The University of Manchester

Department of Electrical and Electronic Engineering

Embedded Systems Project 2022-23

DESIGN REPORT #2

Title: Technical Characterisation

Group Number: 4

Group members:	ID Number	I confirm that this is the group's own work.
Muhammad Bin Suratman	10869503	☒
Muhamad Muhamad Asri	10915564	☒
Zhixin Chen	10816322	☒
Asim Zubair	10541694	☒
Doruk Tan Atila	10866352	☒

Tutor: Dr. Laith Danoon
Date: 2 December 2022

Contents

1. Introduction

This Design Report is designed to discuss the work procedures and decisions in achieving its aim, to select the appropriate control methods for running the buggy.

The movements of the buggy will be monitored through a few methods, mainly by the encoders on the tyres and the line sensors. A lab experiment was carried out to characterise the sensors, ultimately determining which sensors will be used to steer the buggy on the track. These sensors will need to be able to differentiate between the dark and white lines efficiently.

Software and hardware controls are the brains of the project. The software section of this report will explain how the system should work generally and determine some systems constraints, while the control section will delve more into the implementation of algorithms to pilot the buggy throughout the race. The buggy will need the most appropriate control algorithm, which is crucial for smoother movement.

As arrangements of the sensors play a significant role, a proposed arrangement of the sensors will also be briefed in the circuit section along with the suitable NUCLEO-F401RE pins to use.

The circuit should affect how the chassis is designed, as the PCB will sit on top of the chassis together with the microcontroller. Simulations are carried out to decide the best material to be used for construction, and which design has the most optimal weight distribution. The result should produce a robust and lightweight buggy.

2. Software

2.1. Functional Summary

The software of the microcontroller will implement algorithms to create the following features:

1. Line detection and line tracking
2. Speed-sensing and speed adjusting
3. Current-sensing and current adjusting

The inputs for these features will be from external circuits such as line sensors and encoders. The microcontroller must read inputs from the external circuits, make the necessary calculations and output the calculated signals to the external circuits. For the buggy to be as fast as possible without overheating, these processes must be as efficient as possible and within optimized time periods.

2.2. List of Possible Software Constraints

- Reaction Speed in the Worst-Case Scenario

The buggy should be able to react to changes quickly to avoid losing control and ultimately crashing. According to Design Report 1, the buggy's speed can be estimated to be around 2.5 m/s. The sensors are designed so that the distance between the two side sensors is 66.8 mm. Assuming the worst conditions of 14 mm line width, 45° turn in 50 mm, and assuming that the buggy is initially 4 mm away from the line's center, the software's maximum iteration time can be calculated:
(*pb*: The buggy's relative position to the center of the line (mm))
(*d*: Displacement on the path (mm))
(*t*: time (ms))

$$pb = 4 + \cos(0.9°) \times d \qquad (1)$$

Speed of 2.5 m/s means:

$$d = t \times 2.5 \qquad (2)$$

Substituting equation 2 to equation 1:

$$pb = 4 + \cos(0.9°) \times \mathrm{t} \times 2.5 \qquad (3)$$

The buggy's relative position to the center of the line must be less than half of the sensors' width:

$$pb = 4 + \cos(0.9°) \times \mathrm{t} \times 2.5 < 33.4 \qquad (4)$$

Solving equation 4 for time:

$$t < 11.8 \text{ ms} \qquad (5)$$

$$f > 84 \text{ Hz} \qquad (6)$$

Since the microcontroller is 84 MHz, with efficient software the buggy will be able to react quick enough.

- Memory of the Microcontroller

Arm Cortex M4 has 500 Mb of code memory and 1 Gb of RAM [2]. The software will be implemented to save only the data that will be used in future and update the old data. The microcontroller's memory will be able to handle both the program and data stores.

