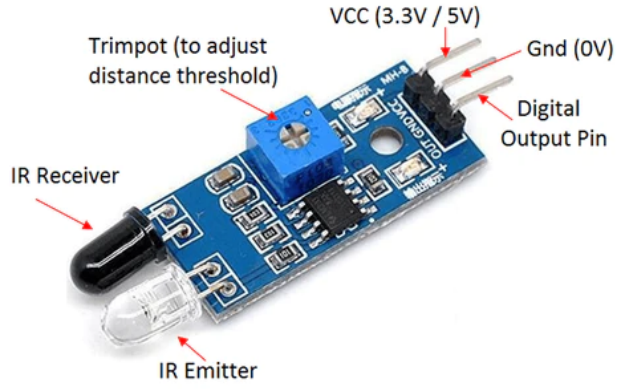

Figure 14: IR sensor

To check and determine the white boundary line there really is only one component suitable for the purpose, the IR sensor. An IR sensor is a component that measures the infrared radiation in the environment and when it detects any it sends an electrical signal as an output. The main benefits of an IR sensor are its low power usage, simple design and convenient features.

There are two types of IR sensors:

1. Active IR sensor.
2. Passive IR sensor

In this case, we will be using an Active IR sensor, there are two main parts to an Active IR sensor: a light-emitting diode as an Infrared Radiation transmitter and a Photodiode as an Infrared Radiation detector. The Infrared light from the LED bounces off objects and back to the Photodiode and a signal is sent informing the microcontroller that the IR sensor has detected an object. Using the scientific phenomenon mentioned earlier we can detect the white boundary line as the colour light is a much more reflective colour than black. So when the robot comes near the boundary line, the IR sensor detects the white line and an output signal is sent to the microcontroller where the motors are controlled to move away from the white line.

A potentiometer is added on the top of the IR sensor to determine the sensitivity of the IR sensor. By turning the potentiometer knob, you can increase or decrease the sensitivity of the sensor to better suit the requirements of your application.

L298N Motor Driver

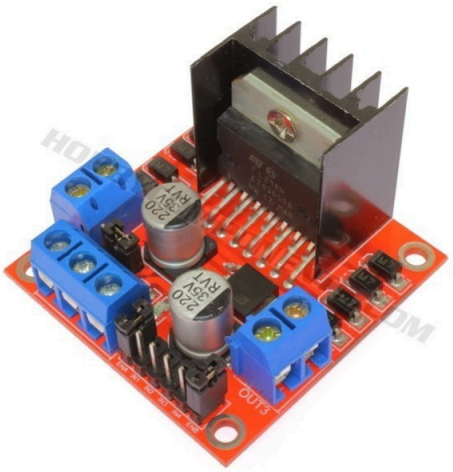

Figure 15: L298N H-Bridge Motor Driver

The L298N motor driver can control the speed and direction of 2 motors simultaneously, so to control all four of our motors we need 2 separate modules.

We can control the speed of the Motors using the PWM function to vary the speed of our motors. We have to remember to use the PWM-applicable output pins on the Arduino Nano. The low-powered PWM signals switch on and off the gate at the MOSFET through which the higher-power motor is driven.

The direction of the motor is determined by a H-bridge circuit. A H-Bridge circuit consists of four switching elements such as transistors or MOSFETs, with the motor in the middle of the circuit forming a ‘H’ like configuration. Depending on the two switching elements activated we can change the direction of the current flow and thus change the rotation direction of the motor

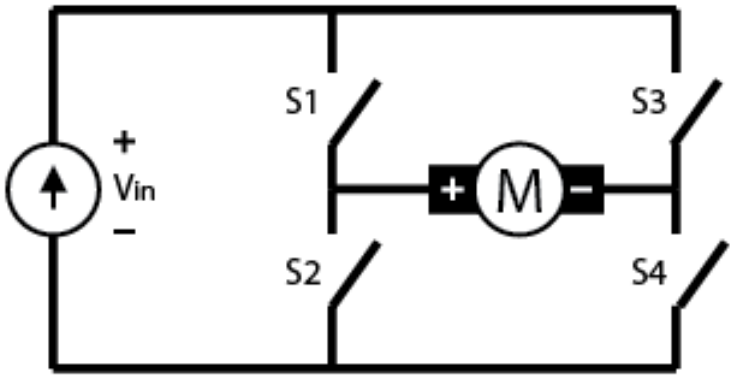

Figure 16: Example of a H-Bridge circuit

GY-521 MPU-6050 3 Axis Gyroscope

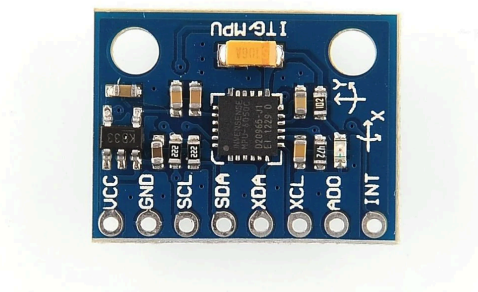

Figure 17: GY-521 MPU-6050

To make our robot as precise and accurate as possible we will include the GY-512 MPU-6050 Gyroscope in our robot. It measures rotational velocity or rate of change of the angular position over time, along the X, Y and Z axis. To calculate these positions the GY-521 uses MEMS technology and the Coriolis effect for measuring.

The Coriolis effect is a phenomenon observed in several rotating systems such as the Earth and the Moon. It causes objects to appear to deviate from straight paths when observed from a rotating reference frame. This apparent deflection is due to the rotation of the reference frame itself. For example, winds present in the northern hemisphere appear to curve to the right, while in the Southern hemisphere, they appear to curve to the left.

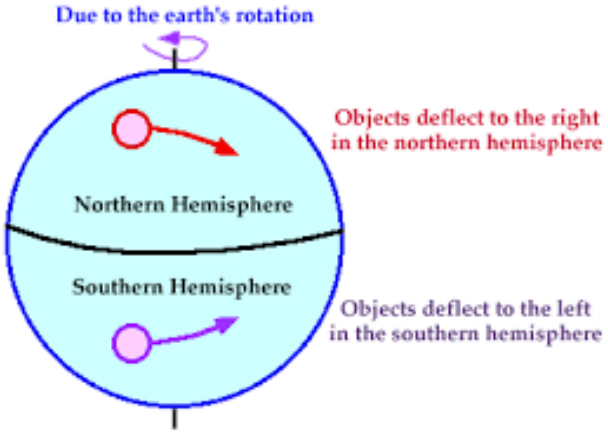

Figure 18: Coriolis effect in action

A mass is moving in a specific direction with a specific velocity and if an external angular rate is applied (shown by the red arrow) a force perpendicular to the velocity is applied (shown by the yellow arrow). This causes a perpendicular displacement of the mass, this displacement can be measured using a capacitor. The change in capacitance can be measured and processed and will correspond to a particular angular rate.

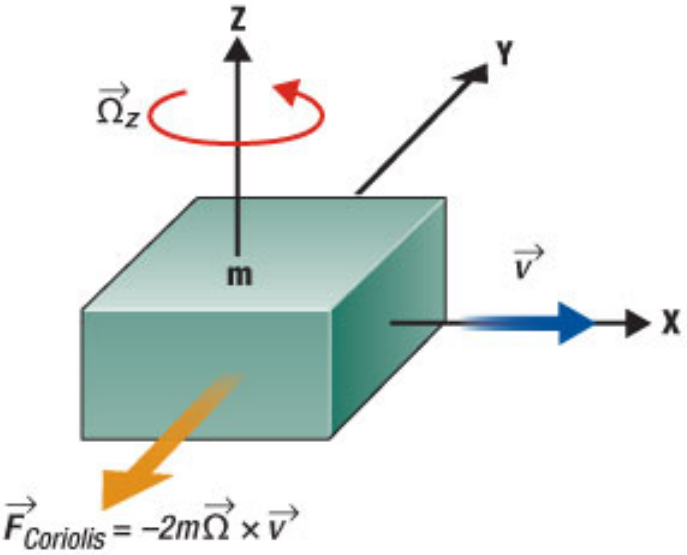

Figure 19: The Coriolis effect on a moving mass

To communicate with the Arduino the GY-521 uses I2C communication to interact with the Arduino Nano. I2C is a low-speed serial communication protocol available on all Arduino boards.

