



Sumo Robot Design & Build

Group 5 – Epsilon

Queen Mary University of London – ECS514U

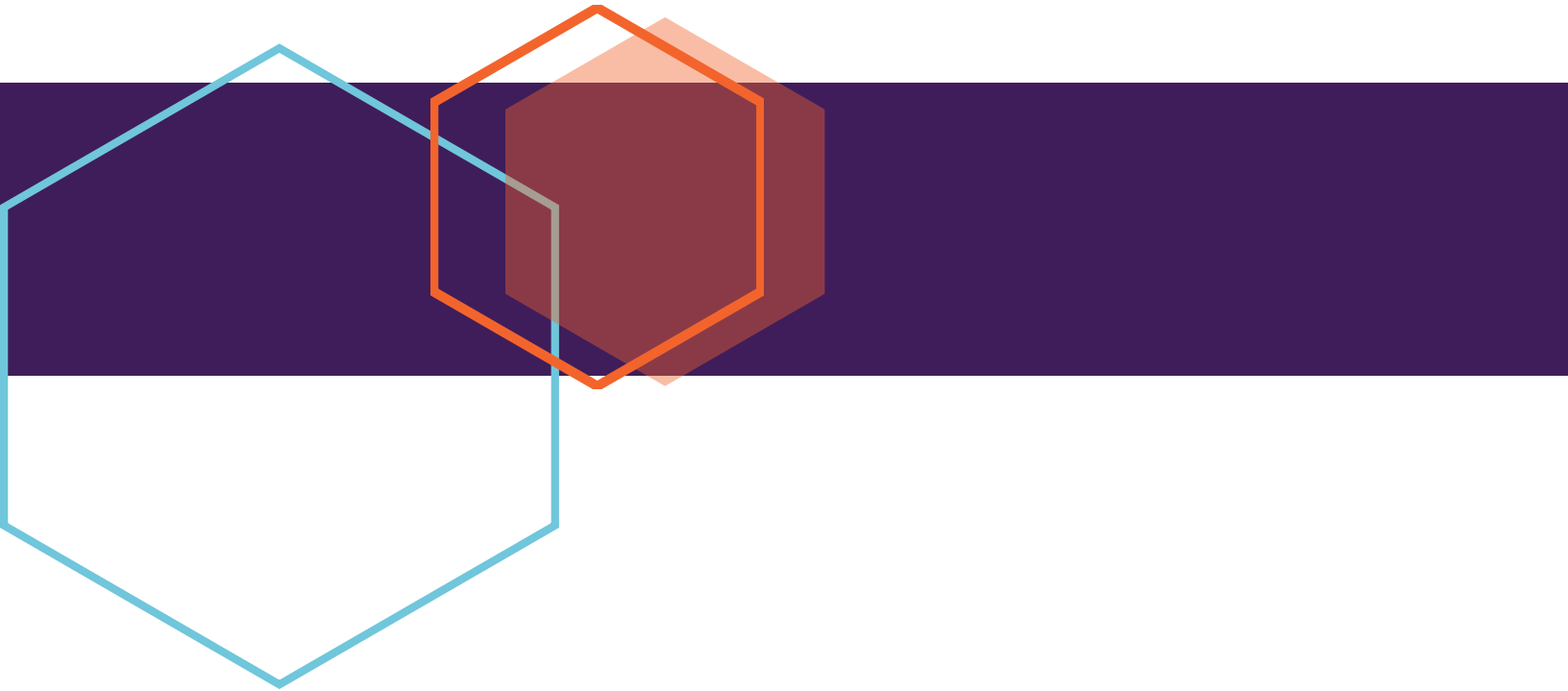
Group Leader – Callum Smith, 161042473: cjusmith@gmail.com

Sumaya Abdalle, 170415039: sumaya_abdalle@outlook.com

Nicholas Lee Hawthorne, 170209298: nickhthorne@gmail.com

Oluwatobi Adebari, 170309820: o.adebari@se17.qmul.ac.uk

Mazlum Tekingunduz, 160345902: m.tekingunduz@se16.qmul.ac.uk



Executive Summary

The objective of this project was to design, build and program an autonomous Sumo-Robot. The Robot had to be able to move within the boundary of the arena without falling off whilst also detecting and pushing other robots out of the ring.

Constraining each group's design and component choices were the user requirements which included a size, weight and budget limit.

There were many other factors considered for the design of the robot, one being the strategy the robot employed inside the ring.

Our Robot featured a unique addition, using an RGB LED that changed colour depending on the state of the robot. This would help us to effectively and efficiently pinpoint at exactly which state an error occurs and give some user interaction and feedback on the robot's processes.

We had a unique chassis design, with stacked layers and other additions including feet and ball casters that enabled our robot to win the competition undefeated with 6 wins and 0 losses.

All these features will be covered in detail in the following report. We will outline our design choices, justifications and the problems we encountered to give an insight into how we built a successful SUMO-ROBOT.

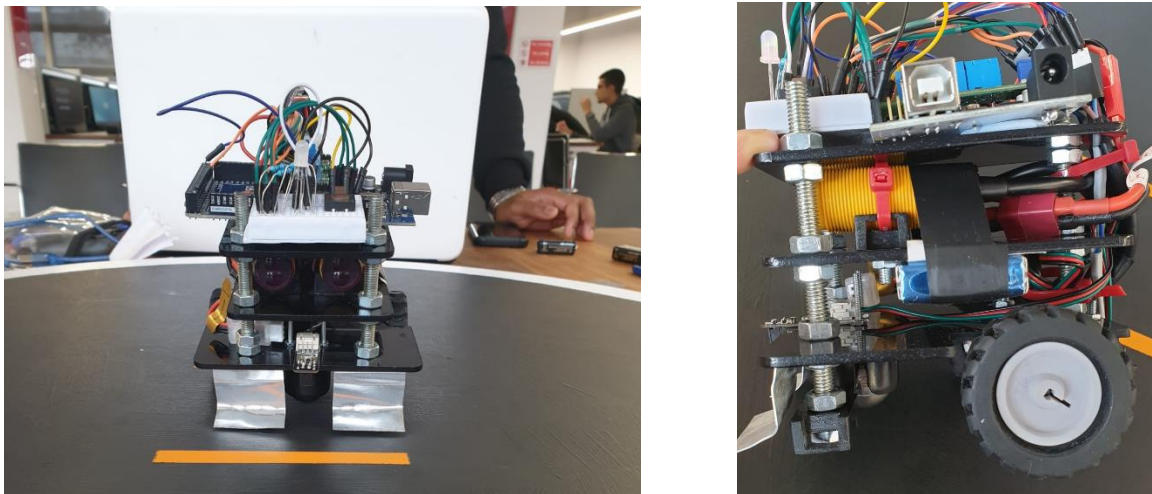


Figure a1: Finished Product of our robot, Epsilon

Index

Chapter 1: Introduction.....	4
Chapter 2: Specification of requirements.....	5-8
--2.1: Problem Statement and Project Aims.....	5
--2.2: User Requirements.....	5
--2.3: Constraints.....	6
--2.4: Input and Output Requirements.....	6
--2.5: Operational Requirements.....	7
--2.6: Functional Requirements.....	7
--2.7: Project Timeline.....	8
Chapter 3: Methodology and Design.....	9-33
--3.1: Design Approach.....	9
--3.2: Microcontroller.....	10-12
--3.3: Chassis.....	13-16
--3.4: Wheels.....	17-19
--3.5: Line Detection Sensor.....	20-21
--3.6: Object Detection Sensor.....	22-23
--3.7: Motor and Motor Driver.....	24-25
--3.8: Power Supply.....	26-27
--3.9: Sumo Robot Entity Overview.....	28-29
--3.10: Code Design and Creation.....	30-33



Chapter 4: Testing and Results..... 34-57

--4.1: Methodology for Testing.....	34
--4.2: White Line Sensor Tests.....	35-40
--4.3: IR Object Detection Tests.....	41-44
--4.4: Motor Tests.....	45-47
--4.5: Motor Driver Shield Tests.....	48-51
--4.6: Battery Tests.....	52-54
--4.7: Wheel and Ball Caster Tests.....	55
--4.8: Functional Testing of the Sumo-Robot.....	56-57

Chapter 5: Discussion..... 57-61

--5.1: Project Challenges.....	58-60
--5.2: Overview.....	60
--5.3: Future Improvements.....	61
--5.4: Conclusion.....	62

Chapter 6: Bibliography..... 63-66

1: Introduction

Robot Sumo is an arena game in which 2 robots, sharing the same height and weight restrictions, battle to push one another out of the ring. The following report will be detailing how we built, from the ground up, a Sumo Robot that navigates the arena, detects opponents inside the arena and tries to push the opponent out of the arena. Our robot must achieve all these requirements autonomously.

Our solution uses two Infrared Colour Detectors coupled with 2 Infrared Distance Sensors providing sensory input to the Microcontroller. The Microcontroller then takes that information and actuates two separately controlled Motors via the Motor Driver. Our Robot implements a feedback system of program status indication via an RGB LED that changes colour depending on the state of the Robot. Our chassis design is highly adaptable, allowing us to make changes and adjustments throughout functional testing to calibrate object sensor direction and weight distribution. The only user interaction required for this Robot is a button press to take the Robot out of standby.

In this report we will discuss the subsystems behind the design of this Robot. We will delve into the design choices for components and code functions for both high and low level and justify those choices through research and testing. We will be detailing how the components work, how they link together and how they work as a unit to form a single entity of a Sumo-Robot. We will conclude this report with our results, followed by a discussion that looks at the challenges faced during the fabrication of this robot and any improvements that could potentially solve some of these issues.

2. Specification of Requirements

2.1 – Problem Statement and Project Aims:

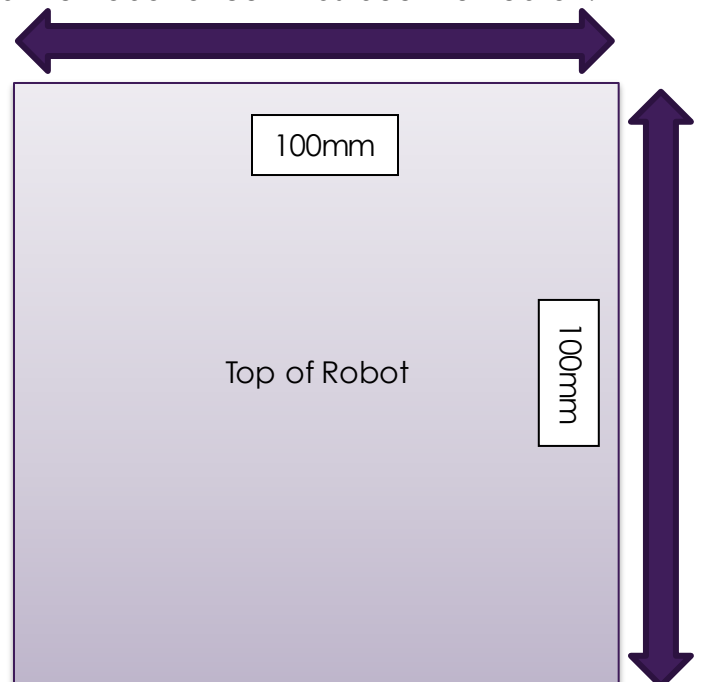
ROBOT-SUMO is an arena game within which two autonomous robots battle to push each other off or out the ring. The robots must be able to autonomously navigate the arena, detect the other robot and push it off as quickly as possible whilst also avoiding getting pushed off itself. The robot must be able to detect the boundaries of the arena and detect and track a moving or static object within the arena. The robot must be battery powered, with enough life to last at least one battle. Any solutions found for phase 1 (static object) must be able to scale towards phase 2 (moving object).

2.2 – User Requirements:

The ROBOT-SUMO must be able to perform all its tasks without any interaction from the user. The robot will navigate the arena with motors and actuators, detect and track objects, and react autonomously based on the information received from the Object and White Line sensors.

The Robot must:

- Move around the arena without falling off.
- Detect another object inside the ring irrespective of whether it is static or moving.
- Be able to push said object out of the ring once detected.
- Have no interaction between the user and the Robot once it has been turned on.
- Perform all these tasks Autonomously.
- Battery Powered.
- Conform to the base size limit:
 - 100mm x 100mm.
- Conform to the maximum weight limit:
 - 500g.
- Be within budget:
 - <£100 in total.
- No Height Limit





2.3 – Constraints:

The IR sensors will be placed in a location to prevent interference from external lighting sources imitating the white arena lines.

Due to the time available for the project we will constrain the functionality of the robot by using fewer sensors; two for object detection and two for edge detection. The microprocessor will not have to interpret as much data from the sensors, reducing the amount of programming required. This will also help keep under the budget limit.

Phase 2 of the project must be considered whilst working on Phase 1 to upgrade the functionality easily and ensure we have a smooth transition between Phase 1 and Phase 2.

The speed of the robot must be limited as there will be no dedicated braking mechanism. This will make it easier to stop the robot within the arena using just motor controls. The size of motors must also be limited as the microcontroller can only supply a restricted current which may not be enough to drive the motors.

Instead of a separate sensor for tracking we will utilize movement of the Robot to scan for the object inside the arena.

We will make the housing of the robot easily accessible so that when changes need to be made, they can be done so easily and without having to take the whole robot apart.

2.4 – Input and Output Requirements – Total I/O Pins required = 12-13:

- A button will be used on the ROBOT-SUMO to activate the function of the robot – 1 Pin.
- An RGB LED will be used as an indicator for the state that the robot is currently functioning in, switching between states based upon sensory information provided by the Object sensors and White Line Sensors – 3 pins.
- 2 Object Detection Sensors will be used to detect the location of the opponent in the arena. – 2 pins.
- 2 White Line Sensors, placed on the front and back of the ROBOT-SUMO, will be used for arena edge detection – 2 pins.
- The actuators will be two motors attached to two independent wheels to control the movement of the ROBOT-SUMO depending on its sensory information. The microcontroller will be controlling its direction and distance of travel – 0 pins.
- A Motor Driver will be required to provide enough current for the motors – 4-5 pins.



2.5 – Operational Requirements:

Mode 1 – Standby:

- In this mode the SUMO-Robot will be in standby waiting for the button to be pressed. This is necessary to allow us to place the robot in the arena without the robot performing its normal functions before the battle begins. A Flashing red LED will indicate that the robot is in Standby mode and once the button is pressed it will engage into Mode 2, Normal Operation.

Mode 2 – Normal Operation:

- ROBOT-SUMO will navigate the arena autonomously, attempting to detect another object inside the arena. If an object is detected it will trigger a process to push that object out of the ring. If the ROBOT-SUMO meets the edge of the arena it will perform a manoeuvre to move away from the edge and carry on tracking for an object. If the ROBOTSUMO fails to detect it will carry on its movement and tracking processes indefinitely. The difference in LED colour will indicate the process or function that the Robot is currently performing.