2.3. Context Diagram

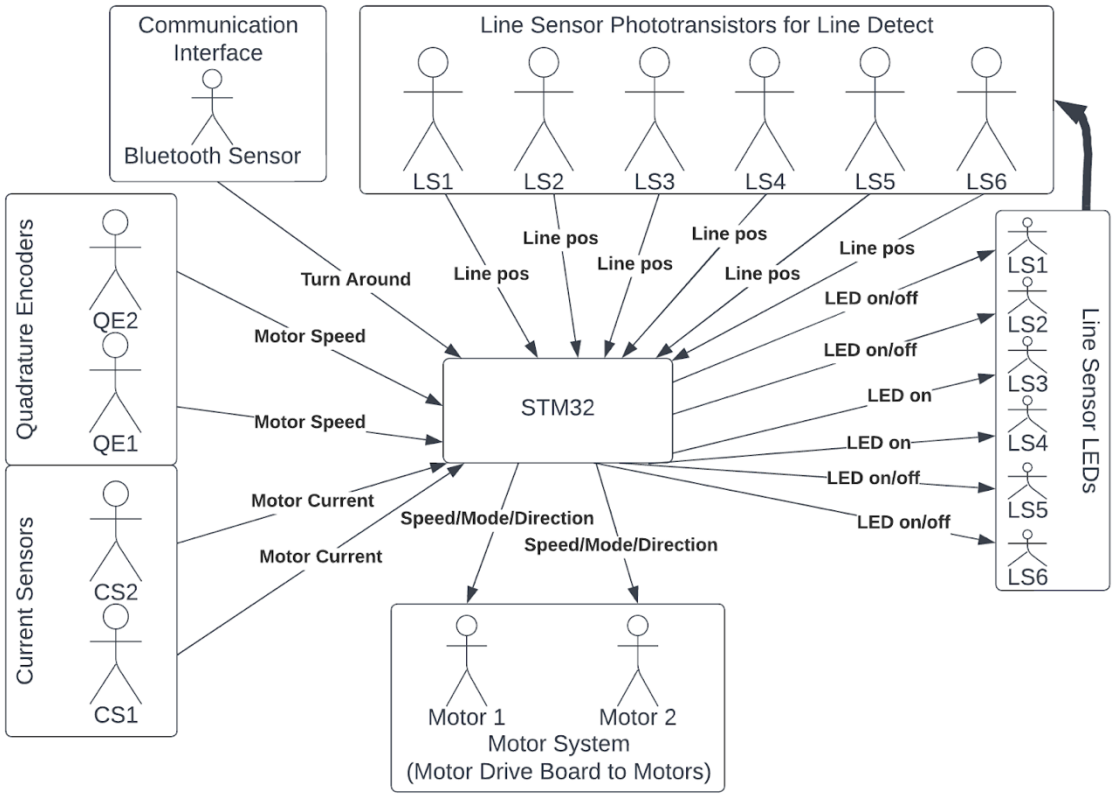

Figure 1 - Context Diagram

2.4. Table of Messages

Table 1 - Table of Messages

Name	Source	Destination	Interface	Signal	Size (Bits)	Arrival Pattern
Line Pos	TCRT5000 Phototransistor	µController	GPIO	Analog	32	Continuous
Turn Around	Bluetooth Module	µController	TBD	Digital	TBD	Conditional
Motor Speed	Quadrature Encoders	µController	GPIO	Digital	TBD	Periodic (f: TBD)
Motor Current	Current Sensors	µController	GPIO	Analog	32	Periodic (f: TBD)
Speed	µController	Motor Drive Board	GPIO	PWM	TBD	Aperiodic
Mode	µController	Motor Drive Board	GPIO	Digital	TBD	Aperiodic
Direction	µController	Motor Drive Board	GPIO	Digital	TBD	Aperiodic
LED on / off	µController	TCRT5000 LED	GPIO	Digital	1	Conditional