The I2C network consists of a master (Arduino) and a slave (GY-521) device, the master and slave devices are connected by two buses. The two buses are called SDA and SCL. The SDA bus is used for sending data back and forth between the slave and master devices, while the SCL bus is used to carry the clock signal used for communication timing. We can use multiple slave devices in conjunction with a single master device as each slave device has a specific I2C address that is used to identify the device. The Arduino has dedicated pins for I2C communication, on the Arduino Nano these are pins A4 and A5. Pin A4 is the SDA pin, and pin A5 is the SCL pin.

Sharp GP2T0A21YKOF

As per our specifications, we need to detect objects from a maximum of 77 cm away so in this case we will use the Sharp GP2T0A21YKOF Optical Sensor.

The Optical sensor works very similarly to the IR sensor but measures the time it takes for the IR signal to be transmitted by the transmitter reflected by an object and then received by the receiver. This time is multiplied by the speed of light (3x10^8) to calculate the distance between the objects. The output is an analog signal, so we must connect the sensor to an analog pin which is capable of analog to digital conversion (ADC). As the ADC can produce a signal range from 0-255 further calculations must be done to convert the signal to its distance. Luckily for us the library *#include "SharpIR.h"* does this job for us. Optical sensors are very accurate and the one we are choosing has a maximum range of 80cm which fits our specifications.

Figure 20: Soldered on circuit

3.1 . CIRCUIT DESIGN

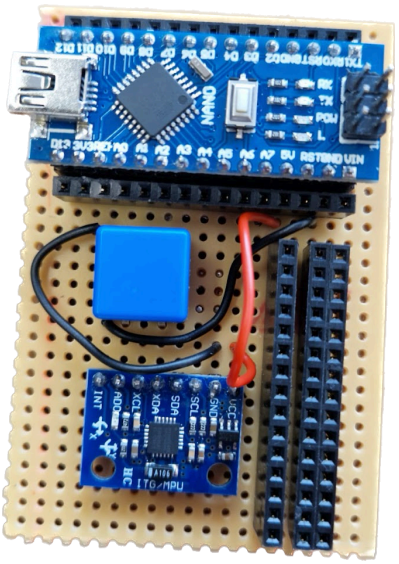

Figure 21: Soldered on circuit

Firstly, we measured and mapped out all our connections and cut some perfboard to size. On perfbarods all the columns are electrically connected making it rather simple to connect two different points

We soldered on any solderable components onto some perfboard noting any connections we had to make. As the microcontroller was the most expensive and vital part of the circuit ,we didn't think it would be suitable to solder it directly to the circuit. Instead we soldered on female pin headers that mirrored the pins on the Arduino nano. We then soldered on female pin headers on each side of the Arduino so we could unplug and plug in components which became very useful when deciding which pins to use. As mentioned by the specification the robot has to wait 5 seconds before activating, so we soldered in a button to initiate the 5-second delay.

More female pin headers were soldered in 3 rows, the row on the left is the unregulated voltage from the 7.4 V batteries for the L289N motor drivers, it's also connected to the Vin port providing power to the Arduino. The middle row is the ground row, which is connected to the ground pin on the Arduino and is used to connect all the components to the ground. This is vital as the Arduino needs a common ground between all the components to work. The row on the left is the 5V row and is connected to the 5V pin on the Arduino, it uses the voltage on the Vin and regulates the voltage to 5V. This row is used to power most of the low-powered sensors and logic for the H-Bridge.

The GY-512 gyroscope works best when it's stationary, so we decided to solder it down to the perfboard. A red wire is routed from the 5V rail to the 5V power pin on the GY-512 and a black wire connects the ground pin to the ground rail. As previously mentioned the SCL pin is connected to the A4 pin on the Arduino and the SDA pin is connected to the A5 pin on the Arduino nano header pin.

A Stripboard track cutter was also used to disrupt the preexisting column connection between two points. For the gyroscope, the track cutter was used to disrupt the connections above the 5V and GND pins as they were connected to incorrect pins on the Arduino nano.

Figure 22: Stripboard Track Cutter

3.2 . CHASSIS DESIGN

For our chassis, we decided on 3 separate platforms where multiple components were to be mounted. There are multiple different ways to manufacture these platforms, but we chose CAD design and 3D printing.

3D printing allows the creation of highly intricate and detailed geometries that may be difficult to achieve with other manufacturing methods like laser cutting. In addition, 3D printing is well suited for rapid prototyping, 3D printing is also very cheap which can help save money if we decide to print many prototypes. Small and intricate parts like brackets can be easily designed and printed in 3D CAD software such as Fusion 360.

We used Fusion 360 to design and model each of our platforms and used the slicing software UltimakerCura to create the STL files which are loaded onto a microSD card and finally loaded into a 3D printer.

The material we used is Polylactic Acid, more commonly known as PLA, which is a thermoplastic monomer derived from renewable sources such as sugar cane or corn starch. It’s a popular choice for 3D printing due to its ease of use and less likelihood of warping while printing. PLA prints can easily be drilled and sanded after printing meaning the material is adaptable to any changes you decide on for the part.

Bottom Platform

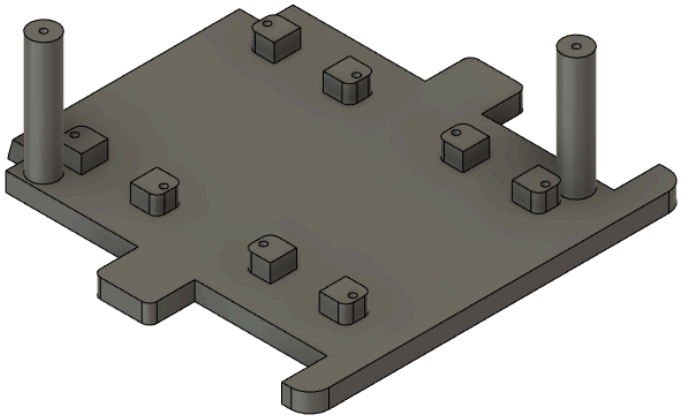

Figure 23: Lower platform CAD file

To compensate for the motors, there are 4 specific slots for the motor to slide into with accompanying mounting holes for M2 screws. The front of the plate is slanted at an angle so an aluminium plate, acting as our scraper, can rest easily along the front of the robot. Two pillars are also printed, they serve as standoffs where the middle platform can be mounted using M2 screws.