2.6 – Functional Requirements:

The ROBOT-SUMO will use two object detection sensors to detect an object, either stationary or in motion, within the arena boundaries. The movement of the motors and actuators will be dependent on the information received from the sensors. If the Robot Detects another object inside the arena the motors will move the robot towards the object and push it off. LED's will be used as a method of indicating whether the sensor has detected an object within the arena boundaries, giving a clear visual output for detection. This will help with low level troubleshooting. If, for example, the sensor does not pick up any information the output of the motors and actuators will navigate the arena with the White Line sensors until an object is detected within the specified range. The Robot will use an IR sensor at the front and back of the housing for detection of the edge of the arena. An increase in light reflected by the white lines will indicate that the robot is near the edge. When the sensor detects a white line the microcontroller will start a procedure that moves the robot away from the edge of the arena dependent on which sensor the information came from and then turn the robot based on a specified angle to resume scanning.

A button will be used to control whether the Robot is on Standby or Normal with an LED to indicate the robots state of operation. While the Robot is powered on the default mode is standby whereas pressing the button will allow it to enter normal operation where object detection will trigger an aggressive pushing function.

2.7 – Project Timeline:

Dates	Tasks to be completed
21 st January	Agree on Main requirements and preliminary design of the robot.
21 st January	Begin Background Research into the components required for our design.
25 th January	Project Specification Deadline.
30 th January	All Main components are bought and ready for testing.
15 th February	Complete and test for Phase 1 of the Robot.
18 th February	Mid Project Presentation and Demonstration
25 th February	Begin Report Writing for Final Submission
13 th March	Complete and test for Phase 2 of the Robot.
13 th March	Prepare for Final Project Presentation and Sumo Robot Battles
20 th March	Final Project Presentation and Robot Sumo Battles
29 th March	Final Project Report Deadline
12 th April	Final Project Viva Deadline

Figure b1: Table showing timescale of Sumo-Robot Building Project.

3. Methodology and Design

3.1- Approach to designing our robot:

We decided to take a modular approach to the design of our robot. Our robot must perform certain functions and it requires components to do so.

We broke down our requirements to a lower level view with more consideration taken for the components. The robot needed to:

- Move around the arena without falling off – requires motors, a motor driver and a White Line Sensor to detect the edge of the arena.
- Detect another object inside the ring irrespective of whether it is static or moving – requires object detection sensors.
- Be able to push said object out of the ring once detected – requires a microcontroller to control motors based on the sensory information.
- Have no interaction between the user and the Robot once it has been turned on – microcontroller with all sensors used as control signals.
- Perform all these tasks Autonomously.
- Battery Powered – need a rechargeable, lightweight battery.
- Conform to the base size limit – components must be small enough to fit within 10x10cm base.
 - 100mm x 100mm.
- Conform to the maximum weight limit – requires fabrication using a material that is light but durable along with lightweight components.
 - 500g.
- Be within budget – components chosen must balance accuracy with economy.
 - <£100 in total.

This Project is split into two phases:

- Phase 1:
 - Design a Robot that can move within a restricted area, detect a stationary object and then push it out.
- Phase 2:
 - Modify this robot so that it can identify another moving robot inside the arena and push it out.

The only difference between phase 1 and phase 2 is the inclusion of a moving object instead of a stationary one. With this in mind, we must design our Robot so it can be easily upgraded to detect a moving object as well as a stationary one.

We began by researching and choosing our Microcontroller, as it's compatibility with other components was the top priority and it is one of the more expensive parts of our robot. This will be followed by all other components, and finally a design of the code based upon the choice of components and how they will interact with our Microcontroller.

3.2 – Microcontroller:

The microcontroller is a very important component of our Sumo-Robot. Victor et al, points out that the microcontroller “ensures all the electrically and mechanical elements are working together to give the desired result” (Victor, Costachioiu and Constantinescu, 2013) [1]. The microcontroller processes input signals received from sensors and then performs functions specified by code implemented on to the microcontroller. Output signals are then sent to actuators to perform output functions. The microcontroller will ensure that our Robot can function autonomously.

When considering which microcontroller to use there are many variables we need to consider. These include:

- Price
- Dimensions
- Power consumption
- Compatibility with other components
- Members' previous experience with microcontrollers

To determine which microcontroller would meet our requirements and ensure we achieve our aims, extensive background was carried out. This would help us decide on a suitable microcontroller that we as a group can work with to carry out our tasks.

Victor's article talks about his experience when working with engineering students to create a SUMO Robot [1]. In his case an Arduino Uno was used shown in figure c1. There are many articles outlining the advantages and the disadvantages of this microcontroller. I, Sarwar points out that Arduino microcontrollers are “Ready to Use” (Sarwar, 2013) [2]. Evidence of this is that the Arduino Uno comes with a starter kit, which includes a 5V regulator and a serial communication interface. As well as that, Arduino has a library of example code and block diagrams to connect LED's and other components. This means that the Arduino Uno is easy to use and to program because there are many resources available to help us program our microcontroller. In addition, Arduino is written in C/C++. As we have taken a module in C programming, we will find the coding aspect much easier on the Arduino microprocessors.

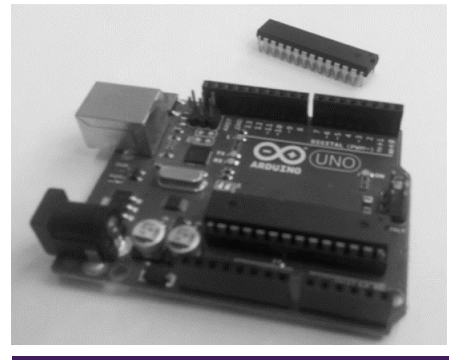


Figure c1: Arduino UNO
Microcontroller

When considering the price, on the Arduino official website an “Arduino Uno Rev3” costs 20 euros (Arduino, 2019) [3]. This is around £17.17. As our budget is £100, choosing this component would take up a small amount of our budget considering that we expect the microcontroller to be one of largest expenses we would have to make. However, when looking at figure c1 we can see that as the Arduino Uno is small. There are only 13 digital and 6 analogue pins. This is important to consider as we have a lot of components to interface with the microcontroller and we need enough pins to ensure we can have full working system.

Instead of using an Arduino Uno, Raspberry Pi (Fig. d1) can also be used as a microcontroller for SUMO robot. A key advantage of Raspberry Pi is that it is more powerful than the Arduino Uno. Jayat states that “Pi is capable of doing multiple tasks at a time like a computer” and is “40 times faster than Arduino” (Jayat, 2016) [4]. This means that this microcontroller can conduct decision making in a shorter amount of time. This is important when battling other opponents because if the Robot can conduct output functions faster it increases our chance of winning. Unlike the Arduino the Raspberry Pi can be coded in over 8 coding languages such as Python, C and C++. This means that if any of our group members are more proficient in another language it would be possible to use a Raspberry Pi.



Figure d1: Raspberry Pi Microcontroller

However, since the Raspberry Pi is much more powerful than the Arduino, it can range from £30 to £40. This would take up a considerable amount of our budget. This could mean that we may not be able to afford other components. Also programming Pi is more complex than Arduino as there are not as many libraries and components that are built to be compatible with Pi as there are for Arduino. From the Raspberry Pi 3 B+ datasheet it can be seen that to function Raspberry Pi needs 5 volts and 0.7 to 2.5A operating current (Raspberry Pi, 2019) [5]. This is a higher operating current than Arduino and if we use this microcontroller our batteries will continue to run out.

After consulting many articles on different versions of Raspberry Pi and Arduino microcontrollers, we as a group have decided to go with the Arduino Mega 2560 (Fig. e1 (Arduino, 2019) [6]).

When comparing the Arduino Mega to other microcontrollers, it is relatively low cost. On Amazon and the official Arduino website the Mega 2560 can range between £15 and £30. As the cost is low, there will be enough money left in the budget for the other components. In addition, the datasheet shows that the Mega is within the 10x10cm user requirement boundary (Fig. d (Arduino, 2019) [6]).



Figure e1: Arduino Mega2560

An advantage of also using this microcontroller is that components are made by companies to be directly compatible with the Arduino. This also means they come with their own libraries. This will make testing easier as interfacing the Arduino Mega with other components such as the sensors will not be difficult. An advantage of Arduino is the ability to “stack with shields” (SparkFun, 2015 [7]). For our design a driver shield is needed to allow us to change the speed of each motor individually. As the Mega 2560 allows interfacing of compatible shields we can meet our design needs. Three members in our group have also used Arduino microcontrollers before. This means that we won’t have much difficulty programming the device and interfacing the components should be a streamlined process.

OVERVIEW	
Microcontroller	ATmega2560
Operating Voltage	5V
Input Voltage (recommended)	7-12V
Input Voltage (limit)	6-20V
Digital I/O Pins	54 (of which 15 provide PWM output)
Analog Input Pins	16
DC Current per I/O Pin	20 mA
DC Current for 3.3V Pin	50 mA
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8 KB
EEPROM	4 KB
Clock Speed	16 MHz
LED_BUILTIN	13
Length	101.52 mm
Width	53.3 mm
Weight	37 g

Figure f1: Technical Specs of Arduino Mega2560

3.3.1 – Preliminary Designs:

The main function of the robot chassis is to house all the components of the robot in the most systematic way possible. With the user requirements such as dimensions and weight in mind, we need a material that is light-weight and sturdy.

One of the early sketches we made was one that housed all the components in one section. This proved to be difficult to implement because the IR distance sensors would have to be taped on the side walls or hung on a pillar extending from the inner floor of the robot. With this model, it would also mean that the front lower part of the robot would be scraping the floor thus damaging the line detector.

This second sketch of a chassis design should be more robust because of the added ramps in the front. With the ramps added, not only are the line sensors protected, it would also act like an attack mechanism helping to push an opponent off the arena.

The reason why this design was discarded was because it was too low and therefore might not detect an opponent as easily as it would have if it was taller. Like the first design, this chassis design would cramp all the components that needed to be housed in a single layer, i.e. the microcontroller, the battery, motors and IR detectors.

Another option is to use stacked layers to house all the components for ease of access to the components, easy modification and upgradeability; a key consideration when working through Phase 1 to Phase 2.

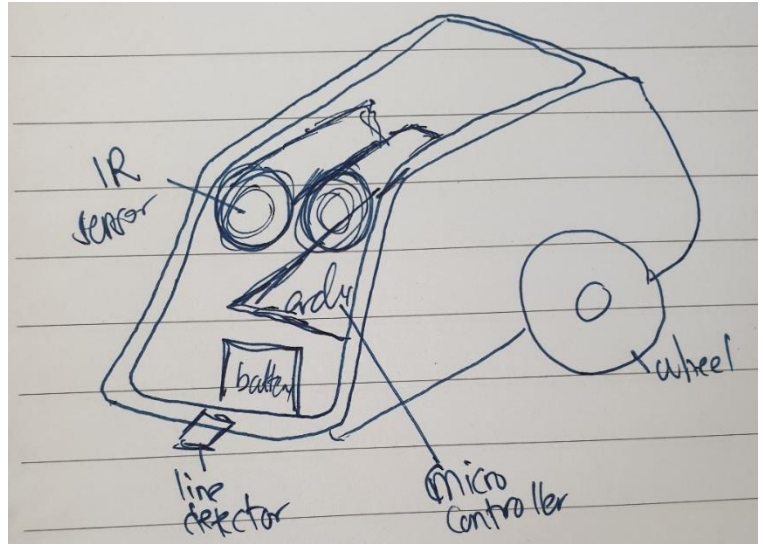


Figure g1: Design A of our Robot

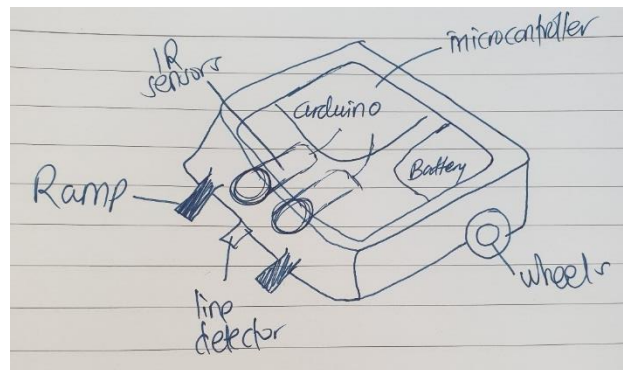


Figure h1: Design B of our Robot

3.3.2 – Chassis Design:

A robot chassis design that has layers would address the problem of component arrangement. Torn between acrylic and aluminium for the material to build the layers, we decided to go for acrylic as it had a lower density, helping us towards our goal of staying within the 500g limit. Acrylic was also chosen because it was easier to drill through to make room for the pillars to hold the layers together. With this structure, we could easily place components on specific layers and change them if necessary.

Made from aluminium threaded rod, the pillars held the layers together. Even though the rods were heavy, we had in mind that most of our components would be light. Therefore, linking the layers with these rods helped bring the weight close to the maximum limit. The pillars were also placed strategically to make room for the actual components as we did not want them to be on the way of larger elements like the battery and the microcontroller.

The lowest layer had the wheels and motors attached to them. Because the motors drove the wheels, we had to place them reasonably close to each other. This was why cut outs were made on the acrylic of that layer; to make room for free wheel movement. This layer also had both the front and back line sensors. To make this line sensor as efficient as possible, we placed it in an area that wouldn't be under the shade of the rest of the other components.

The second layer had the IR distance sensors. We placed these sensors in this middle layer because if we had placed them in the lower layer, it would have been too low for it to sense an object of a specific height. Too high and it would be difficult to calibrate it to sense objects that were on a lower elevation.