Our motors mustn’t wobble or move around while the robot is functioning as this can make the robot move erratically. Therefore we designed and 3D printed 4 custom brackets that clamp the motors down to ensure they don't wobble during operation. Using vernier callipers we measured the width of the motors to guarantee they were the correct fit

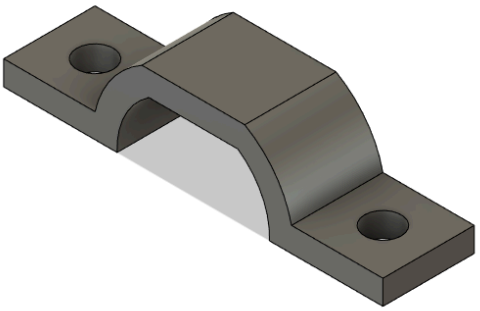

Figure 24: Motor Clamp

Middle Platform

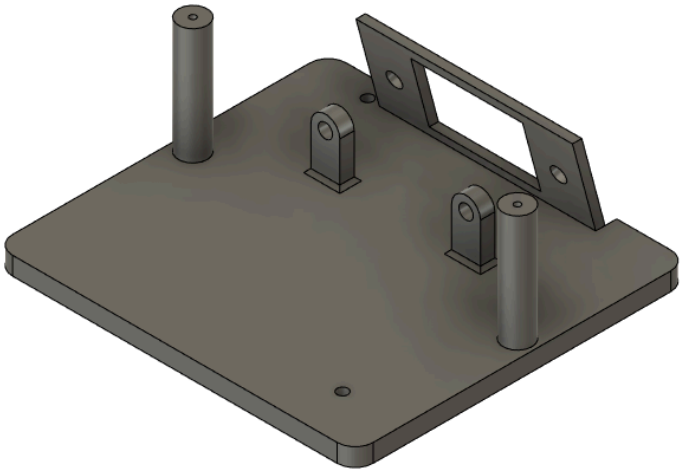

Figure 25: Middle Platform

The two L298N motor drivers are fixed on the Middle Platform using a hot glue gun. There are two holes on the bottom of the plate mirroring the two pillars on the bottom Platform to screw it in place. There are two mounting pillars for the optical IR sensor to be attached using nuts and bolts. The middle platform also includes a rectangular slant with two holes and another rectangle cut out of it, the two holes are used to screw down to an aluminium plate and a rectangle is cut so the Optical IR sensor has an unobstructed view. Just like the bottom platform, two pillars were also printed to act as a mounting point for the top platform.

Top Platform

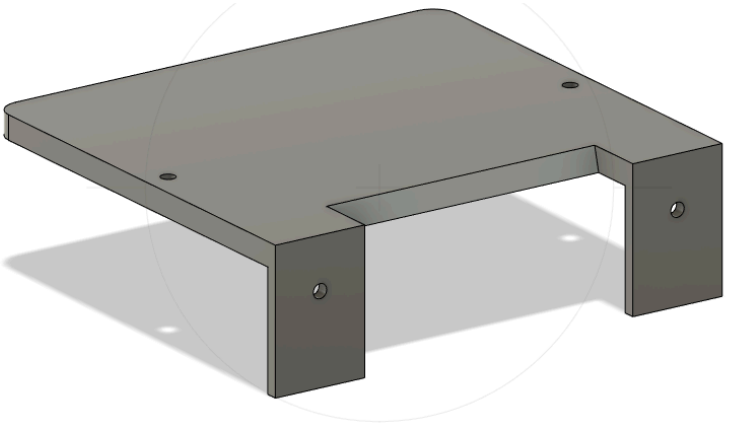

Figure 26: Top platform

The battery pack and circuit are mounted onto the top platform. The 18650-battery holder is attached using hot glue and the perfboard circuit can be slid onto rails and stuck on the platform using double-sided tape. Two holes are included to screw down the middle platform. At the front of the platform there are two overhangs, this is where the two IR detectors are attached, looking down at the doyo surface. The reason why the overhangs are at the front of the robot is that when an IR signal detects the white boundary line it gives the microcontroller more time to send instructions to the motors to counter the movement due to its position.

3.3 Design Improvements

Cable management

A few improvements were made to the chassis to ensure our wires were protected and kept organised.

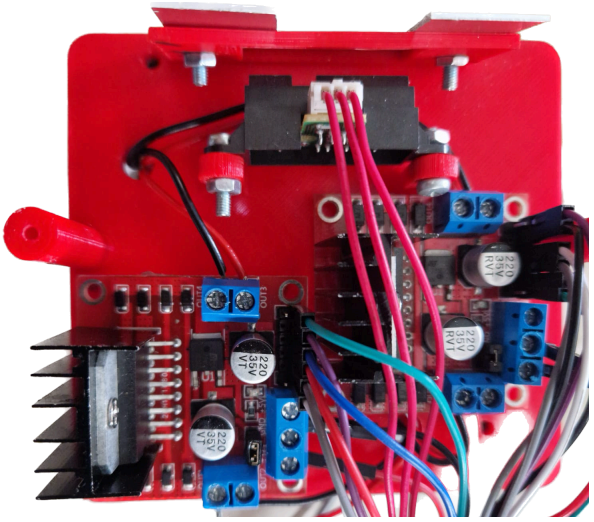

Figure 27: Completed Middle Platform

Figure 27 shows the middle platform of the sumo robot, alterations were made to ensure cables were safe during operation. The middle platform has 2 holes (1 is visible on the top left) drilled into it, to allow the motor wires to be fed into the motor drivers without leaving wires exposed on the exterior of the chassis. Leaving wires dangling on the exterior is prone to collision damage, which could lead to vital connections being damaged, therefore it is recommended to make this amendment.

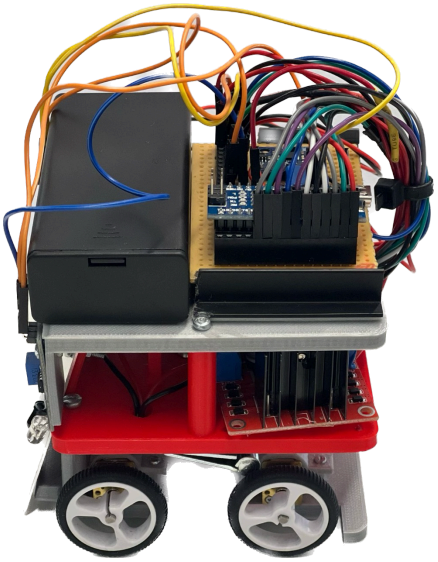

Figure 28: Side View of Sumo robot

The sumo robot consists of many connections between the middle and the top platform. To keep the wires compact and organised the wires were zip-tied. This keeps all the wires together rather than having cables all around the exterior.

Cables were also colour-coded, in this model all the wires from 1 of our H bridges were labelled with black tape on the ends of the wire, the other H bridge was left without the tape. All grounds were connected using black wires and the live wires were connected using red wires. This adjustment allows for easy debugging, for instance, when a wire disconnects the labels and colour coding allow the user to easily identify the disconnected wire and place it back into the correct place.

Increasing weight

The model weighed around 370 grams, to ensure our robot was as close to the weight limit as possible, extra mass was added.

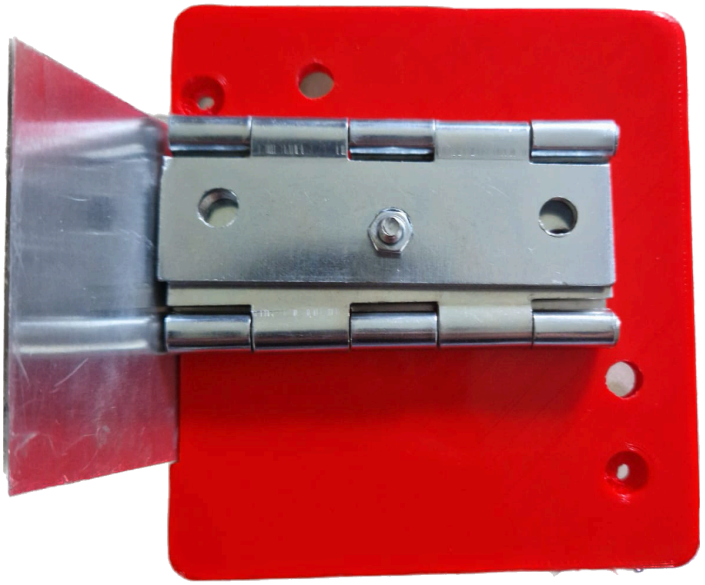

Figure 29: Bottom View of the middle platform

We decided to add a door hinge, which added an extra 100 grams leaving the final mass at 470 grams. The door hinge was screwed to the bottom of the middle platform, it is easily removable.

The placement of the door hinge is relatively low, this is to lower the centre of gravity. Having a lower centre of gravity provides greater stability. A higher centre of gravity increases the likelihood of the robot toppling from a push.

The benefits of increasing weight are that the opponents require a larger force to push the robot effectively. The added mass also improves the traction by increasing the downward force on the tyres, as a result of this the friction between the tyres and the surface increases. This prevents wheel spin and loss of control when the car is braking or accelerating

Repositioning of the optical sensor

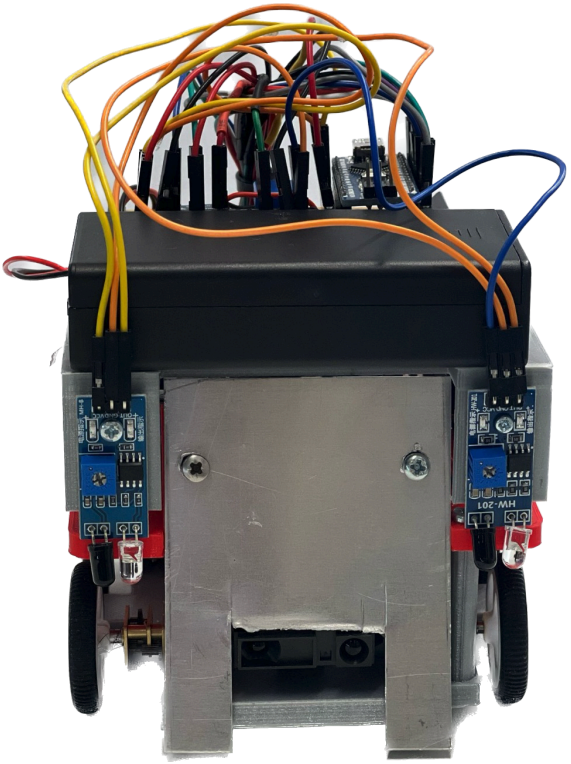

Figure 29: Front View of the Sumo robot

The optical sensor was located previously on the second platform which resulted in it being elevated at a height of 60 mm. However, this height is too excessive and can be disadvantageous as it is common for Mini sumo robots to be less than 60mm tall. This results in the optical sensor not being able to detect smaller sumo robots.

To combat this issue the optical sensor was moved to the bottom platform. This required us to drill a hole in the bottom platform to allow placement for an L-shaped Right Angle Support Bracket which the sensor connects to. This left our sensor at a height between 10mm to 20mm this height ensures that robots of smaller heights can be effectively detected.

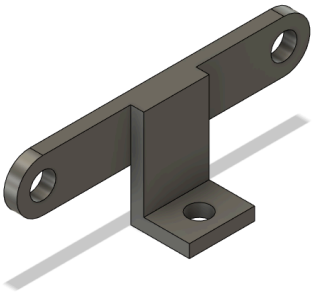

Figure: 30 L-Shaped Support Bracket

4. Analysis

As per the specification the robot will need to be able to attack and defend depending on information gathered by the sensors. This can be done by writing a program using the Arduino IDE Software

```
const int Left_IR_SENSOR = 14;
const int Right_IR_SENSOR = 15;
const int push_button = 16;

volatile int flag = 0;

#include "SharpIR.h"

#define IRPin A7
#define model 1080

int distance_cm = 50;

SharpIR mySensor = SharpIR(IRPin, model);
#include <Wire.h>
const int MPU = 0x68; // MPU6050 I2C address
float AccX, AccY, AccZ;
float GyroX, GyroY, GyroZ;
float accAngleX, accAngleY, gyroAngleX, gyroAngleY, gyroAngleZ;
float roll, pitch, yaw;
float AccErrorX, AccErrorY, GyroErrorX, GyroErrorY, GyroErrorZ;
float elapsedTime, currentTime, previousTime;
int c = 0;

const int FR_EN = 3;
const int FR_in1 = 5;
const int FR_in2 = 4;

const int FL_EN = 6;
const int FL_in1 = 8;
const int FL_in2 = 7;

const int BL_EN = 11;
const int BL_in1 = 2;
const int BL_in2 = 13;

const int BR_EN = 9;
const int BR_in1 = 12;
const int BR_in2 = 10;
int i = 100;
int j = 100;
```

Figure 31: Global Variables

Before starting any of our code we have to declare all the variables we need, this includes the variables for gyroscope calculation and the variables for the pins that correspond with the correct digital pins. Various libraries are also declared such as *SharpIR.h* and *<Wire.h>.*

They are used to calculate the distance using the optical sensor and to communicate with the gyroscope using I2C respectively. The address line for the gyroscope has been labelled as a “const int” This means that this value is constant and won't change. The flag variable is labelled as a “volatile int” which means that this value will change during an interrupt. The fact that these variables are initialised out of the main block of code makes these global variables helpful as we can use them in various functions and interrupt sequences.

```
void setup() {
  Serial.begin(9600);
  Wire.begin();                      // Initialize comunication
  Wire.beginTransmission(MPU);       // Start communication with MPU6050 // MPU=0x68
  Wire.write(0x6B);                  // Talk to the register 6B
  Wire.write(0x00);                  // Make reset - place a 0 into the 6B register
  Wire.endTransmission(true);        //end the transmission
  pinMode(FR_EN, OUTPUT);
  pinMode(FR_in1, OUTPUT);
  pinMode(FR_in2, OUTPUT);

  pinMode(FL_EN, OUTPUT);
  pinMode(FL_in1, OUTPUT);
  pinMode(FL_in2, OUTPUT);

  pinMode(BL_EN, OUTPUT);
  pinMode(BL_in1, OUTPUT);
  pinMode(BL_in2, OUTPUT);

  pinMode(BR_EN, OUTPUT);
  pinMode(BR_in1, OUTPUT);
  pinMode(BR_in2, OUTPUT);

    digitalWrite(FR_in1, LOW);
  digitalWrite(FR_in2, LOW);


  digitalWrite(FL_in1, LOW);
  digitalWrite(FL_in2, LOW);

  digitalWrite(BL_in1, LOW);
  digitalWrite(BL_in2, LOW);

  digitalWrite(BR_in1, LOW);
  digitalWrite(BR_in2, LOW);

 pinMode(push_button, INPUT_PULLUP);
 pinMode(Left_IR_SENSOR, INPUT);
 pinMode(Right_IR_SENSOR, INPUT);
PCICR |= B00000010;

PCMSK1 |= B00000111;
}
```

Figure 32: Setup()

The setup function is where the various components and systems are configured. We start with the line *Serial.begin(9600)* that allows serial communication with the serial monitor on the computer, the “9600” part is the stated baud rate, the Arduino and serial monitor must have the same baud rate to avoid “baud rate mismatch”. The Serial monitor is very important in debugging and fixing issues with your code, for example, you can make sure your code follows through with a function by adding a command prompt like *Serial.println(“finished”);* so if your Arduino runs through your code successfully the word “finished” will appear in your Serial monitor. To initiate communication via I2C you must use the code *Wire.begin();*. As we

previously declared “MPU” as the dedicated I2C address, the second line of code *Wire.beginTransmission(MPU)* selects our gyroscope as the device to communicate with, as previously mentioned using I2C you can talk to multiple slaves devices so we need to address the right one. The next two lines of code talk to the 6B register in our gyroscope which determines the sensitivity of the gyroscope

Register (Hex)	Register (Decimal)	Bit7	Bit6	Bit5	Bit4	Bit3	Bit2	Bit1	Bit0
6B	107	DEVICE _RESET	SLEEP	CYCLE	-	TEMP_DIS	CLKSEL[2:0]		

Figure: 33 6B register

To reset the register, write a zero into the register with *Wire. write(0x00);* the transmission is then ended. The next chunk of code determines which pins are inputs and outputs, all the pins related to the motor drivers are configured as outputs and all the pins relating to the sensors are labelled as inputs, this is done by using the *pinMode();* function. As we didn’t include a resistor with our button design, we are using the inbuilt pull-up resistor built into every single digital pin. The next chunks of code initialise the motors to stop moving. The motors are controlled by an H bridge and the two inputs control which combination of switches are closed or opened, as we have set both the inputs to low it means that the two switches that are closed are on the same side, this blocks the positive power supply with the ground and no current flows through the motor.