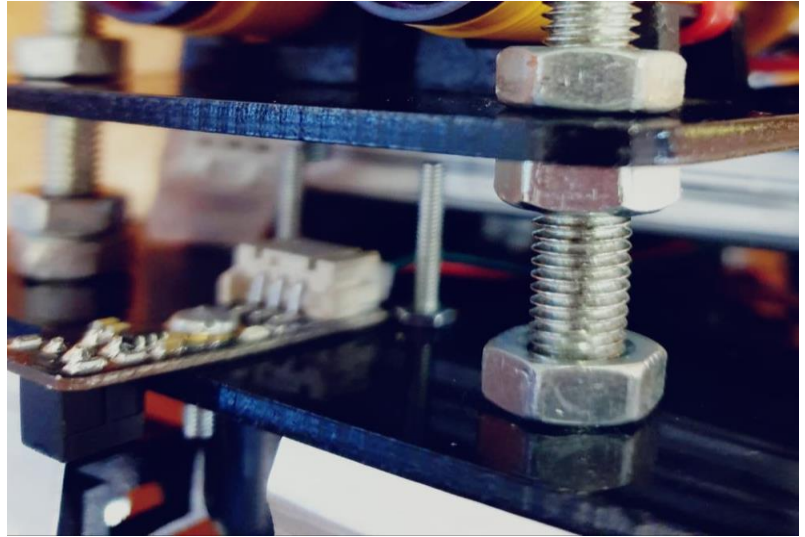


Figure i1: Rod Fixings of our Robot

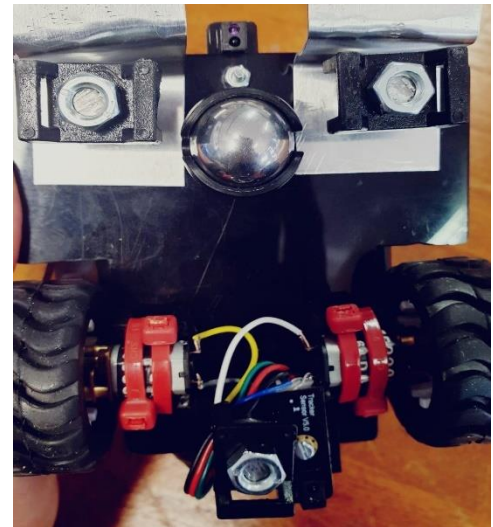


Figure j1: Underside of our Robot

We taped the battery beneath this layer because the battery was one of the heavier components and it being in the middle meant the center of the gravity was below the pivotal point, so this helps the robot balance itself [8].

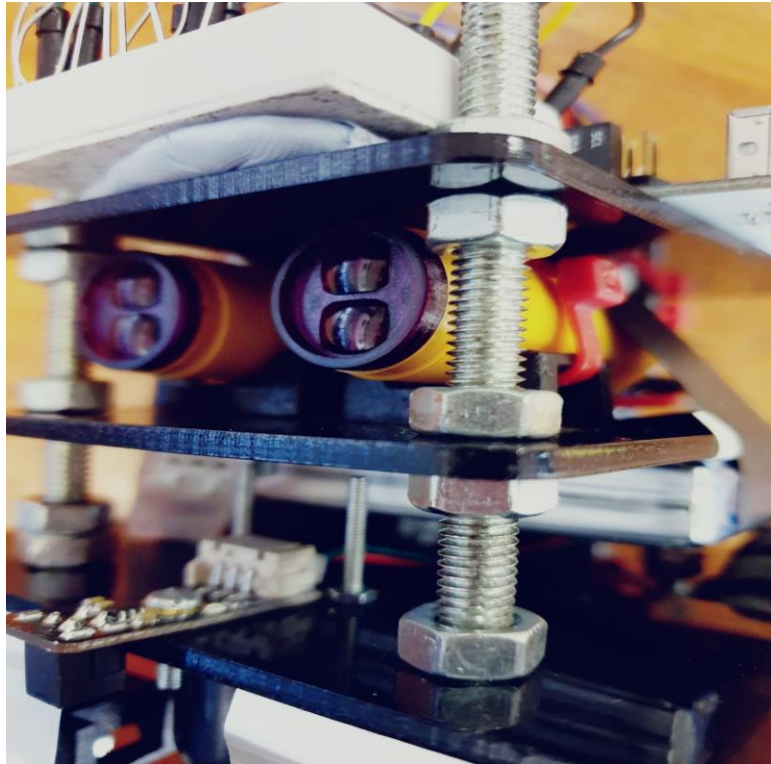


Figure 11: Middle Layer of Robot

The third uppermost layer had the Arduino microcontroller. This was placed here because access to the pins would be simple compared to if it was on the lower or middle layer. The user interactive LED that changed colour depending on which state the robot was in provided a good feedback system to anyone viewing the robot [9].

Despite all this, the robot was not as stable as we expected it to be, so we extended the pillars to the bottom of the robot just above the wheels. With added mounts that acted like feet and this helped stabilize the robot.

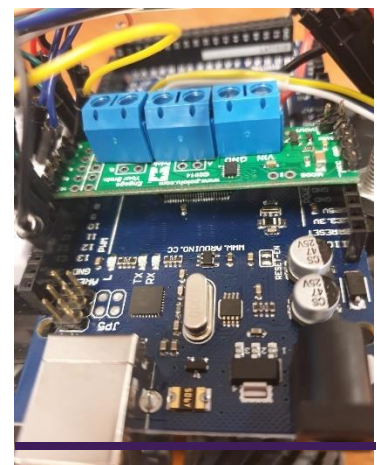


Figure m1: Top Layer of Robot

Although the layers are now populated with the components, it is not ready to fight. The need for offence mechanism is required and that is why we implemented the ramps of design sketch B. The ramps were made from half a millimeter thick aluminium sheets. This meant that it was malleable and easily to mount to the bottom layer.

Implementing layers onto our sumo robot let us manipulate angles and height at which the sensors work quite easily. This chassis considered the flexibility to mount any load under the axis and this would eventually stabilize the robot. Limitation of time and budget forced us to sacrifice a compact chassis that would keep the wiring on the controller enclosed especially during the competition.



Figure 01: Front View of Robot with Scooper ramp

3.4 – Wheels:

An important component for the movement of our SUMOBOT, are the wheels. Many factors need to be considered when choosing the wheels, including the grip on the surface, the chassis the wheels are attached to, how fast the Robot will be moving and the weight the wheels will be carrying.

Our outer, back wheel material should be lightweight as we want to minimize the impact it has on the final weight of the Robot.

Since our Robot will be attempting to push another object out of the arena, the outer material of our wheels must have high traction on the surface of the arena. The material we choose will need to have a relatively high coefficient of friction when compared to the other materials in its price range. Common materials include:

- Rubber
- Plastic
- Steel

Rubber is a cheap material with a relatively high coefficient of friction (higher than plastic and steel)[10] which makes rubber a good option for our robot.

Rubber is also very light meaning it would not contribute significantly to the total weight of the robot. Steel when compared to rubber or plastic has a higher mass per unit square meter, so this makes steel a bad choice for the two back wheels. The mass of steel (stainless) per cubic meter is 7480 -8000 kg/cu m³. [11]

There are many different types of wheels that can be used depending on the terrain you want to move across.

Omni-Wheels:

Omni-Wheels have three wheels arranged in a triangular pattern, or four arranged at 90 degrees. Omni-wheels have small free-spinning roller wheels on the outside diameter. The central wheel is usually driven. This allows the wheel to be driven forward but slip sideways. This allows for omni-directional movement of the vehicle which is illustrated in Figure p1. [12]



Figure p1: Omni-Wheel

Mecanum:

These currently need two pairs of specific wheels arranged two left and two right. Each wheel pulls the robot at an angle instead of forward and as such, you can get a variety of additional motion above and beyond that of a normal four wheel drive robot [13].

Tri-Star Design:

As can be seen in Figure r1 [15], the Tri-Star design has 3 wheels of “equal diameter about the axis of rotation” in the shape of a Tri-Star. This design allows better vertical terrain traversal, this is due to fact that when one of the 3 wheels stops rotating, the others can continue their rotation due to the pulley system the wheel incorporates as seen in Figure r1 [15].

Figure t1 [15] illustrates how the Tri-star design would go about traversing a vertical obstacle such as a staircase or a deep hole



Figure t1: How a Tri-Star Wheel might move across obstacles and stairs.

The configuration of wheels that we use will also be important since this will directly affect how the robot stands, and how stable the robot is. We will now delve deeper into some of these configurations.



Figure q1: Mecanum Wheel Design [14]

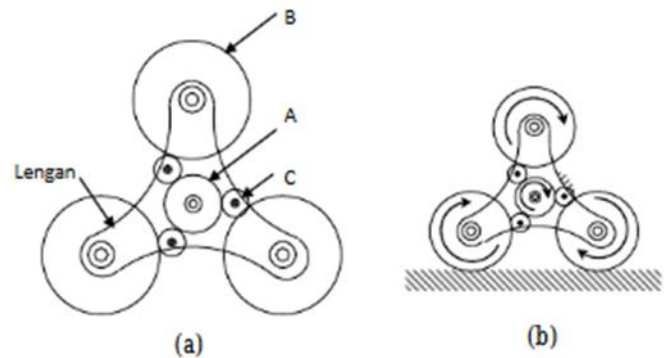


Figure r1: Tri Star Wheel Design

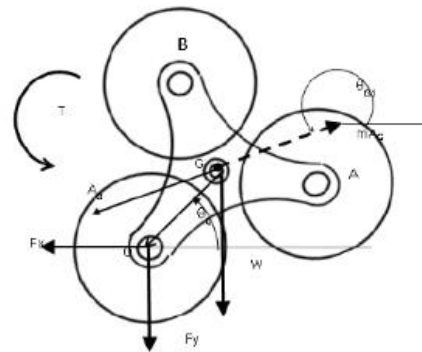


Figure s1: Pulley System of a Tri-star wheel

A popular type of configuration is the “track system” as seen in Figure u1 [16]. This configuration involves linking the wheels on each side together by using a type of gear belt.

This makes the wheels function like a conveyor belt. This allows the robot to make more contact with the surface, increasing the traction significantly more than if the robot were to have the rather simple 2, 4- or 6-wheel configuration.

The disadvantage however of this specific configuration is that since there is a higher area of contact with the surface, there is a higher resistance [17], thus resulting in more power being needed to move the robot compared to have a normal wheel configuration. More powerful motors and a high battery would then need to be considered, which could drive the total cost of the robot higher.



Figure u1: Track Wheel System

Another popular type of configuration is the 2-wheel configuration. This configuration involves using only 2 rubber wheels and an idle wheel acting as a pivot to move the robot around.

The idle wheel can be anything if it is able to move around in multiple directions at any time, so many robots choose to either use a ball caster. A ball caster is a steel ball that is housed in a case that holds it in such a way that the ball can still make contact with the surface and rotate 360 degrees in all directions. Ball casters are made of cheap materials, so tend to be inexpensive, which would be good for our robot since we have a cost limit of £100.



Figure v1: Ball Caster Wheel

The configuration also only uses two wheels, so the price is decreased even more, or more high-quality wheels can be bought. This configuration is very versatile based on the users' needs and would be a good pick for our SUMOBOT. Two wheels suitable for this purpose are the Pololu Wheel Pair from Active Robots. They are plastic with a rubber tyre meaning it has enough traction and will be light to fit the requirements of our robot.



Figure w1: Pololu plastic and rubber wheels

3.5 – Line Detection Sensor:

The goal of a Line Detector is to identify the edge of the arena when moving within the arena. If the white boundary of the arena is detected, the robot should perform a function to move away from the edge. Line Sensors play a vital role in turning any robotic system into an autonomous robot with self-driving, decision making capabilities [18].

The navigation decisions of the sumo robot are controlled using Line Sensors and IR Distance Sensors. Depending on where the robot is in the arena, the next movement will be decided by the microcontroller. If it detects the white edges, it would move back into the black portion of the arena. The arena in this competition is black with a white border. Due to this, we need a Line Sensor that is good at discriminating the difference between white and black surfaces.

The sensor must also be immune to natural light interference as while the competition will be held inside, the big windows inside the lab let lots of light in and could interfere with the operation of the component.

A sensor that satisfied these conditions was SparkFun Accessories Line Follower. This line sensor was very accurate in terms detection according to the data sheet. However not only was it expensive (£29), it was also bulky. Hence, this component is not appropriate for our specific requirements and it has some issues dealing with natural light interference [19].

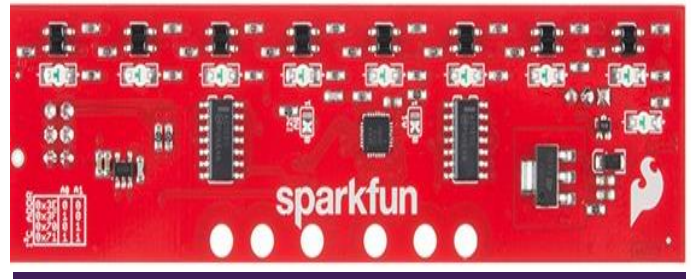


Figure x1: SparkFun Line Follower

Therefore, we need a compact line sensor that is immune to natural light interference. Furthermore, we need two of these line tracking sensors, one for the front and back, so prioritizing price considerations is a focal point of our research.

Another option is a GBB Colour Sensor. This sensor was not only small but can also detect a wide range of visible colours. This sensor powers 4 LEDs, each with 16 photodiodes to read the 4 different light-to-frequency conversions. The colour sensor also has an oscillator which produces a square wave output whose frequency is proportional to the intensity of the chosen colour [20]. Nevertheless, this sensor has a high power consumption because of all these features that are unnecessary for the function of our Robot.



Figure y1: GBB Colour Sensor



Consequently, we decided to go with the Light Tracking Sensor, Gravity series. This sensor was affordable, immune to natural light interference, compact, compatible to our microcontroller platform and according to the datasheet, sensitive to change in surface colours between black and white. This sensor also has an integrated status LED which is extremely helpful for troubleshooting and testing.

The signal provided to the microcontroller with this component is either a '1' or a '0' increasing the ease with which we can program our Robot using this component using conditional statements [21].

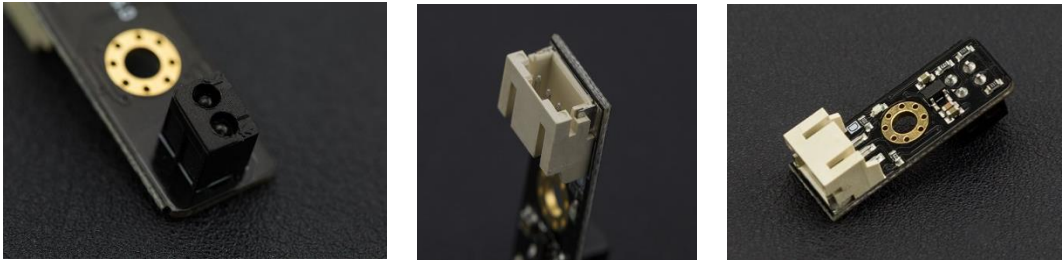


Figure z1: Showing the Transceiver, plug connection and Status LED on top respectively.

3.6 – Object Detection Sensor:

One of the essential requirements of a SUMO-Robot is the ability to detect an object or opponent inside the arena. Taking into consideration the size of the arena (77cm diameter), the object detection method must be short range, responsive and cheap to implement in accordance with our budget constraints. A cursory search of object detection methods for SUMO-Robots found that object detection is generally achieved using Ultrasonic Sensors or IR transceiver sensors [22][23].

A common Ultrasonic Sensor is the HC-SR04, an ultrasonic sensor module designed for use primarily with the Arduino. According to the datasheet [24], the angle of detection is $<15^\circ$ and HC-SR04 modules detect an object by sending 8 pulses of ultrasonic waves and waiting to see if it can detect any pulses back, taking the time to travel there and back for its distance calculation. However, at 15° this angle of detection makes the object location inaccurate and is therefore unsuitable for our design.



Figure a2: HC-SR04 Ultrasonic Sensor

This could be remedied by placing two Ultrasonic Sensors next to each other with overlapping angles of detection, but because of the detection method used we risk encountering interference issues between the two sensors. However, this method of increasing the accuracy of front facing sensors should be incorporated into the final design for the object detection.

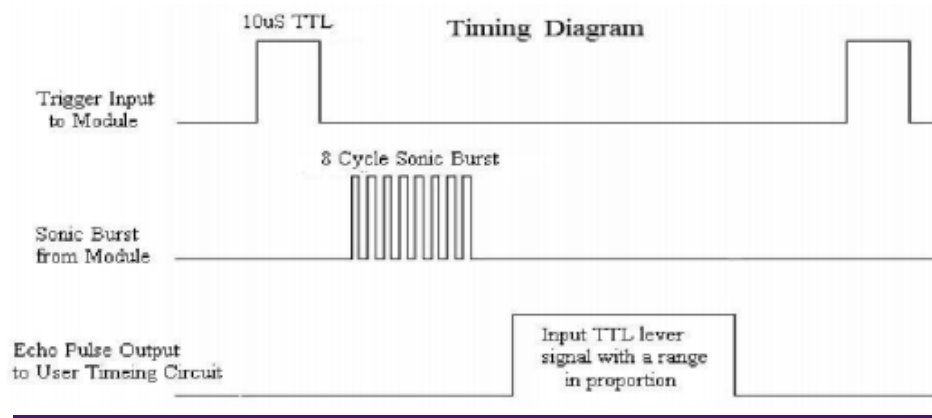


Figure b2: Timing Diagram for HC-SR04 Operation

An alternative method of object detection is the use of IR transceivers. The design of our robot chassis necessitates an IR emitter receiver pair with a variable resistor to calibrate the sensing range.

The component also needs to be self-contained in housing as our robot will be built in stacked layers with the components inside exposed. Collision with other SUMO-Robots is unavoidable and the component must be able to withstand general wear and tear.

There are many common IR sensors that can be used for object detection [25]. One that fits our specification is the SEN0019 by DFRobot. With an adjustable range of 3cm-80cm it is perfect for the size of the Arena [26].

IR sensors do not suffer from the same interference problems that ultrasonic sensors do when placed next to each other, meaning our method of accurate pinpointing of the object location can be used. This sensor is also small, comes pre-installed with an adjustable resistor to calibrate the sensing length as well as an indicator LED to show the status of detection.



Figure c2: SEN0019 IR Transceiver

It is self-contained in housing, durable, light and cheap making this a perfect addition to the SUMO-Robot. It also has easy connection to Arduino, only using one of the digital ports which is essential because of the limited availability of Digital I/O pins on suitably small microcontrollers.

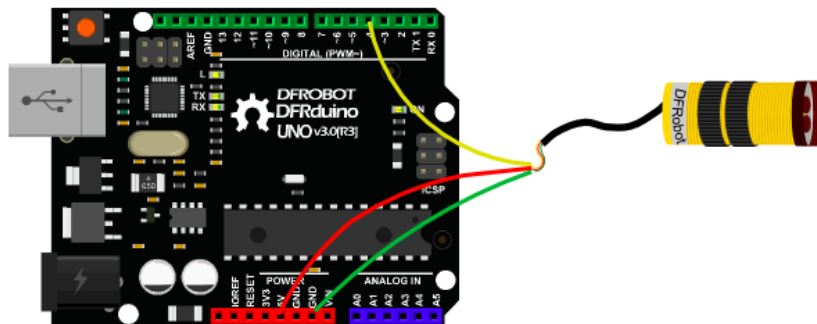


Figure d2: Wiring Diagram of SEN0019 IR Transceiver

3.7.1 – Motors and Actuation:

The motor will be the actuator and driving force for our Robot. For the design choice of the motor, we must consider the strategy that our robot will employ. Whether we want our robot to be fast with less torque, or slower with more torque is a key discussion. This is important as it will be main factor when deciding between motors. For our purpose, which is a SUMO-ROBOT battle, we decided as a group that we wanted our robot to be fast in the arena as this would provide an edge against the other robots. Some of the design choices include we will consider are servo motors, stepper motor and brushed DC motors.

Due to the size specifications of 10x10cm we decided it would be best to use smaller motors to keep within the requirements. Both the motors would have to be less than or equal to 5cm, end to end, to be placed next to each other. Our motor also needs to be light and robust.

One choice of motor could have been a servo motor, however soon into research we realized it would not be a viable option. Primarily because a servo motor is designed for more precise tasks such as moving a robotic arm [27] as the movement is often measured in degrees rather than rpm. These motors are also more expensive than DC motors. An alternative to a servo motor would be to use a stepper motor which is essentially a servo motor that uses a different method of motorization [27].



Figure e2: Servo Motor [29]

Lastly, there is the option of using a Brushed DC motor that runs on Direct Current power. These are good motors to use as they are robust, and they can provide a high output speed of 145rpm with a torque of 700gcm according to the data sheet [28]. There were other, more powerful options available in terms of a higher output speed however we didn't think this was necessary due to the weight of the robot. They require a supply of 6v and this will need to be considered for our battery choice. They will require a motor driver but incorporating a motor driver will increase our ability to control the motors accurately. These motors were also small enough to fit side by side as they had a length of 24.25mm which included the shaft and a width of 10mm [28], so they didn't take up much space.

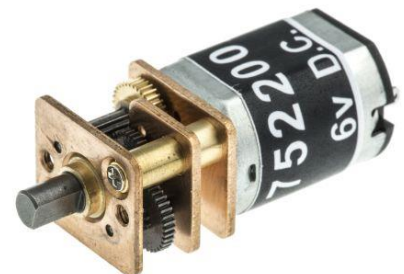


Figure f2: 6v Brushed DC Motor [30]

With the component costing around £10 per motor they were fairly priced when considering the speed and torque, it can provide making this a good design choice for our motors and actuation.

3.7.2 – Motor Driver:

A motor driver is the device that allows for control of the motors [31], it can vary the speed of the motors and change the polarity of the motors. This will allow for pivoting rotation of the robot if either of the motors is set to a lower speed than the other. The Motor driver will also allow the battery to provide a higher current than the motors require.

An important objective in the SUMO-ROBOT project is to be able to move around inside the arena and therefore the motor driver is imperative in the design.

In our initial research we found that there are two main types of devices that could control motor speed. These are motor drivers and motor controllers, however early on we decided we would use a motor driver rather than a motor controller because they are cheaper [32] and would allow us to stay under budget.

A requirement that we set out for ourselves was that the driver had to be small and not take too many pins from the microcontroller. From this we found two options for the motor driver, either the DRV8835 Dual Motor Driver Shield or the L298N Dual Motor Driver.

The DRV8835 Dual Motor Driver Shield is a driver that can control the speed of two motors. It can deliver a continuous current of 1.2A per motor [31]. This driver comes with an Arduino Library [31] which made it easier to program to our requirements. The driver comes packed with protection against over-current and over-temperature [31] meaning we would not have the problem of over-heating and we can use the driver for longer periods of time. The component was also small and has no weight significance, weighing 2.3g [33], and it fit directly on to the microprocessor via digital pins 6,7,8,9 and 10 for its control lines [31].

Based on these advantages we have chosen this driver in our SUMO-ROBOT as this component will help fulfil the user requirements.

The L298N Dual Motor Driver does not fit on to the microprocessor without jumper wires, so this would take up a lot of space on the chassis of the robot. In terms of dimensions (43x43x26mm) this driver is too big and tall for our requirements [34].



Figure g2: DRV8835 Dual Motor Driver Shield [31]

3.8 – Power Supply

The power supply is an important part of the Robot since it has the role of supplying power to each component in the Robot.

The most important factor to consider is the total amount of power the Robot requires to function and the voltage requirements of our components. This would determine the capacity and voltage of the battery we need. We could calculate this by looking at the current and voltage rating of each component on their datasheets.

We need a battery which can supply enough power for the duration of a whole round. We also would want a battery which could last a long session of testing since we would be testing frequently.

Since we do have a weight limit, we would also want the battery to be light, so that we could utilize the weight in more important areas. The size limit of 10x10 would also need to be considered when choosing a battery for the Robot.

Considering the above factors, Lithium-polymer would be the most suitable battery type, since the capacity-weight ratio of Lithium Polymer is generally higher than that of other batteries such as Nickel-Metal Hydroxide, Lithium-ion, and Alkaline batteries. Lithium polymer batteries also tend to be cheaper than other battery types at high capacities.

To have a better idea for what battery would be best for the robot, we did simple calculations to work out the total current draw. These calculations involved totaling up the current rating of each component to see the current that the battery needs to supply at maximum load. It also involved considering the highest voltage rating of all the components. We had to look at the datasheets of all the components for this. The component with the highest voltage rating turned out to be the motors, with a voltage rating of 6 Volts at the highest speed and 3 volts at the lowest.



Figure h2: 9v Duracell Rechargeable Battery



Figure i2: Lithium Polymer Battery



We collected the necessary voltage and current reading from the appropriate datasheets for the components:

Motors: 6v @0.13A MAX 1.5~3v @0.32A MIN

IR Sensors: 5v @100mA

Line Sensors: 3.3~5v @ 20mA

Microcontroller: 5v @40mA per IO pin

$$\text{Current Supply} = \sum (\text{Currents required by component})$$

Since in the case of the battery we would like to take the worst-case scenario and assume that the motors require more current or voltage than necessary. So, we would take the highest current rating:

$$0.32+0.32+0.10+0.10+0.02+0.02+0.04+0.04 = 0.960\text{A and a max voltage of 6 volts}$$

Now that we have a value for the total current, we can choose a battery which is around that amount so that our robot lasts around an hour of continuous use with each component at maximum load.

We also would need to take the necessary precautions when taking care of the battery so that it wouldn't degrade quickly. With Lithium Polymer batteries, the state which you leave the batteries in depend on what state you expect to leave them in. If you are storing them, then they need to be left at a charge that is approximately 3.8v per cell. They can be stored at a higher voltage; however, they aren't as stable, and will degrade faster.

Consequently, we have chosen a Lithium Polymer battery as its efficient, cheap and light, fitting all requirements for our Robot.

3.9 – Entire Sumo Robot Entity:

Now all our components are chosen we can detail how they will fit together. Below is a block diagram outlining the connections between the components.

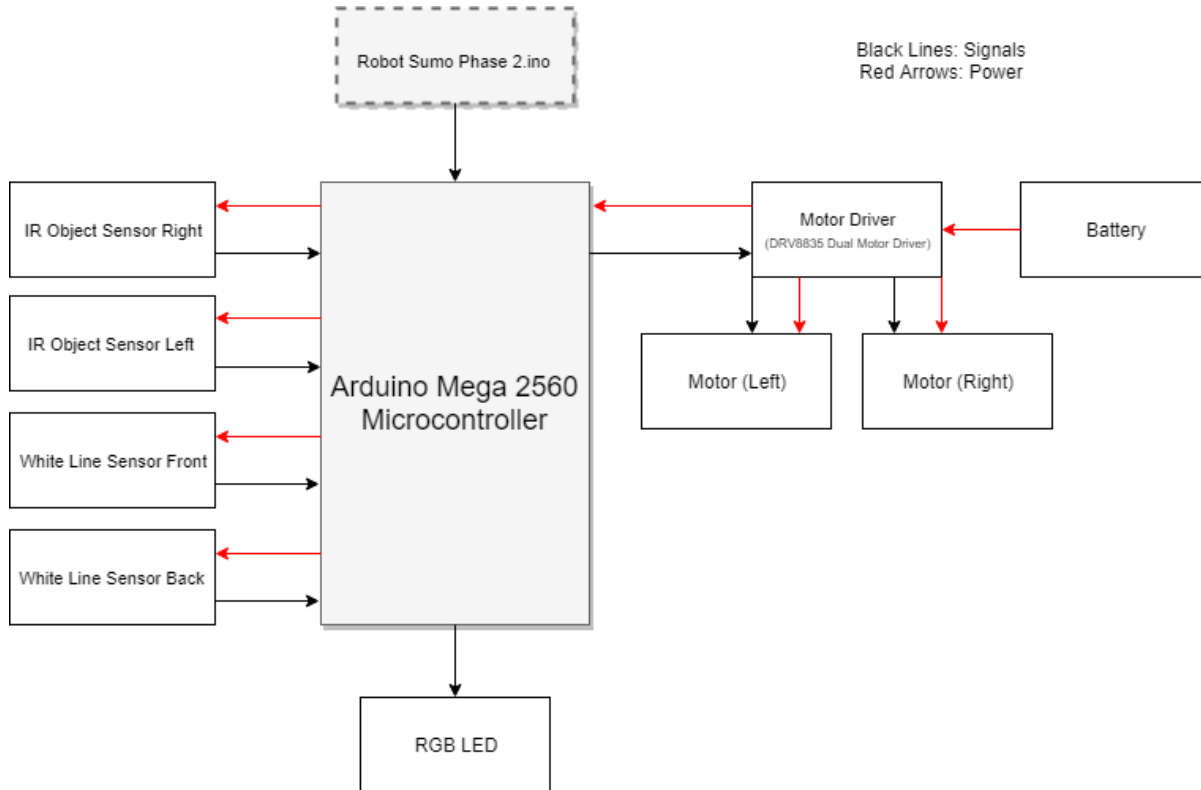


Figure j2: Block Diagram detailing how the components will be connected. Black Lines are signals and red lines are power.

Line Sensors:

The line sensors receive power directly from the 5V output of the Arduino Mega 2560 Microcontroller and utilizes a TTL signal which is either HIGH or LOW depending on whether it has detected a white or black surface [35]. This TTL signal is received by the Arduino Mega 2560 Microcontroller and processed. A decision is then made by the microcontroller based on the value of the TTL signal which will determine what action the motors take.

IR Sensors:

The battery supplies power to the IR Sensors, which sends an electrical pulse which is either HIGH or LOW, depending on whether it has detected an object. The signal is then sent to the Microprocessor via the IO pins, where the microprocessor decodes the analogue signal to digital and sends an analogue signal to the motors via the motor driver.

Motor Driver & Motors:

The motor driver is the only component directly connected to the battery, since it redistributes the power to other components of the SUMOBOT. Since the motors use the PWN pins for their output the motor driver is connected to the se pins instead so that it can control the speed of the motors. The PWM (pulse-width modulated) signal is a signal that the motor requires as input [36]. This pulse is sent by the microcontroller through the pins, and has a specific speed, which is determined by the width of the pulse sent.

Status LED:

This RGB Led is connected using 3 signal pins of the Arduino. Depending on the state that the robot is in the RBG LED will be given different colour values each pin to change the output colour of the LED. This will be outlined in the code design in the next section.