The last two lines of code initialise the pin change interrupts. On the Arduino Nano, there are 3 parts where pin change interrupt can be sensed and triggered, they are listed below:

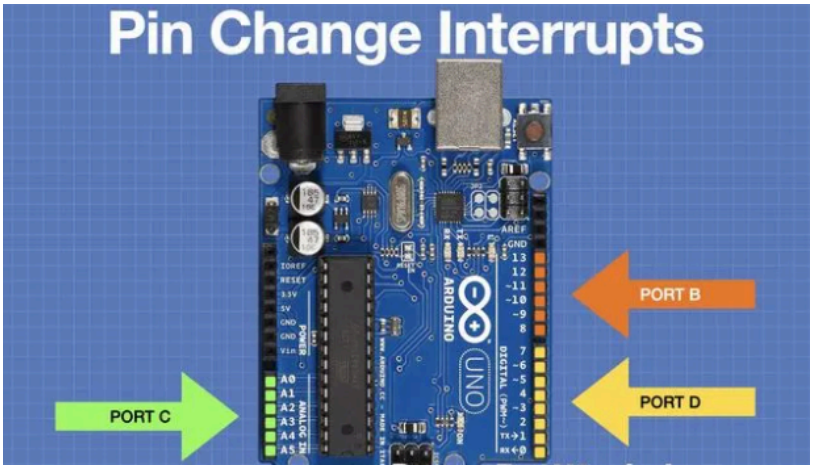

Figure: 34 Pin Change Interrupts Ports

In our case we will be using port C, to select the port we have to enter the correct binary combination into the PCINT register. For port C the combination is B00000010. Next, we have to tell the system which pins to monitor in the port, the button is connected to A2 and the IR sensors are connected to A0 and A1 so the binary combination for this is B00000111.

Currently, the robot is stationary awaiting the press of the button to start operation.

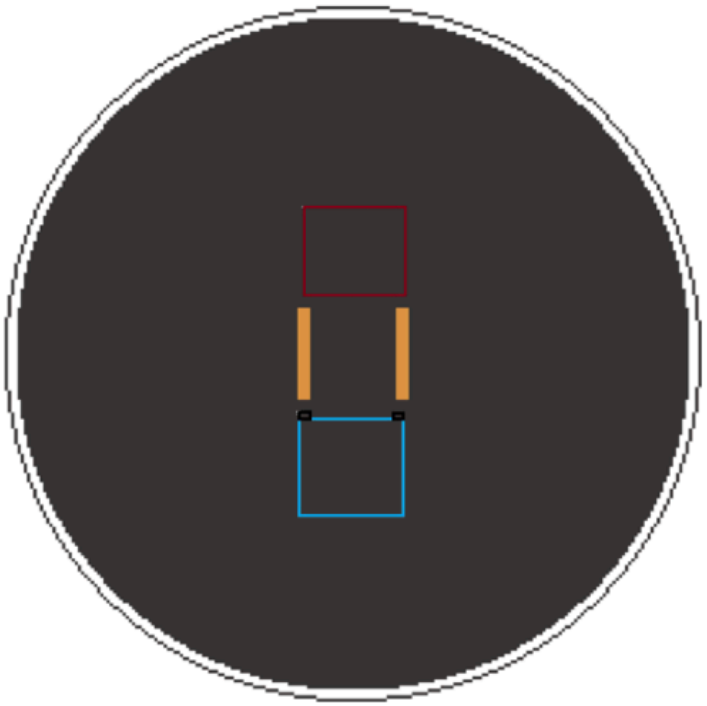

Figure 35: Starting Position of sumo robot ours (blue) and enemy (red)

Once the button is pressed the volatile variable flag is set to 1, as all the inputs are connected to the same interrupt port, we have to do further tests to determine which interrupt was set. When the button is pressed it sends out a LOW signal, if a low signal is read on the pin of the button, we can then guarantee that the button was pressed. The interrupt service routine finishes

```
ISR (PCINT1_vect) {
if(digitalRead(push_button) == LOW){

  flag = 1;
  Serial.println("button");
}
else if(digitalRead(Left_IR_SENSOR) == LOW){

  if(mySensor.distance() <= distance_cm){
    flag = 4;
  }
  else {
    flag = 3;
  }


}
else if(digitalRead(Right_IR_SENSOR) == LOW){


 if(mySensor.distance() <= distance_cm){
    flag = 4;
  }
  else {
  flag = 2;
  }
}

}
```

Figure: 35 Interrupt Service Routine

When flag 1 is set the Arduino then runs an internal delay using the *delay()* command; this command uses milliseconds as input, so to achieve a 5-second delay as per the specifications we would need to write *delay(5000)*. The code then triggers a random number generator ranging from 0-1, this gives us a 50% chance of getting 1 or 0 and depending on the answer the motor will turn either clockwise or anticlockwise.

Let's say the random number generator generates the value '0'. Then the inputs for the H-bridge are set for the right-side wheels to move forward and the left-side wheels to move backwards; this would cause a clockwise turning motion. The yaw angle is set to zero. The code then enters a while loop that can only be broken out when the robot has turned to a specific turning angle in this case 80. Inside the while loop, the Arduino constantly monitors the angle as it is rotating using the gyroscope. When the robot has turned an angle of 80 degrees the while loop is broken and the flag is set to 4. If the random number generator generates a 1 the same thing happens but the robot rotates counterclockwise at an angle of -80 degrees.

```
  if(randomNumber == 0){


digitalWrite(FR_in1, HIGH);
 digitalWrite(FR_in2, LOW);


 digitalWrite(FL_in1, HIGH);
 digitalWrite(FL_in2, LOW);

 digitalWrite(BL_in1, HIGH);
 digitalWrite(BL_in2, LOW);

 digitalWrite(BR_in1, HIGH);
 digitalWrite(BR_in2, LOW);


yaw = 0;
  while(yaw <= 80){
previousTime = currentTime;        // Previous time is stored before the actual time read
currentTime = millis();            // Current time actual time read
elapsedTime = (currentTime - previousTime) / 1000; // Divide by 1000 to get seconds
Wire.beginTransmission(MPU);
Wire.write(0x43); // Gyro data first register address 0x43
Wire.endTransmission(false);
Wire.requestFrom(MPU, 6, true); // Read 4 registers total, each axis value is stored in 2 registers

GyroX = (Wire.read() << 8 | Wire.read()) / 131.0; // For a 250deg/s range we have to divide first the
GyroY = (Wire.read() << 8 | Wire.read()) / 131.0;
GyroZ = (Wire.read() << 8 | Wire.read()) / 131.0;

GyroZ = GyroZ - 1.65; // GyroErrorZ ~ (-0.8)

yaw =  yaw + GyroZ * elapsedTime;

Serial.println(yaw);
  }
  flag = 4;
  Wire.endTransmission(true);
```

Figure 36: turning code

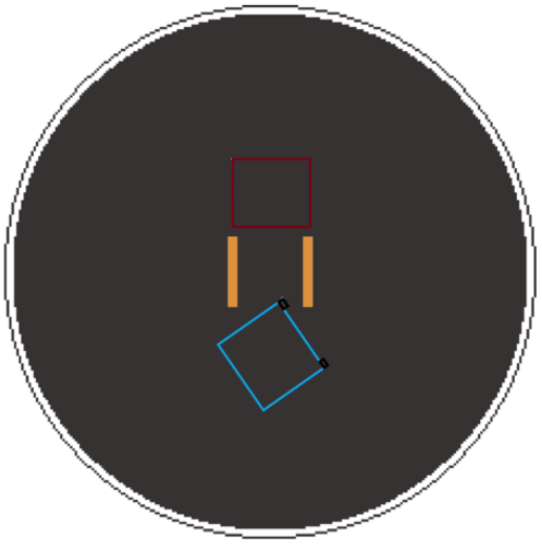

Figure: 37 initial turning position of the robot

Right now the robot should be facing away from the enemy towards the boundary of the dohyo, after turning the flag is set to 4 which sets the inputs to the H-Bridge to drive the motors forward towards the edge of the dohyo. This is the main part of our defence sequence mentioned by the specification, by continuously moving around the dohyo we will be nearly impossible to hit due to the unpredictability of our movements, therefore we can ensure an iron-clad defence.

```
while(flag == 4){
Serial.print("forward");

digitalWrite(FR_in1, LOW);
  digitalWrite(FR_in2, HIGH);


  digitalWrite(FL_in1, HIGH);
  digitalWrite(FL_in2, LOW);

  digitalWrite(BL_in1, HIGH);
  digitalWrite(BL_in2, LOW);

  digitalWrite(BR_in1, LOW);
  digitalWrite(BR_in2, HIGH);

}
```

Figure 38: flag 4 while loop

The flag of the robot won't change as it is a while loop so the robot will continue to travel forward until an IR sensor detects a white line and breaks out of the loop. Looking back at **Figure 35** depending on the IR sensor that triggered the Interrupt a different flag will be set where it will turn clockwise or anticlockwise just like at the beginning. The only difference to this code is that while turning the Arduino will constantly be checking the Optical sensor to see if there are any opponents in the vicinity and if any opponents are spotted the *"break;"* function is executed and while the loop is terminated, the flag is then set to 4 which instructs the robot to charge straight at the enemy. This is the main part of our attack strategy per the specifications.

```
if(mySensor.distance() <= distance_cm){
    break;
  }
```

Figure 39: break function

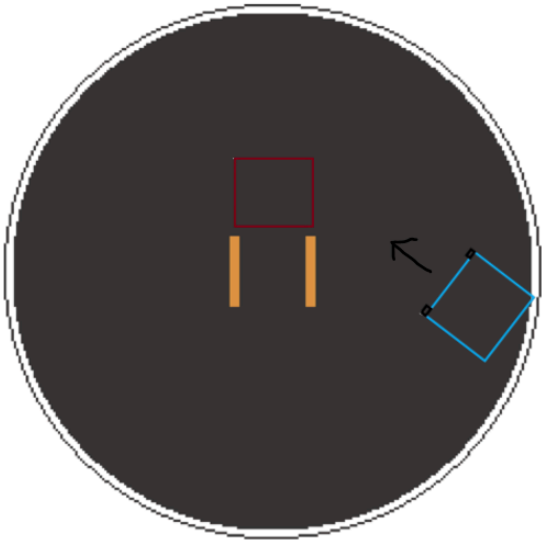

Figure 40: The robot turns to face the opponent

One problem with our design is that once our robot clashes with the enemy due to the position of our IR sensors and the opponent's proximity, the IR sensors will trigger an interrupt telling the Arduino to turn at an angle. To be able to push our opponent off we need to prevent this from happening. To rectify this we added safety measures to help the Arduino identify whether it is at the edge of the dohyo or right in front of the enemy. Referring back to the Interrupt Service Routine in **Figure 35**. Once the IR sensor interrupt is triggered the Arduino reads the distance measured from the optical sensor and determines whether it's at a boundary or in front of an enemy. If the output distance is less than the predetermined distance, we accurately say there is an enemy in front. This causes flag 4 to be set which instructs the robot to move forward eventually pushing the enemy off the doyo.

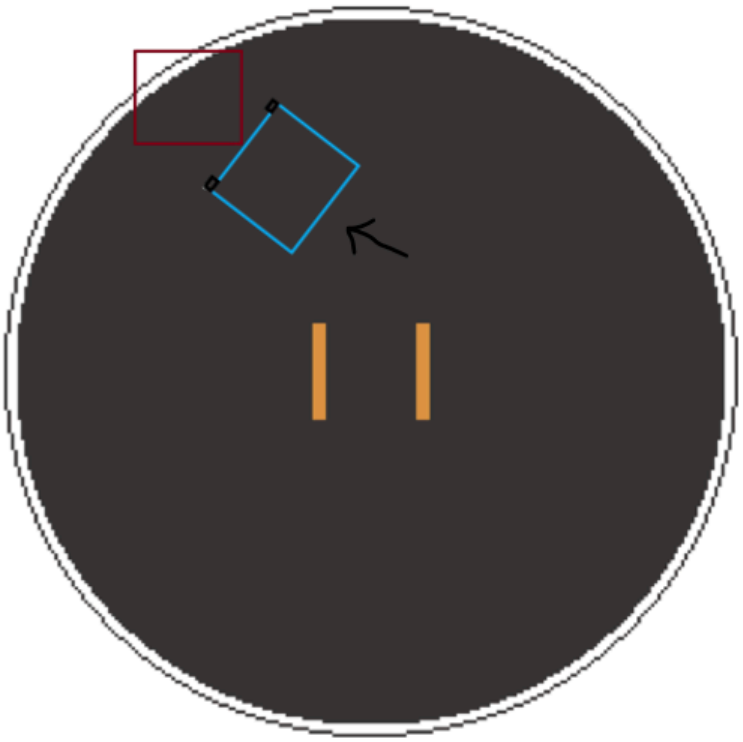

Figure 41: Sumo robot attack function

5. Testing and Results

5.1 Motors

The components being tested are 4 of the **Pimoroni 50:1 Micro Metal Gear Motor,** details on this motor can be found on page 9 of this document.

5.1.I Testing: Maximum pushing force

The pushing force of the sumo robot was tested under loads ranging from 0.5kg to 2kg. Data found on page 9 suggest the motor will stall when pushing 2kg, therefore 2kg was the maximum value tested.

- Initially, the robot was tested with only the bottom platform, 4 motors and 2 H bridge motor drivers, the motors were each powered using a DC power supply at 7.2V. The initial test weight of the robot was 0.1kg. They could effectively move weights up to 1.8kg with a linear decrease in speed as the load weight was increased from 0.5kg to 1.8kg. Beyond 1.8kg the motors would stall.

- Lastly, The complete robot was tested, the robot weighed 0.47kg and was powered by 2 Li-ion batteries (Information on the batteries can be found on page 12). The robot could effectively move Loads up to 1.4kg and would stall beyond 1.4kg.

Result:

The completed robot was able to push loads up to 1.4kg which is 0.6kg lower than the 2kgcm stall torque mentioned on page 9. The maximum pushing force decreases as the weight of the robot increases this is because the friction between the surface and the tyres increases. Friction will oppose the torque resulting in a lower stall torque, However, a pushing force of 1.4kg is more than sufficient as robots weigh a maximum of 0.5kg in the Mini Sumo Robot class.