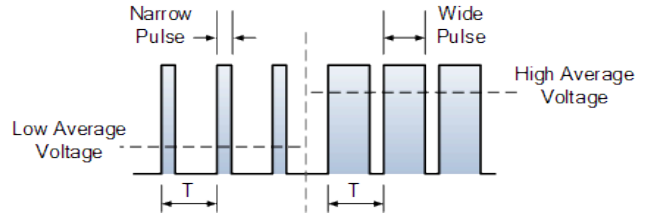


Figure k2: Pulse Width Signal Example

Final Price List:

The table below outlines our costs for each component. Overall our final total is £93.97. This is below the user required budget of £100. Our aim to spend less than the budget has been achieved.

Component	Quantity	Supplier ID	Total Cost
Line Tracking Sensor Module	2	SEN0017	£9.02
Infrared Proximity Sensor	2	SEN0019	£13.75
Brushed 6V DC Geared motor-145rpm	2	7527-2002	£20.86
Dual Motor Driver Shield for Arduino	1	DRV8835	£5.92
Wheel 42x19mm pair (Back wheels)	1	1090	£6.71
Ball Caster with 3/4" Metal Ball	1	955	£3.78
Reely Aluminium Sheets (For shunter) - 400x200x0.5mm	1	50-7395	£3.45
Gens ace LiPo Battery Pack	1	B-25C-1000-2S1P	£10.00
Arduino Mega 2560 R3 Controller Board	1	UK-EL-CB-003	£12.99
3mm Navy Blue Gloss Acrylic sheet -420x297mm	1	3NAVYBLUEACR	£7.49
		Final Cost:	£93.97

Figure I2: Table shows price list for all components used in the Robot.



3.10 – Code Design:

Now we have chosen all our components we can begin to build the program to control the autonomy of the Sumo Robot.

High Level Design:

We chose an Arduino Microcontroller therefore the language we will be using is C++. This language is easy to understand, pick up and troubleshoot, with an exhaustive selection of libraries to choose from to make the coding of the Robot easier. The Arduino also has an IDE called Sketch which is extremely proficient at facilitating the upload of the program to our Microcontroller.

When designing the code, we must take into consideration the priorities of the functions. While the robot does want to win, its priority is not to lose, i.e. fall out of the arena. With this logic we can conclude that the most important state is to avoid the edge of the arena. We must structure the code to take consideration of this fact and ensure that this state has highest priority. All states must have a way of entering the avoid edge state throughout the process to avoid falling out of the arena.

Our robot also scans its environment for opponents through movement of the entire robot. The robot must be able to enter attacking state during turning motion and normal movement of the robot for it to scan the area for objects.

Low Level:

At the low level of design our code must be prepared to interpret signals from the sensors and provide actuation instructions to the motors. Due to the choice of sensors, the signals received by the microcontroller are either a '1' or a '0' for both pairs of sensors. This makes it extremely easy to program conditional statements based on the sensors input. The choice of motor and motor driver allows us to use libraries for the control of the motors making turning and reversing easy to implement.

Code Design:

We can break down the function of our robot into 5 states:

- Standby
- Scanning Loop
- Attack (Object detected)
- Avoid Edge Front (White Line Detected at Front)
- Avoid Edge Back (White Line Detected at Back)

The code must be built to implement these 5 states as functions inside the main program. Calling them from a main loop checking the status of all the sensors.

The following state flow diagram shows expected behavior of the robot with the labels indicating the conditions by which it enters from state 'n' to state 'm'. The colour of the state also represents the colour of the LED while in that state.

Following the structure of this code will allow us to assign the correct hierarchies and state transitions between processes.

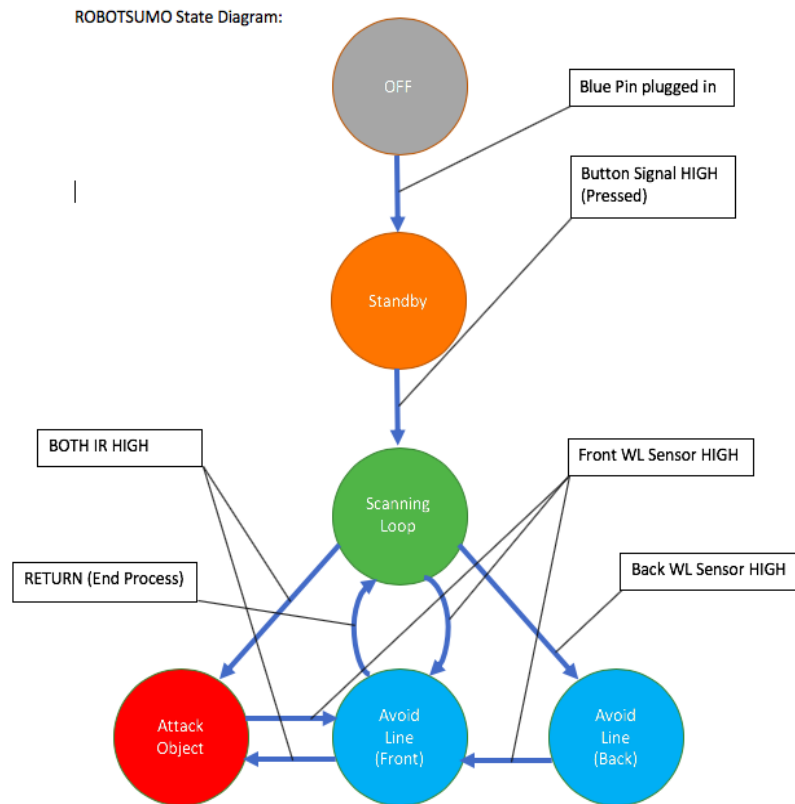


Figure m2: State Diagram showing state transitions between functions of the program.

Now we have the functions mapped out we can write the code.

Code Analysis:

Below is an Explanation of how we built the code to perform the functions we wanted it to do.

```
//Libraries
#include <DRV8835MotorShield.h> // Library for motor control
DRV8835MotorShield motors; // Declaring type motor

const int IRObjct1 = 1; // IR sensor 1 to pin 1.
const int IRObjct2 = 2; // IR sensor 2 to pin 2.
const int LedDisp1 = 12; // IR sensor 1 LED
const int LedDisp2 = 13; // IR sensor 2 LED
const int WLine1 = 3; // White Line Sensor 1
const int WLine2 = 4; // White Line Sensor 2
const int ENABLE = 5; // Enable Button to Start default mode
const int LED_RED = 11; // Status Indicator LED
const int LED_BLUE = 12; // Status Indicator LED
const int LED_GREEN = 13; // Status Indicator LED
const int SPEED = 400; // Defining a constant speed
int redval; // placeholders for RGB values
int greenval;
int blueval;
```

This bit of code includes the library which is used (Motor Driver Library).

This code indicates which pins the components are set to.

//White Line Sensors

```
pinMode(WLine1, INPUT); // Enabling pin 3 as INPUT
pinMode(WLine2, INPUT); // Enabling pin 4 as INPUT
```

The pins that correspond to each component are enabled

```
//Main Loop
void loop() {
//Motors Forward
motors.setM1Speed(SPEED); //search speed
motors.setM2Speed(SPEED);

analogWrite(LED_RED, 0); // LED OFF (Green for scan)
analogWrite(LED_BLUE, 0);
analogWrite(LED_GREEN, 255);

//Sensor IF statements
if(digitalRead(WLine1) == HIGH){ // Operation for Front White Line Sensor
FrontTurn();
}
if(digitalRead(WLine2) == HIGH){ // Condition for Back White Line Sensor
BackTurn();
}
if(digitalRead(IRObjct1) == LOW && digitalRead(IRObjct2) == LOW){
// When both IR sensors are high, call attack function
High --> Attack Function
Push();
}
}
```

The motors are set to speed which is a global variable defined at the top.

The LED is set to green showing that the state of the robot is in scan mode.

These are the conditional statements for other functions to be called, for example if the front white line sensor is activated it calls the FrontTurn() function.

Sumo Robot – Group 5

...

//Function to turn away from line When Front Sensor for White line is activated

```
void FrontTurn() {
  analogWrite(LED_RED, 0); // LED OFF (flashes RED)
  analogWrite(LED_BLUE, 255);
  analogWrite(LED_GREEN, 0);
  motors.flipM1(true);
  // Flipping motors to run backwards and reverse away from white line
  motors.flipM2(true);
  motors.setM1Speed(SPEED);
  motors.setM2Speed(SPEED);
  for(int i = 0; i <= 750; i++){
    delay(1);
    if(digitalRead(WLine2) == HIGH){ // Delay to avoid falling off
      motors.flipM1(false);
      motors.flipM2(false);
    }
  }
  BackTurn();
}
```

- LED is set to blue to indicate that the white line has been detected and the robot is moving away.
- Motors are flipped so the robot can reverse.
- For loop to make sure the motors are flipped in the forward direction again until another function called.

```
BackTurn();
}
// Reverse for a determinate amount of time
motors.flipM1(false); // Flipping motors to run forwards
for(int i = 0; i <= 250; i++){
  motors.setM1Speed(120);
  motors.setM2Speed(120);
  delay(2);
  if(digitalRead(IRObj1) == LOW && digitalRead(IRObj2) == LOW){ // When both IR sensors are
    High --> Attack Function
    motors.flipM2(false);
    Push();
  }
}
motors.flipM2(false);
loop();
//}
```

- One of the motors is flipped so that the robot has one wheel going forward and one in reverse causing a rotation through the central axis

//Function to turn away from line when Back Sensor for White line is activated

```
void BackTurn() {
  analogWrite(LED_RED, 125); // PURPLE to indicate Attack Function
  analogWrite(LED_BLUE, 125);
  analogWrite(LED_GREEN, 0);
  motors.setM1Speed(SPEED);
  motors.setM2Speed(SPEED);
  while(digitalRead(WLine1) == LOW){
    delay(1);
  }
}
```

- This code is executed if white line sensor 2 is activated. The motors are set to their max speed and LED is purple indicating the back white line sensor was used

//Function to push an object out of the arena

```
void Push() {
  analogWrite(LED_RED, 255); // RED to indicate Attack Function
  analogWrite(LED_BLUE, 0);
  analogWrite(LED_GREEN, 0);
  motors.setM1Speed(0);
  motors.setM2Speed(0);
  delay(50);
  motors.setM1Speed(SPEED);
  motors.setM2Speed(SPEED);
  while(digitalRead(WLine1) == LOW){
    delay(1);
  }
  FrontTurn();
}
```

- The LED is red indicating that an object has been detected with the use of the two IR sensors

- The motors are set to maximum speed to ensure that the object detected is pushed



4. Testing and Results

4.1 – Methodology of testing:

With our components chosen and delivered, code designed, and chassis built we now go onto the testing. We will be performing tests on the operation of the components and lastly the functionality of our entire robot with an analysis of our final piece of code. Our tests will be based around the operation of the component as well as the suitability of that component based on our specification and operating environment.

4.2 – White Line Sensor Tests:

High Level Design:

The purpose of this component is to detect the white line boundary of the ROBOT-SUMO arena. There is one sensor situated on the back of the Robot and one situated on the front of the robot to be able to detect the white line from both directions. This component will have to be working under all lighting conditions likely to occur in the Arena area. The component will also have to be situated on the chassis within a functional white line detection range.

Low Level Design:

This component is comprised of a circuit that provides a '1' signal to the microcontroller if the IR transceiver detects the colour white and an integrated LED that lights up under the same conditions. When the microcontroller receives a '1' signal it activates the motors in a backwards direction to move the robot away from the edge of the arena.

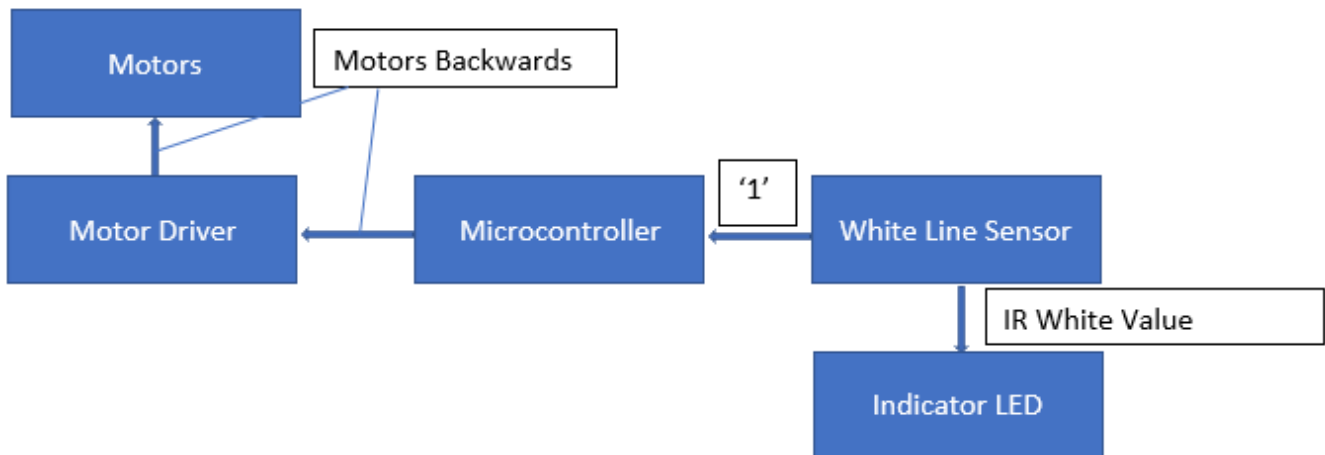


Figure n2: Block Diagram showing the interaction between Front White Line Sensor, the Microcontroller and Motors.

Test 1 – Functionality of White Line Sensor:

Test Goal:

The goal of this test is to check the functionality of the white line sensor and see whether it can provide a '1' signal to the microcontroller when attempting to detect various white surfaces.

Test Method:

The Sensor will be put into a simple circuit connected to the Arduino to process the signal. A program is written to process that signal and show the output of the white line detector circuit on the Serial Monitor of the Arduino Environment. A range of white and black surfaces will be put within 10-20mm of the sensor and we will observe whether a 1 or 0 will be output to the Serial Monitor. The component is expected to provide a '1' to the microcontroller when above a white surface.