5.1.II Testing: Breaking speed

A suitable motor speed was required to be acquired to allow the motors to effectively break. The motors must be able to effectively break fast enough when approaching the white perimeter of the dohyo, and maintain full control.

- Initially, the motors were tested at their maximum speed (420 RPM). At this speed the chance of the robot being able to effectively break was low, this is because the motors did not have enough time to effectively break.

- The breaking speed was then tested by altering the speed using the enable pins on the L298N motor drivers (Information about the motor driver can be found on page 15). The Enable pin was set from 255 (Max) to 180, at 255 the duty cycle of the PWM is 100% and at 180 it is 70% therefore at 180 the speed is 70% of the maximum (294 RPM).

Result:

From testing the speed from 100% to 70% of the maximum speed we found the robot was able to effectively break at 70% with minimal chances of error. Having a speed even lower has little effect but at the cost of a slower movement speed. Therefore we used the maximum speed at which the robot could effectively brake, this being 294 RPM (70% of maximum speed).

5.2 Gyroscope

The component being tested is the **GY-521 MPU-6050 3 Axis Gyroscope,** details on this gyroscope can be found on pages 16-17 of this document.

5.2.I Testing: Turning angle

The angle where the robots turned had to be altered multiple times, we had to make sure the path we set our robot out on would effectively search the area around the dohyo. We also had to make sure the searching angle was large enough to identify any enemies but too large that if no enemies could be located we wouldn't be facing the same boundary edge.

Result:

After several tests, we concluded the optimum angle of searching was 80 degrees. Thales theorem states if points A, B and C are points of a circle where line AC is the diameter of the circle, angle ABC is a right angle; this means if we used an angle of 90 as a turning angle we would effectively search half of the board as the board is a circle. We finalised on an angle a little less than 90 so it wouldn’t end up facing the edge again if an opponent was not identified

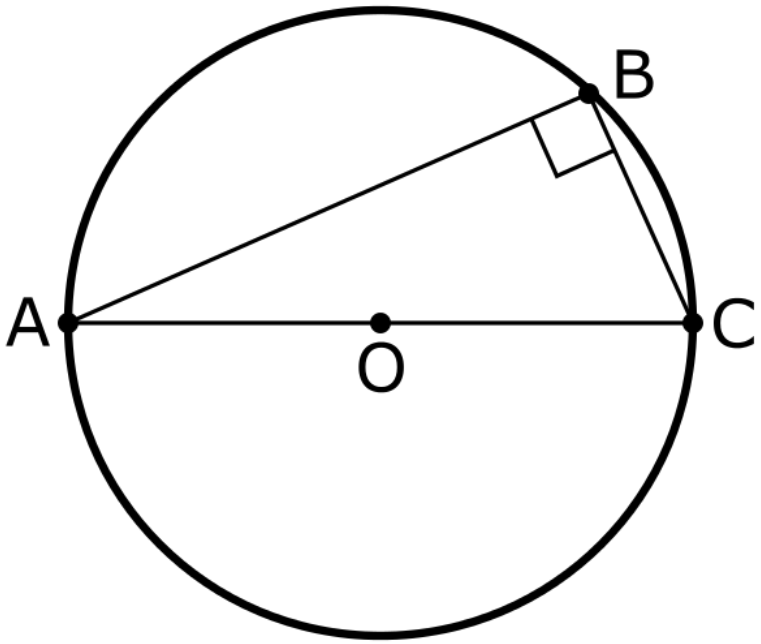

Figure 42: Thales Theorem example

5.3 Sensors

The components being tested are the IR sensor and the Optical sensor. Details on the IR sensor can be found on page 14, Details on the optical sensor can be found on page 18.

5.3.I Testing: IR sensor range

Our IR sensor had a maximum specified range of 20cms there were 2 of them located on the front, they are downward facing and were utilised for determining the white perimeter.

- Firstly, we tested for the maximum range we found that the range was just short of 10 cm, beyond 10cms the IR sensor was not functional.

- The range is altered by utilising the potentiometer on the sensor. We found through testing that 1 of the IR sensors had a highly sensitive potentiometer, This meant the operating range of the IR sensor was difficult to calibrate.

Results:

The IR sensors had a range that was more than viable; the 10cm range allowed us to have them elevated, and this could reduce the chances of them being damaged due to collision as they were at a height of 60 mms.

6. Discussion

All individual components were tested and mostly functioned as expected. We were successfully able to interface the Arduino Nano with I/O components. The robot was able to navigate the dohyo, detect opponents and the white boundary, turn with high precision utilising the gyroscope and push opponents off the dohyo.

In our model, 1 of the IR sensor modules would occasionally need recalibrating. This was because the potentiometer was very sensitive, and the operating range we required the IR sensor to function in was problematic to recalibrate. The IR sensor not functioning as expected posed a big problem as it sent crucial information to initiate a turn away from the white boundary. This issue was present in the demonstration, ultimately resulting in our robot's edge detection not functioning on 1 side.

The robot's ability to traverse was functioning at a high standard. The robot was able to accelerate to its capped speed relatively quickly due to the traction and torque provided by the motors and wheels. The robot was able to detect a white edge and decelerate quickly enough to prevent the robot from exiting the dohyo.

The gyroscope was advantageous as it allowed us to accurately make turns. Through testing, we were able to find a suitable angle to turn from the perimeter. 80 degrees was most suitable as it allows us to traverse around different points of the perimeter (This is explained on page 40).

Effectiveness and accuracy of code

To ensure the maximum speed and efficiency of the robot we had to make sure that the code was optimised and not convoluted. By adding global variables to each of the pins we reduced the need for writing the pin number each time we wanted to interact with it, also adding global variables makes the code much easier to understand from an outside perspective.

Using Interrupts drastically increased the speed and responsiveness of the program as it allows the Arduino to carry out other tasks instead of constantly checking whether an IR sensor has been set for example. This decreased the processing power needed to run the program. By using flags and conditional statements our code can adapt its behaviour depending on various conditions.

Overall our sumo robot was able to meet the specifications and requirements. Our search strategy was successfully implemented due to the efficiency of the code. We also believe that our attack and defend strategy is very innovative compared to more common attack and defend methods.

7. Further work

If we had an increased budget and time frame we would have improved our sumo robot by doing the following-

Implementing smaller motor drivers- to increase the compactness of the robot would allow us to have more space for the robot so we could have added an improved scraper. In addition to this, smaller motors would require less power, prolonging the robot's operational time.

Improved motors- would revolutionise our sumo robot's performance by increasing its speed, torque, and RPM. With greater speed, and could swiftly manoeuvre around opponents

Higher torque would empower the robot to exert stronger pushing force, enabling it to overpower opponents more effectively.

Also, improved precision would enhance the robot's control and stability, allowing for more accurate movements and strategic movements within the dohyo. Upgraded motors would elevate the sumo robot's agility, power, and control.

Faster microcontroller- improves the capabilities of our sumo robot by enhancing its processing speed, memory capacity, and overall efficiency. With a faster processor, the robot could execute commands more swiftly, enabling quicker reactions to opponent movements and more responsive navigation.

More complex algorithms and strategies can be implemented into our code with the increased memory capacity from an improved microcontroller, this would therefore improve our robot's decision-making abilities and overall performance.

Furthermore, enhanced efficiency enables the microcontroller to optimise power usage, prolonging our robot's operational lifespan and potentially providing a competitive advantage in prolonged competitions where there might be a stalemate.

To conclude, a better microcontroller would empower our sumo robot with greater speed, intelligence, and endurance, therefore enhancing its performance in the dohyo.

More accurate sensors - A more accurate infrared (IR) sensor would greatly enhance the performance of a sumo robot by extending its range of detection, increasing sensitivity to opponent signals, and reducing false positives and negatives.

This would enable faster response times, precise targeting of the opponent, and improved navigation in the dohyo. The robot would have an advantage in detecting opponents, reacting to their movements, and staying within the white line boundary of the dohyo.

Higher power battery- performance, operational time and power output would all be significantly improved if a higher power battery was equipped. A greater capacity battery allows the robot to sustain longer matches without the need to recharge or replace them, giving it an endurance and design advantage over other groups.

Moreover, the higher power output would provide increased torque to the motors, enabling the robot to exert stronger pushing force against opponents and manoeuvre more effectively in the dohyo allowing us to win.

Implementing thicker, higher grip wheels would improve traction, stability, and manoeuvrability on the dohyo. Increased thickness and grip of the wheels will have better traction against the floor, allowing the robot to maintain control and exert more force while it is pushing opponents without spinning out due to loss of control

Enhancing the grip would reduce slippage in dohyo, especially near the boundary and improve the stability and responsiveness of the robot to sudden changes in direction and movement especially when it detects the white line.

8. Strengths

Incredible propulsion that can apply up to 1kg of force ability gives us overwhelming strength in the dohyo and allows us to easily defeat our opponents.
With this power at our disposal, our robot can effectively manoeuvre and carefully position its enemies to gain a tactical advantage or eliminate the competition. Precise turning ability and powerful optical sensor technology provide superior agility and situational awareness

The ability to perform precise manoeuvres using our optical sensors, our robot can effortlessly move around the dohyo, quickly respond to changing conditions, and carefully adjust its position for best performance. Optical sensors improve functionality by constantly scanning the surroundings, allowing the robot to detect and track enemies with incredible accuracy

IR sensors accurately detect white borders and navigate perfectly -This innovative technology allows robots to stay within the boundaries of the arena, avoid penalties, and maintain a strategic position during combat. The IR sensor detects surrounding white edges and prompts the robot to perform quick planned manoeuvres to avoid them. This feature demonstrates not only the robot's agility and autonomy but also its reliability in avoiding potential obstacles on the dohyo.

Our lithium-ion battery provides enough power to power the engine even in extreme conditions such as stalling. Due to their high energy density and discharge rate, these batteries provide enough power to overcome resistance and keep the engine running at full power. This ability allows our sumo robot to maintain thrust and manoeuvrability even when faced with formidable opponents and obstacles.

Implementing interrupts, our Sumo robots deliver fast, responsive movements that improve performance on the battlefield Interrupts allow a robot's microcontroller to immediately interrupt its current task and respond to high-priority events, such as sensor input or enemy movement, in real-time. This allows critical actions to be processed without delay, allowing robots to respond quickly to changing conditions and perform precise operations with minimal delay.

9. Weaknesses

Challenges with our IR sensors, particularly in reliably detecting boundaries. There have been instances where our sensors failed to detect the boundary lines accurately, which resulted in losses during the tournament as our robot went off the edge

The height of our sumo robot was an obstacle, especially in the dynamic environment of the sumo ring.

The robot's inherent height can cause instability issues when making sharp turns Instability affects the ability to perform precise manoeuvres and can affect performance during games where turns are particularly difficult to navigate and require careful consideration to maintain balance and control.

This highly relevant challenge highlights the importance of constantly evolving design and strategy to ensure optimal performance.

The design of our ramps presents unique challenges, particularly regarding their placement and potential impact on movement and stability.

If the ramp is not optimally placed, it will prevent the robot from moving smoothly and may cause it to stumble or lose traction. In addition, a misplaced ramp will make the robot unstable, increasing the risk of falling or losing control, especially during high-speed manoeuvres or collisions with enemies. Solving this issue is important for the ramp to effectively serve its intended purpose and improve performance in the sumo arena while minimising the risk of negative effects on movement and stability.

Weight management presented a significant challenge for our sumo robot, as the distribution of components had not been optimised to achieve an ideal centre of gravity. Without proper distribution, our robot experienced issues related to stability and manoeuvrability, especially during aggressive movements or when encountering opponents. An imbalanced centre of gravity led to difficulties in maintaining control and increased the risk of tipping over during matches.

Adjusting the potentiometer of the IR sensor [MAJOR ISSUE] proved difficult for our sumo robot. Fine-tuning this component was critical for improving sensor sensitivity and precision, assuring accurate identification of opponents and borders. However, problems occurred during this procedure, as incorrect modifications could lead to inconsistent readings or even sensor breakdown. To achieve the proper balance of sensitivity and reliability, the potentiometer needed to be calibrated with precision and rigorous experimentation. Despite these problems, fine-tuning the potentiometer was ultimately necessary to improve the overall performance of our sumo robot, allowing it to detect opponents and limits with greater precision and consistency.

The ambient lighting conditions had a considerable impact on our sumo robot's performance, especially during IR sensor calibration and operation. Variations in lighting levels could interfere with sensor readings, resulting in inaccurate opponent detection and boundary recognition. Bright brightness, for example, may cause sensor saturation, resulting in inaccurate readings or lower sensitivity. In contrast, low-light situations may reduce sensor efficacy, limiting our robot's capacity to effectively recognise opponents or borders.

10. Conclusion

To conclude, based on the specifications required the sumo robot is highly capable of participating in competitive contests, the robot's design includes a robust drive system that's capable of exerting force pushing objects over 1kg which is vital to overpower opponents. It can make accurate turns and detect any nearby opponents in the vicinity as well as manoeuvring around the white border of the sumo ring demonstrating high sensory abilities. The use of interrupts in its programming ensures the robot's swiftness and responsiveness during the competition, the use of lithium-ion batteries which are known for their high energy density allows the robot's motors to reach peak performance ensuring it can operate at full capacity while attacking and defending without compromising operational time or safety. Overall, the robot is engineered for power, accuracy and durability with the ability to perform well in competitive environments.

11. References

[1] Arduino.cc. (2024). Available at:
https://docs.arduino.cc/static/47180601da55c458736e09d26b8bfab5/d0d8c/Atmega168PinMap2.png [Accessed 12 Apr. 2024].

{www.geeetech.com. (n.d.). Arduino Nano - Geeetech Wiki. [online] Available at: https://www.geeetech.com/wiki/index.php/Arduino_Nano.

Arduino.cc. (2024). Available at: https://docs.arduino.cc/learn/microcontrollers/analog-output/.

Wikipedia Contributors (2019). Lithium-ion battery. [online] Wikipedia. Available at: https://en.wikipedia.org/wiki/Lithium-ion_battery.

Dejan (2019). Arduino DC Motor Control Tutorial - L298N | PWM | H-Bridge - HowToMechatronics. [online] HowToMechatronics. Available at:

https://howtomechatronics.com/tutorials/arduino/arduino-dc-motor-control-tutorial-l298n-pwm-h-bridge/

https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/

Dejan (2019). Arduino and MPU6050 Accelerometer and Gyroscope Tutorial - HowToMechatronics. [online] HowToMechatronics. Available at: https://howtomechatronics.com/tutorials/arduino/arduino-and-mpu6050-accelerometer-and-gyroscope-tutorial/.

HowToMechatronics (2015). MEMS Accelerometer Gyroscope Magnetometer & Arduino. [online] HowToMechatronics. Available at: https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/

https://howtomechatronics.com/how-it-works/electrical-engineering/mems-accelerometer-gyrocope-magnetometer-arduino/

www.twi-global.com. (n.d.). What is PLA? (Everything You Need To Know). [online] Available at: https://www.twi-global.com/technical-knowledge/faqs/what-is-pla#:~:text=Polylactic%20acid%2C%20also%20known%20as,distillation%20and%20polymerization%20of%20petroleum.