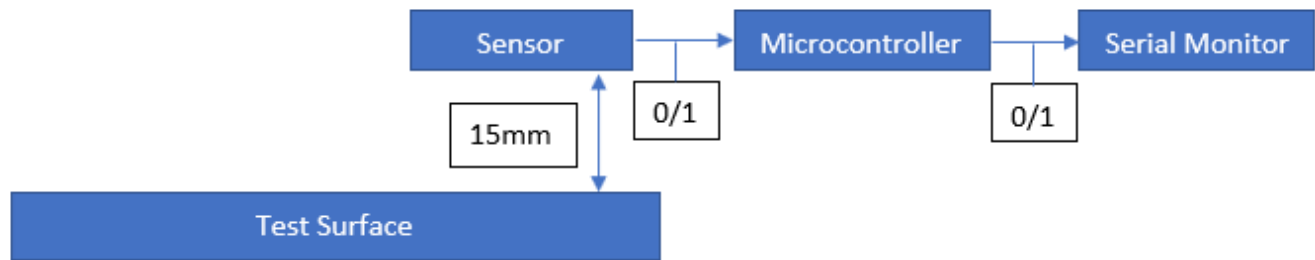


Figure o2: Diagram describing the test setup for Test 1

Results:

Surface	Output to SM	Fail or Success
White Paper	1	Success
White Card	1	Success
White Edge of Arena	1	Success
Printed Paper (Black)	1	Fail
Black Card	0	Success
Black Surface of Arena	0	Success

Figure p2: Table showing different surfaces used to test the White Line Sensor

Discussion:

Generally, the white line sensor outputs a 1 while detecting a white surface and a 0 while detecting a black or dark surface. For the important surfaces (white edge of arena and Black surface of arena), the white line sensor is functional, and the test is considered a success. However, the test failed when trying to use the white line sensor on paper that had been printed black. This could be because the printer ink and the finish on the printer paper has made the black too glossy and reflective, therefore making the transceiver interpret the glossy black as white due to the increased IR reflected by the surface.

Conclusion:

From the results of the test we conclude that this component is working as expected and will be sufficient for operation on the arena surface, however some other surfaces prove to be problematic.

Test 2 – Operation of White Line Sensor under different lighting conditions:

Test goal:

The goal of this test is to see whether the white line sensor will function the same on the arena under different lighting conditions that are possible in the lab.

Test Method:

The sensor will be placed into the same simple Arduino circuit used to test the different black and white surfaces. The program from the previous test is used to output the white line sensor signal to the Serial Monitor of the Arduino Environment. A range of different lighting conditions that occur in the Electronics Lab are tested to see whether a 0 or a 1 will be output to the Serial Monitor when the white line sensor is over the edge of the arena.

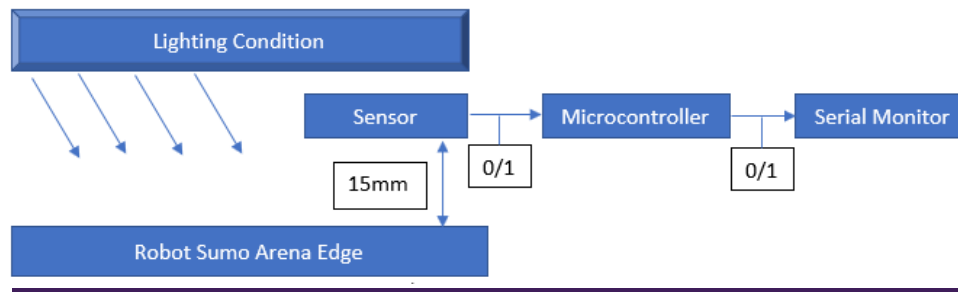


Figure q2: Diagram showing the test setup for test 2

Results:

Lighting	Output to SM	Fail or Success
Day (Lights on, blinds open)	1	Success
Night (Lights on, blinds open)	1	Success
Day (lights off, blinds open)	1	Success
Night (lights off, blinds open)	1	Success
Under a Hood	1	Success
Night (lights off, blinds closed)	1	Success
In Direct Sunlight (Sensing black)	1	Fail

Figure r2: Table showing output of White Line sensor for common lighting conditions in the lab

Discussion:

The White line sensor only fails if the lightning conditions are close enough to pitch black. For the first 4, most probable, conditions the white line sensor provides a '1' to the microcontroller as expected. However, when there is direct sunlight on the black surface of the arena the white line sensor interprets this light interference as the detection of white. This is because the light intensity is greatly increased and even the black surface is reflecting enough light to confuse the sensor.

Conclusion:

From the results of the test we conclude that the rare lighting conditions where the white line sensor fails are implausible and therefore this sensor is suitable for our requirements.

Test 3 – Operation of White Line Sensor at a range of heights:

Test goal:

The goal of this test is to see the functional range of detection with respect to the height of the sensor away from the surface to find the range of heights with which the sensor will have to be situated on the robot.

Test Method:

The sensor will be placed into the same simple Arduino circuit used to test the different black and white surfaces. The program from the previous test is used to output the white line sensor signal to the Serial Monitor of the Arduino Environment. Using a ruler a range of heights from both white a black surfaces are tested to see the functional detection range of the white line sensor. The output of the sensor will be seen on the Serial Monitor of the Arduino Environment.

Results – Table showing output of White Line Sensor at different heights

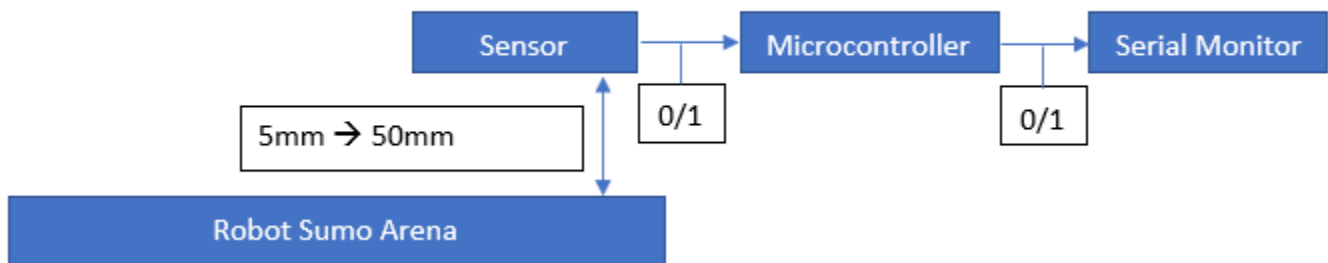


Figure s2: Diagram showing the test setup for test 3

Results:

Height (mm)	Black (Arena)	White (Arena)	Outcome (B)	Outcome (W)
5	1	1	Fail	Success
10	1	1	Fail	Success
15	0	1	Success	Success
20	0	1	Success	Success
25	0	1	Success	Success
30	0	0	Success	Fail
35	0	0	Success	Fail
40	0	0	Success	Fail
45	0	0	Success	Fail
50	0	0	Success	Fail

Figure t2: Table showing output of White Line Sensor at different heights, highlighted is the range of successful operation

Discussion:

The outcome of this test shows that the operating range of our White Line Sensor on the arena surface is between 15-25mm. The data sheet shows that the operating range of this sensor is between 10-20mm [21] but we found that with the arena surface that range is increased slightly.

Conclusion:

The results of this test indicate the height at which the sensor should be placed. Placing the sensor in the middle of this range at 20mm from the floor leaves us well within safe detection range.

4.3 – IR Object Detection Tests:

High level design:

The role of the two IR sensors is to identify an object in the ring below a specific distance.

Low Level design:

At the back of the two IR sensors there are LED lights. An If statement was implemented into the software to activate the sensors, which lights up the LED on the sensor when the sensor has detected an object. When an object is placed in front of the sensor, these output signals are processed to the microcontroller which extracts the information from these signals and processes them. A decision is made and signals are sent to control the motors. As the motors are connected to the wheels these turn and move toward the object at a speed set beforehand. The output produced was the movement of the entire robot chassis toward the object and an attempt to push it off the edge.

Test 1 – IR Sensor output when detecting object at distance x away.

Test goal:

For this test, the goal will be testing if the IR sensors work as intended- detect an object at different distances. The test will be successful if the IR sensors detect an object up to 35cm as that is the distance the Robot will detect opposing Robots from. When the IR sensor detects an object the output is '1' and if no object is detected the IR sensor outputs '0'.

Test method:

The object we used to test if the IR sensors work is a brown box with the dimensions 7x10x7cm (L x W x H) as this will be similar to the dimensions of other Robots during the demonstration. Using a ruler, we put the object at different distances to the IR sensors and recorded the outputs declared by the sensors. The test was repeated 3 times for each sensor to ensure the reliability of results. Figure u2 shows a model of what the IR test looked like (Ring has a 77cm diameter).

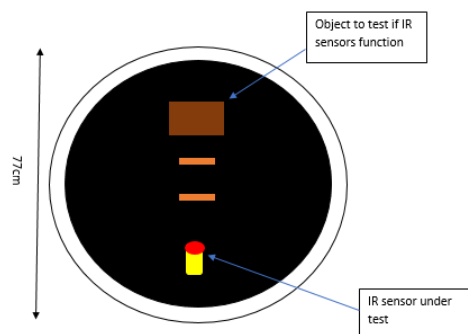


Figure u2: Diagram showing test method

Test results:

Distance away from sensor (measured with a ruler (cm))	IR sensor 1 (1- detects ,0- doesn't detect)			IR sensor 2 (1- detects ,0- doesn't detect)		
	Test 1	Test 2	Test 3	Test 1	Test 2	Test 3
5	1	1	1	1	1	1
10	1	1	1	1	1	1
15	1	1	1	1	1	1
20	1	1	1	1	1	1
25	1	1	1	1	1	1
30	1	1	1	1	1	1
35	1	1	1	1	1	1
40	0	0	0	0	0	0
45	0	0	0	0	0	0

Figure v2: Table of results showing IR outputs at different sensing distances

Conclusions:

From the results produced it can be concluded that the test for each IR sensor was successful. This is because throughout all 3 tests for each IR sensor the output result indicated that the sensor had detected the object at all the distances. As the test was successful more than once the results are reliable and the goal of observing if the IR sensors work has been completed. To take this further, the IR sensors will need to be tested in conjunction with the rest of the components to ensure that inputs to the sensors are accurately processed and communicated to the microprocessor to send signal outputs for the actuators to perform.

Test 2 – Testing the IR sensors with other components

Test goal:

For this test, the goal will be testing if the IR sensors work together with the rest as the components as the design intended- detect an object at different distances causing the movement of the Robot toward that object. Continuing onward from the last test, the test would be successful if the IR sensors detect an object up to 35cm and the Robot moved towards the object and pushed it.

Test method:

Similar to beforehand the object being used is a brown box with the dimensions 7x10x7cm (L x W x H). Using a ruler, we put the object at different distances to the Robot, activated the Robot and recorded what we observed. The Arduino code that we used to test the function of the IR sensors was:

```
" if(digitalRead(IRObjct1) == LOW && digitalRead(IRObjct2) == LOW){ // When both IR
sensors are //High --> Attack Function
  Push(); "
```

The push function declared set the motor speed to 100. We set the speed low so we could observe everything the Robot did once we activated the Robot. The test will be repeated 3 times for each sensor to ensure the reliability of results.
(Ring has a 77cm diameter).

Results:

Distance away from the Robot (measured with a ruler (cm))	Observations		
	Test 1	Test 2	Test 3
5	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box
10	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box
15	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box

20	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box
25	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box
30	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box
35	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box	Both LED lights on the IR sensors activated and the Robot moved toward the box

Figure w2: Table of results showing the outputs of the robot

Conclusion:

From the results produced it can be concluded that the test was successful. At all distances the Robot detected the object and moved toward the object. As the test was successful more than once the results are reliable and the goal of observing if the IR sensors work with the other components has been completed.

The datasheet provided by the manufacturer does not include the angles at which objects will be detected by the IR sensors. To ensure that the Sumo Robot detects opponents accurately the Robot will only go into attack function when both sensors are high. This increases accuracy of detection because when an object activates both sensors it will be directly in front of the Robot as the sensors detection range must overlap.

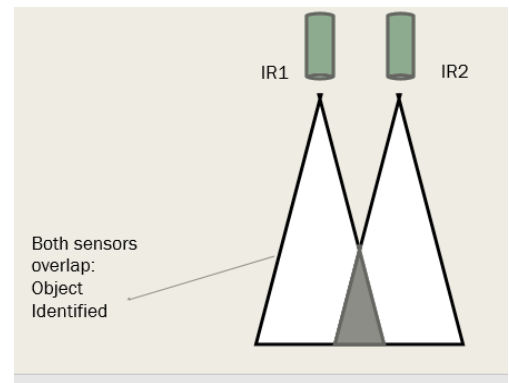


Figure x2: Overlapping sensing angle of IR Sensor



4.4 – Motor Tests:

High level design:

This component is the driving force of the Sumo-Robot, it is the actuator of the robot. This converts the electrical energy supplied from the battery into torque which allows for the robot to move around the arena.

Low level design:

The component is a brushed dc motor made of metal and has a shaft diameter of 3mm. Each wheel is connected to a respective motor M1 & M2 and the motors connect to the motor driver with jumper cables. The power to the motors are supplied via the motor driver as the battery is connected to the motor driver.

Test 1 – Motor function:

Test Goal:

This is a simple test to see whether the motor works when connected to the 9V DC battery supply and wheel.

Test Method:

The wheel is put in tact with the motor through the shaft and the motor is connected to the battery via jumper cables. This was simply to see whether the motor has managed to turn the wheels, if this is the case the respective motor has passed the test.

Test results:

Motor	Wheels turn?	Pass or fail
M1	yes	PASS
M2	yes	PASS

Figure y2: Table showing results of Motor Functional test

Conclusion:

From the results we see that both motors are functioning as they should be, and both have passed the functionality test.

Test 2 – Are both motors working with the same output:

Test Goal:

This test is to see whether both the motors are working with the same speed as it was advertised, this is important as this is how the robot will be able to go in a straight line.

Test Method:

The motors will be connected to the wheels and the battery and put on the chassis so that they can move around the arena. A ruler is put next to the robot and if the robot goes in a straight line then this means both wheels are turning at the same speed meaning that both motors have the same rpm.

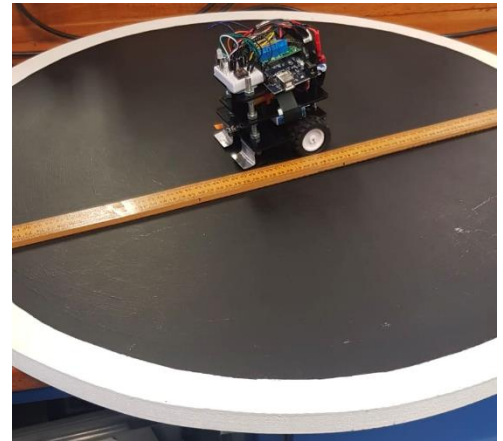


Figure x: Test Setup for rpm test

Test results:

Distance (cm)	Has it gone off course?	Pass or fail
20	No	Pass
40	No	Pass
60	Yes	Fail

Figure z2: Results table showing whether the robot went off course when trying to move in a straight line

Conclusion:

This test shows that the wheels output the same power for the most part of 77cm diameter of the arena. The reasons that the robot has gone off the course of the straight line could be due to discrepancies in power supplied from the battery. Resistance also plays a major factor in this test as if there is a higher resistance, the robot is likely to not go fully straight. Another reason could be due to misalignment of the wheels, resulting in moving off course over long distances.

Test 3 – Forward and reverse direction test:Test goal:

This is to test whether the motors are bi-directional, i.e. whether the motor can achieve a forward and reverse motion.

Test Method:

The motors will be connected to the battery with reversed polarity to test whether the robot moves backwards. The polarity will then be flipped again to test whether it goes forward. It is connected again with the battery the wheels and the chassis.

Test results:

Polarity	Forward or Reverse	Pass or fail
Positive	Forward	Pass
Negative	Reverse	Pass

Figure a3: Results of the polarity test using the motor driver

Conclusion:

We see from the results that the motors have achieved both directions forward and reverse and therefore is as advertised, bi-directional. This will help with the robot as it becomes more mobile being able to go in different directions.

4.5 – Motor Driver Shield Tests:

High level design:

This component is the control mechanism of the motors, without this the motors will always be at consistent speeds. It can vary the speed of the motors and also change the polarity of the motors.

Low level design:

This component connects directly to the Arduino via 5 digital pins-6,7,8,9 and 10, a ground pin, a 5v pin and a Vcc pin. It has a battery input and 2 motor inputs. It supplies all the components including the microprocessor with power as the battery is connected to it.

Test 1 – Driver function test

Test goal:

The objective of this test is to see whether the driver can control the speed outputted by the motors.

Test method:

The driver has an included Arduino library which lets us code the motors. The aim is to upload this code onto the microprocessor with different speed adjustments each time and visually check if the speed of the robot is different.

Test results:

Speed	Noticable Increase	Pass or fail
100	-	Pass
200	yes	Pass
400(max)	Yes	Pass

Figure b3: Results for the Motor driver functional test

Conclusion:

We saw that the code when changed to a high speed resulted in the robot moving faster meaning the motors output had been increased. This shows that the driver has passed its function test. The robot was also going straight meaning that the motors M1 & M2 were coherent.

Test 2 – Turning the robot

Test goal:

To see whether we can get the robot to turn clockwise and anticlockwise, i.e. turning right or left.

Test method:

Change the code so that each motor is set to a different speed causing the robot to pivot.

Example of code:

```
for(int i = 0; i <=400; i++){
    motors.setM1Speed(150);
    motors.setM2Speed(0);
```

This is when motor 1 is set to a high speed and motor 2 is set to 0 meaning the robot will pivot right (clockwise).

If the code is changed to be like this:

```
for(int i = 0; i <=400; i++){
    motors.setM1Speed(0);
    motors.setM2Speed(150);
```

then the robot will pivot left (anticlockwise)

Test results:

Intended Direction	Visual Result	Pass or fail
Clockwise(right)	Pivots right	pass
Anti-clockwise(left)	Pivots left	pass

Figure c3: Results table showing the outcome of the turning test.

Conclusion:

The driver has allowed for us to be able to code the robot into turning left or right, this is very key for the mobility of the robot. This test was a pass overall.

Test 3 – Code & component testing using the motor driver speed control

Test goal:

To check whether functions made on the code are working.

Test method:

Set different speeds on different functions that the robot needs to achieve. This gives us a visual check that the code works. For example, as shown below to test whether the push function was working we set the motors to a lower speed than before. However, this could not be used during the battle so once an LED was added we coded it so that during the push function the LED would be red.

Example:

```
//Function to push an object out of the arena
void Push() {
  analogWrite(LED_RED, 255); // RED to indicate Attack Function
  analogWrite(LED_BLUE, 0);
  analogWrite(LED_GREEN, 0);
  ....

  motors.setM1Speed(250);
  motors.setM2Speed(250);

  if(digitalRead(IRObject1) == LOW && digitalRead(IRObject2) == LOW){
    // When both IR sensors are High --> Attack Function

    Push();
  }

  void Push() {
    motors.setM1Speed(0);
    motors.setM2Speed(0);
    delay(50);
    motors.setM1Speed(100);
    motors.setM2Speed(100);
  }
```

This was a revised test done after the initial test which was lowering the speeds if the function was called.

Initial testing procedure, speed lowered from 250 to 100 once the push function is called.

This shows that initially the motors were set to speed 250, and if the condition that both the IR sensors are high is met, then the Push function is called and the motors are set to a lower speed allowing us to visually see that the motor is slower meaning that the code has been executed and showing that the IR sensors are working.

Test results:

Code Function	Component used	Component working?	Code working?	Pass or fail
Push	IR sensors	yes	yes	<i>pass</i>
Front turn	White line sensor	yes	yes	<i>pass</i>
Back turn	White line sensor	yes	yes	<i>pass</i>

Figure d3: Table of results showing the code and component testing for the Motor driver.

Conclusion:

From the results we can see that the components and the codes that use those components to make a change on the actuator works. These are done visually whether the robot moved faster than its initial speed or lower than its initial speed. As these speeds were observed it means that the code was executed correctly and that the components were working. Whilst working through this test we decided that rather than changing the speeds of the motors to see if a function works, it would be better to add an LED which will represent the function that is being executed with a different colour as mentioned above.

4.6 – Battery Tests:

High level design:

The power supply for our robot has the purpose of supplying the required power to each component of the SUMOBOT to function properly. The power supply we use is a battery of the LiPo (Lithium Polymer) variant and supplies power to primarily the sensors and motors.

Low Level design:

The battery has two main cables, only one of which we use to power the SUMOBOT, the 3-pin charging cable, and the power cable. The power cable of the battery has a female T-Plug connector, so to connect it to the motor driver, we use a male T-Plug to female JST cable, which is connected to a male JST cable. This male JST cable ends with a positive and negative stranded wire which can be screwed in to the motor driver terminals easily. The battery supplies power to the motor driver, which is connected to the Arduino and the motors. Power is supplied to the Arduino (microcontroller) via the motor driver also, which in turn supplies power to the IR Sensors, and Line sensor.

Test 1 – Verifying the voltage of the battery

Test goal:

The first test for the batteries will be measuring their voltage and checking it against the datasheet for them. The goal of this test is to check if the battery can supply the advertised voltage.

Test method:

We used a multimeter to measure the voltage across the battery, by connecting the positive and negative probes to their respective positive and negative connections of the battery. The multimeter then showed the reading of the voltage between the two wires which we recorded.

Test results:

	Test 1	Test 2	Test 3
Reading	ATP MY-64 (V)	Rapid 318 DMM (V)	Cen-Tech 11 Function DMM
1	7.56	7.55	7.56
2	7.56	7.56	7.56
3	7.56	7.55	7.56

Figure e3: Table of results showing voltage readings for the LiPo Battery



Conclusion:

The test shows that the voltage we measured on the battery is 0.16v higher than advertised. This should cause no issues as the voltage already exceeds what is required for our components.

Testing the Power Supply with other components

Test goal:

In this test will be testing the component give the proper output when the battery is supplying power to it. If the component powers up and output be an LED for the two sensors and a sound for the motors.

Test method:

We will power each component individually and observe the output of each component to check if they power up. Each component has their own way to indicate that they work as follows:

- The IR Sensors have red LEDs at the back which will turn on if an object is detected.
- The white line sensor has a blue LED which turns on if a white surface is detected.
- The motors can very clearly be observed to be working as the shaft will start to turn and should increase speed when the value is increased in the code.
- The Arduino (microcontroller) also has multiple LEDs (RX and TX in particular) which light up when the Arduino is powered on

We will then power the SUMOBOT with all the components together and observe whether the battery is able to supply enough current to all the components. We can observe if the components are getting enough current using the same check as above.

Results:

Outputs				
Reading	IR Sensor(s)	Line Sensor(s)	Motor(s)	Microcontroller
1	Both LED lights on the IR Sensors activated.	Both LED lights on the IR Sensors activated.	Shaft rotates and the speed can be changed by changing it within the code.	RX, TX and POWER LEDs light up.
2	Both LED lights on the IR Sensors activated.	Both LED lights on the IR Sensors activated.	Shaft rotates and the speed can be changed by changing it within the code.	RX, TX and POWER LEDs light up.
3	Both LED lights on the IR Sensors activated.	Both LED lights on the IR Sensors activated.	Shaft rotates and the speed can be changed by changing it within the code.	RX, TX and POWER LEDs light up.
4	Both LED lights on the IR Sensors activated.	Both LED lights on the IR Sensors activated.	Shaft rotates and the speed can be changed by changing it within the code.	RX, TX and POWER LEDs light up.

Figure f3: results verifying whether the battery can supply enough current to all the components separately and all together.

Conclusion:

The results show that the battery does supply enough voltage and current to each component individually and in conjunction to one another, which means that the battery is suitable for use in this SUMOBOT.

4.7 – Wheel and Ball Caster Test:

Testing the Wheels and Castor

High level design:

The wheels allow the Robot to move around the arena smoothly while carrying a load.

Low Level design:

The wheels have a shaft which is connected to the shaft of the motors. The motors provide a rotational force, which is converted to rotation of the wheels, which allows the Robot to move. These wheels bear the weight of the SUMOBOT which is 0.5kg alongside the ball caster. The ball caster is drilled into the backside of the bottom layer so that it counteracts the imbalance brought about by the wheels being towards the back. The ball casters also support the weight of the SUMOBOT alongside the back wheels.

Test goal:

In this test, we will be testing the wheels on a variety of different surfaces to test how they perform under different materials.

Test method:

We record the time it takes for the SUMOBOT to travel 0.5m on the material using a stopwatch. We drew a white line on the material to signify the start and finish. The stopwatch started as soon as the SUMOBOT was turned on and stopped as soon as the SUMOBOT passed the finish line.

Results:

Time to travel 0.5m (seconds)				
Reading	Wood (Plywood)	Metal(Aluminium)	Ceramic	Plastic (Acrylic)
1	4.26	4.98	5.12	4.76
2	4.32	4.65	5.64	4.62
3	4.23	4.78	5.98	4.63
4	4.34	4.72	5.45	4.64

Figure g3: Results table showing time to travel 0.5m on various surfaces.

Conclusion:

The results show, as expected, the materials with less coefficient of friction allow the SUMOBOT to travel faster.

4.8 – Functional Testing for the Sumo Robot as a single Entity:

Design:

All the components have been put together and put into three layers. The layers are as follows:

- Layer 1: Motors, wheels and white line sensors.
- Layer 2: Battery and Infrared (IR) sensors.
- Layer 3: Mini Breadboard and Microprocessor.

The Sensors provide the signals to the microcontroller allowing it to make decisions and actuate the motors depending on the signals and how we have programmed the Robot to react to them.

Test 1 – autonomous functionality:

Test Goal:

To see whether all the components can work together to ensure the robot is behaving autonomously and according to our code design.

Test Method:

Put the whole robot in the arena and see whether the robot behaves in the way it has been programmed. For example, does it move back when the white line is reached. An object has also been placed to see whether the push function works. To get our result we will be looking at the colour of the LED to see if it corresponds with the expected colour.

Test Results:

Function test	Hit the white line?	Object Detected	LED colour	Expected colour	Does it go back to the main loop (LED BLUE)	Pass or fail
Push function	No	Yes	Red	Red	Yes	Pass
FrontTurn function	Yes	No	Green	Green	Yes	Pass
BackTurn function	Yes	No	Purple	purple	Yes	Pass

Figure h3: Results table showing the functions of the robot based on its LED indicator status.

Conclusion:

From this test we can see that the robot is aware when there is an object arena, it is aware when it is in the boundary of the arena as it has executed the necessary functions when placed on the arena. All the components have successfully worked together, the input data from the sensors is understood and as an output the respective function is executed. It is doing exactly what the code requires it to do and it also manages to go back to the main loop, so it can go from one state to another state if the conditions are met and therefore it has passed the autonomous functionality test.

Test 2 – finding the right speed of the robotTest goal:

To find the speed of the robot that allows for maximum stability and agility.

Test Method:

The functions will each be set to different speeds and we will visually check whether the robot does any wheelies or any bouncing due to these wheelies. Also, will test at which speed the sensors work as they should be, as if the robot is too fast it may not be able to detect the white or line or the object early enough.

Test Results:

Speed	Object detection	White line detection	Does it do wheelies?	Does it bounce?	Stability level	Pass or Fail
100	Yes	Yes	No	No	High	Pass
200	Yes	Yes	No	No	High	Pass
300	Yes	Yes	Yes	Insignificant	Sufficient	Pass
425	Yes	No	Yes	Significant	Very Low	Fail

Figure i3: Table of results showing the behavior of the robot at differing speeds

Conclusion:

You can see from the results that speeds lower than 425 have insignificant stability and bouncing issues. We have decided to set the speed of the motors to just under 400 as the issues are negligible but we would like our robot to be travelling near to maximum speed.

5. Discussion

In this section of the report we will talk about the challenges we faced in the project, how we overcame them and talk about what our robot did well and didn't do so well, leading onto potential future improvements.

5.1 – Challenges:

5.1.1 – Battery Consumption:

Our first Iteration for Phase 1 used a 9v Battery as the power supply to the components, where we found that it would only supply enough current for the whole robot for a very short time. Even with a reduced power consumption, as the back White Line Sensor was not yet incorporated, we had a massive power deficit. We upgraded our battery to the LiPo version as described in our background research and found that this battery not only supplied more than enough current for all our components to work, it gave our robot extremely long battery life and a steady current supply for most of its battery life. This improvement made testing and troubleshooting much easier as we could test the functionality of our Robot for longer and more exhaustively.

5.1.2 – White Line Sensor Height:

Our Robot was angled slightly downwards towards the front to improve the operation of the IR Distance sensors picking up smaller opponent robots. An unintended consequence of this was the back of the robot being too high for the back White Line sensor to detect the boundaries of the arena. We remedied this situation by remounting the back White line sensor with spacers to bring it closer to the floor. After this change the back White Line sensor operated correctly.

5.1.3 – Weight Distribution and Stability:

Once we had all the components working in tandem at slower speeds, we increased the speed of the robot to test its limitations. At higher speeds we found that the Robot would perform 'Wheelies' (rearing up) when changing direction. At this point the cables for the components were not dressed tightly to the Robot, which we believed to be a contributing cause of the instability. We dressed the cables tightly to the spine of the chassis. This improved the weight distribution, but the rearing was still too pronounced for the robot to be stable towards the edge of the arena, where it would rear up off the side and increase the height of the front White Line Sensor so that it wouldn't be able to detect the edge. For our



second fix we moved the structural rods of the chassis down towards the floor and mounted some smooth feet to the rods using cable clips. This reduced the height of the 'wheelie' and allowed it to stabilize fast enough for the front White Line Sensor to operate in time to keep it in the arena.

5.1.4 – Ball Caster degradation:

During most of the testing of our robot throughout the weeks leading up to the demonstration the ball caster worked perfectly. However, on the day of the competition our Robot suddenly started to move around the arena bouncing up and down oscillating between bouncing off the front and back feet. We decided to remove the ball caster and found that it had become extremely rough and wasn't moving smoothly causing this bouncing oscillation. We tried to use some WD-40 as lubrication but that only fixed the problem temporarily before quickly returning after some time in the arena. Finally, we decided to remove the ball entirely, move the front feet down slightly and have the robot sliding on its front feet to move, which we discovered increased the stability and completely removed any oscillations and bouncing when moving around normally.

5.1.5 – Cable Management:

A minor problem we faced was our cable management from the Arduino to the Breadboard. We made various last-minute changes and thought that the inclusion of a permanent circuit was not something we would be able to complete in time for the competition. Occasionally, after long periods of testing and changes, the signal cables would fall out breaking the circuits of some sensors. Our fix for this, while only temporary, was enough to get through the competition without any issues. It required constant monitoring of the state of the cables to make sure that they were fully plugged in and during the competition we faced no issues. This is something to consider for future improvements.

5.1.6 – Addition of a Scooper:

Our Chassis design was made to the limit of the size dimensions. The material we were going to use for the scooper had a delay in arriving and was eventually delivered on the day of the competition. We found that the addition of a scooper that went across the entire front of the robot would cover the white line sensor and increase the dimensions of the robot so that it was no longer within the requirements of 10cm x 10cm base. We decided to split the scooper into two parts both mounted to the underside of the bottom of the robot. This fix still allowed us to have a scooper but didn't cover the front sensor and in turn didn't bring the robot above the specified dimensions.

5.1.7 – Arduino Upload Timeouts:

A major problem we faced during the functional testing of our robot was the upload of programs to the Arduino from some PC's. The Arduino would fail to communicate with the computer correctly and timeout, preventing us from uploading the program. A troubleshooting search online found that it seems to be a common bug with Arduino microcontrollers for some PC's. The issue was hard to replicate reliably and this caused us to have to just change the upload ports on the PC regularly or just wait some time before trying again.

5.1.8 – Delay(x) function in C++:

When using the delay() function in C++ to control the distance the robot travelled in the arena the microcontroller would not check the sensors for input and the microcontroller would not perform any actions. This resulted in our robot not checking the sensors while rotating which was a pivotal part of our strategy. We fixed this problem by nesting a delay(1) in a for loop of (real delay) length. We wanted the delay to function in a way that the microcontroller would still evaluate the input from the sensors during delay functions.

5.2 – Overview:

Overall, we feel confident that our robot was designed well as we were the undefeated winners of the Sumo-Robot battle. All of user requirements such as size, weight and budget constraints were met. The robot was able to easily identify an object and push it out, it had no problems staying within the borders of the arena and it indicated which state it was in. We feel that the use of the RGB LED was a good addition to our Sumo-Robot as it made troubleshooting a whole lot easier and improved on user interaction. Another positive design option was the use of layers which allowed for easy changes in the robot throughout fabrication. Another component option that was useful was our battery, it was rechargeable and had a port to charge it, so we didn't have to take the battery out to charge it. Some things our robot didn't do so well was that our components were exposed, there was no casing around the robot which made it susceptible to damage. Another disadvantage we had that was the stability of the robot, the robot was heavier on the back side than the front side causing the robot to do wheelies when accelerating. To minimise the number of wheelies we had to limit the speed of the motors and put feet on underside of the robot, meaning its speed could have been increased for even more effective movement if the weight distribution was fixed.



5.3 – Future Improvements:

Due to the time and budget constraints, we compromised on many features both in the software and chassis that we were interested in implementing. We therefore managed to design a fully functional robot that could detect an opponent on an arena and push it at the same time not be thrown off the ring. With more time and an increased budget some improvements that we considered are:

1. White line strip on ramp – Having a white line on the shunter would have made an opponent's line sensor detects “the edge” and would back away making it difficult for the opponents to attack us.
2. Protective casing – Having a shielding outer cover would protect the components from damage during the fight. Also, it would prevent wires from getting caught onto an opponent's chassis and would eventually disable our robot.
3. 4 wheels and 4 motors – This feature would not only increase stability of our robot but also its pushing power and speed. This would give our robot the advantage as it would search the arena faster and push the opponent with much more force.
4. Colour sensor – Although enough research was done on a colour sensor, we didn't implement it because they were too expensive. However, they would have been an improvement as the sensor would be able to be easily calibrated to work with any surface.
5. Larger heat shrinks – Our final design had many stray wires which could have detached during the battles. To prevent this, the wires could have been cut down further and combined in a larger heat shrink to protect them.
6. Adjust search function in code – With our final code the Robot would only initiate the push function if both line sensors were high. An improvement to this would have been to implement a function in the code where if one object detection sensor was high the Robot would turn to that direction and continue the search function. This would increase our chances of finding the opponent and ending the match earlier.
7. Solder a permanent circuit – If we had more time an important improvement would be to make the signal circuits a permanent feature with a circuit board and solder. Our signal wires could sometimes come loose and it was occasionally an issue.



5.4 – Conclusion:

The aim of this Design & Build Group Project was to design and create a SUMO ROBOT that can autonomously move inside an arena without falling off the edge and push opposing objects of the edge. The SUMO ROBOT also had to comply with the user requirements declared. Overall, our project was completed to a high standard; complying with all requirements and rules of ROBOT-SUMO. From the descriptions and explanations outlined in our report, we have demonstrated how we have created a fully functioning and competitive SUMO ROBOT. During the in-lab final presentation, we were able to show our functioning prototype carrying out expected actions. This led to us winning the tournament. Due to modularity of our design, looking to the future there are many improvements that could be made to further develop our SUMO ROBOT.

To be successful at this project we had to work well as a team. This included having good communication, co-operation and giving constructive criticism. As we had different skillsets and previous experience in different fields this helped in our efforts to complete all tasks in areas which each team member is most proficient. In addition, this design problem facilitated each members' advancement leading us all to acquire new transferrable skills and knowledge which will benefit us in the future.

6. Bibliography

References List:

- [1]: Victor, H., Costachioiu, T. and Constantinescu, R. (2013). Building your own sumo robot in some small simple steps - IEEE Conference Publication. [Online] ieeexplore.ieee.org. Available at: <https://ieeexplore.ieee.org/document/6636203> [Accessed 29 Mar. 2019].
- [2]: I. Sarwar, "Advantages and Disadvantages of Using Arduino | Engineer Experiences", Engineer Experiences, 2016. [Online]. Available: <http://engineerexperiences.com/advantages-and-disadvantages.html>. [Accessed: 29-Mar- 2019].
- [3]: Arduino, "Arduino Uno Rev3", [Store.arduino.cc](http://store.arduino.cc), 2019. [Online]. Available: <https://store.arduino.cc/arduino-uno-rev3>. [Accessed: 29- Mar- 2019].
- [4]: Jayat, "Arduino vs Raspberry Pi - Difference between the Two", [Circuitdigest.com](http://circuitdigest.com), 2016. [Online]. Available: <https://circuitdigest.com/article/arduino-vs-raspberryp-pi-difference-between-the-two>. [Accessed: 29- Mar- 2019].
- [5]: LLC Raspberry Pi, "Raspberry Pi 3 Model B+", [Docs-emea.rs-online.com](http://docs-emea.rs-online.com), 2019. [Online]. Available: <https://docs-emea.rs-online.com/webdocs/162c/0900766b8162cdf1.pdf>. [Accessed: 29- Mar- 2019].
- [6]: Arduino, "Arduino - ArduinoMega2560", [Arduino.cc](http://arduino.cc), 2019. [Online]. Available: <https://www.arduino.cc/en/Guide/ArduinoMega2560>. [Accessed: 29- Mar- 2019].
- [7]: S. Electronics, Director, SparkFun Arduino Comparison Guide. [Film]. USA: SparkFun Electronics, 2015.
- [8]: H. Ferdinando and H. Khoswanto, "Design and evaluation of two-wheeled balancing robot chassis: Case study for Lego bricks," in 2011 International Symposium on Innovations in Intelligent Systems and Applications, Istanbul, Turkey, 2011.
- [9]: B. Tognazzini, "First Principles of Interaction Design," Interaction Design Solutions for the Real World, 5th March 2014.
- [10]: "Friction and Friction Coefficients," Engineering Toolbox, 2004. [Online]. Available: https://www.engineeringtoolbox.com/friction-coefficients-d_778.html . [Accessed 28 February 2019].
- [11]: R. Walker, "SI metric," SI metric, 24 February 2016. [Online]. Available: https://www.simetric.co.uk/si_materials.htm. [Accessed 27 February 2019].

- [12]: Contact Rotacaster Wheel Limited. (2017). Omni and Multi Directional Wheels - Rotacaster. [online] Available at: <https://www.rotacaster.com.au/rotacaster-wheels.html> [Accessed 1 Mar. 2019].
- [13]: cbenson, "What Wheels Should My Robot Use?," RobotShop, 17 September 2018. [Online]. Available: <https://www.robotshop.com/community/tutorials/show/what-wheels-should-my-robot-use>. [Accessed 29 February 2019].
- [14]: The Robotics Institute, Ed., "Autonomous Mobile Robots Annual Report 1985, Mobile Robot Laboratory," Autonomous Mobile Robots Annual Report 1985, Mobile Robot Laboratory, pp. 128–129, Feb. 1985. [Accessed 1 Mar. 2019].
- [15]: W. H. P. ., P. Rafiuddin Syam, "Design of Wheeled Mobile Robot with Tri-Star Wheel as Rescue Robot," International Journal of Smart Material and Mechatronics, vol. 1, no. 1, p. 32, 2014. [Accessed 1 Mar. 2019]
- [16]: C. Martin, "Robot Pioneers, Robotics In Law Enforcement," Lucent Press, p. 64, 2018. [Accessed 1 Mar. 2019]
- [17]: cbenson, "What Wheels Should My Robot Use?," RobotShop, 17 September 2018. [Online]. Available: <https://www.robotshop.com/community/tutorials/show/what-wheels-should-my-robot-use>. [Accessed 29 February 2019].
- [18]: S. Patil, "Design and implementation of advanced auto calibrating line following sensor for coloured surfaces with a white line," in 2016 IEEE 1st International Conference on Power Electronics, Intelligent Control and Energy Systems(ICPEICES), Delhi, India, 2016.
- [19]: Datasheet, "Miniature Reflective Object Sensor," Fairchild Semiconductor, [Online]. Available: <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/QRE1113.pdf>. [Accessed 25 February 2019].
- [20]: Datasheet, "TAOS," July 2009. [Online]. Available: <http://image.dfrobot.com/image/data/SEN0101/TCS3200%20TCS3210.pdf>. [Accessed 25th February 2019].
- [21]: Datasheet, "DFROBOT," [Online]. Available: http://www.farnell.com/datasheets/2700147.pdf?_ga=2.184849261.57573029.1553806780-1695445115.1553806780. [Accessed 27th February 2019].
- [22]: Device Plus Editorial Team. (2016, May. 26). All About The Sumo Robot Competition And Technology [Online]. Available: <https://www.deviceplus.com/connect/all-about-the-sumo-robot-competition-and-technology/> [Accessed: 26th February 2019]

- [23]: ahmedazouz. Arduino Sumo Robot [Online]. Available: <https://www.instructables.com/id/How-to-Make-Arduino-Sumo-Robot/> [Accessed: 26th February 2019]
- [24]: ElecFreaks, "Ultrasonic Ranging Module HC-SR04" HC-SR04 Datasheet, N/A [Not Found] <https://cdn.sparkfun.com/datasheets/Sensors/Proximity/HCSR04.pdf> [Accessed: 26th February 2019]
- [25]: D. George. (2014, Oct. 10). Infrared Sensors List Used in Robotic Projects [Online]. Available: <https://www.smashingrobotics.com/infrared-sensors-list-used-in-robotic-projects/> [Accessed: 26th February 2019]
- [26]: DFRobot. (2018, Nov. 29). Adjustable Infrared Sensor Switch (SKU:SEN0019) [Online]. Available: [https://www.dfrobot.com/wiki/index.php/Adjustable_Infrared_Sensor_Switch_\(SKU:SEN0019\)](https://www.dfrobot.com/wiki/index.php/Adjustable_Infrared_Sensor_Switch_(SKU:SEN0019)) [Accessed: 26th February 2019]
- [27]: RS-online, "951D series 12mm dia. SPUR SUB MINIATURE GEARED MOTOR" 752-2002 Datasheet. Available: <https://docs-emea.rs-online.com/webdocs/107d/0900766b8107d25d.pdf>
- [28]: O. RS PRO, "RS PRO, 6 V dc, 700 gcm, Brushed DC Motor, Output Speed 145 rpm | RS Components", Uk.rs-online.com. [Online]. Available: <https://uk.rs-online.com/web/p/dc-geared-motors/7522002/>. [Accessed: 28- Mar- 2019].
- [29]: SG90 Micro Servo Motor. 2019. [Online Image]. Available: <https://www.jsumo.com/sg90-micro-servo-motor>
- [30]: 752-2002 Brushed DC Motor. [Online Image]. Available: <https://uk.rs-online.com/web/p/dc-geared-motors/7522002/>
- [31]: "DRV8835 Dual Motor Driver Shield for Arduino", Cool Components. [Online]. Available: <https://coolcomponents.co.uk/products/drv8835-dual-motor-driver-shield-for-arduino>. [Accessed: 28- Mar- 2019].
- [32]: "Motor Drivers vs. Motor Controllers", Core Electronics, 2017. [Online]. Available: <https://core-electronics.com.au/tutorials/motor-drivers-vs-motor-controllers.html>. [Accessed: 28- Mar- 2019].
- [33]: "Pololu DRV8835 Dual Motor Driver Shield for Arduino", Pololu.com. [Online]. Available: <https://www.pololu.com/product/2511/specs>. [Accessed: 29- Mar- 2019].
- [34]: Instructables.com. [Online]. Available: <https://www.instructables.com/id/Arduino-Modules-L298N-Dual-H-Bridge-Motor-Controll/>. [Accessed: 29- Mar- 2019].



[35]: "SEN0017 - Add-On Board, Line Tracking Sensor Module, Gravity Series, Arduino, Digital Interface," 2019. [Online]. Available: https://uk.farnell.com/df-robot/sen0017/digital-line-tracking-sensor-arduino/dp/2946112?ost=sen0017&ddkey=https%3Aen-GB%2FElement14_United_Kingdom%2Fsearch. [Accessed 12 March 2019].

[36]: "Pololu DRV8835 Dual Motor Driver Shield for Arduino," 2019. [Online]. Available: <https://www.pololu.com/product/2511>. [Accessed 13 March 2019].

END