2.5. Use Case Diagram

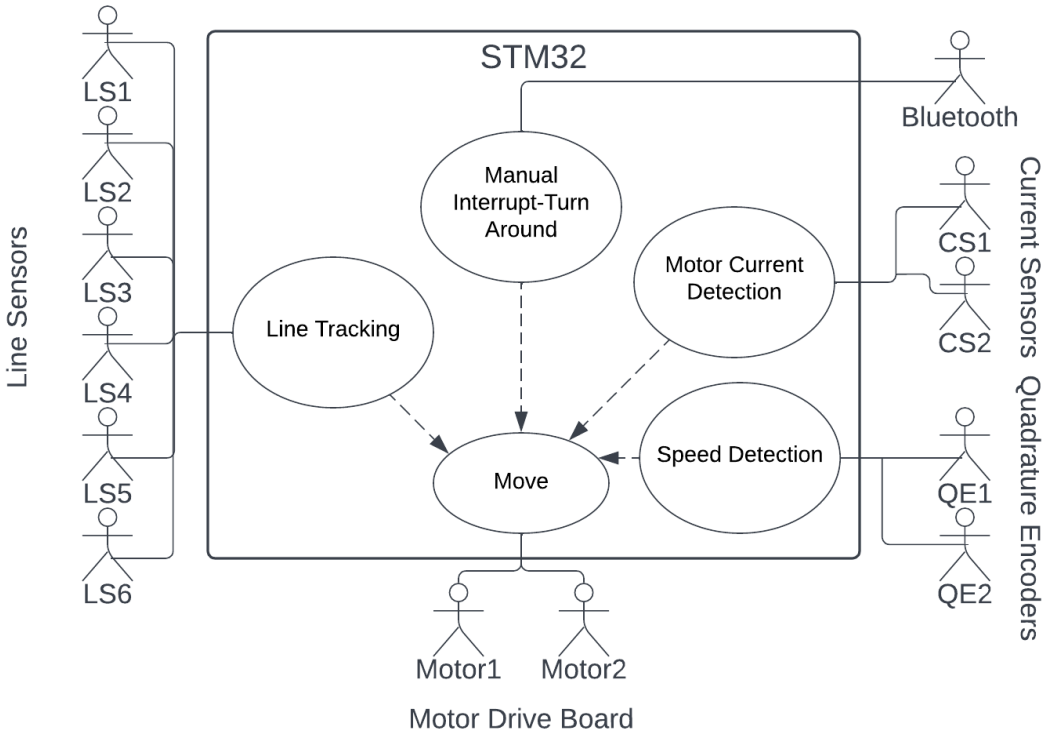

Figure 2 - Use Case Diagram

2.6. Use Case Diagram Descriptions

In the following diagrams, the shapes refer to:

- Algorithm refers to the algorithms executed by the microcontroller's CPU.
- Digital Check refers to conditions to be determined by the algorithms. The result of the checks will determine the next function.
- Peripheral Output refers to predefined functions that will send outputs to the external circuitry. The output sent will either be predetermined or determined by the algorithm.

Use Case 1 - Line Tracking

The most significant feature of the buggy is perhaps the line tracking. This case refers to the line tracking function of the buggy and includes externals:

- Input from line sensors' phototransistors (red)
- Output to line sensors' LED (green)
- Output to the motor drive board (blue)

The software will be designed to conserve power while accurately implementing the PID control algorithm. The following diagram shows how the software will be implemented:

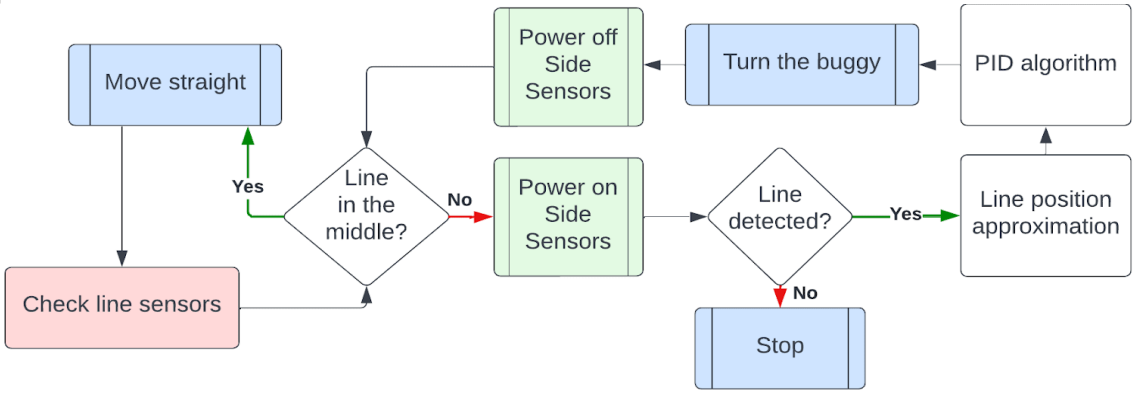

Figure 3 - Line Tracking Use Case Diagram

Use Case 2 - Speed Detection

Speed detection will be used to always keep a constant speed, which will make it easier to rotate and stop quickly. This case refers to the speed-detecting function of the buggy and includes externals:

- Input from quadrature encoders
- Output to the motor drive board

The software will be designed so that the speed sensing takes place within predefined intervals using the ticker API. The following diagram shows how the software will be implemented:

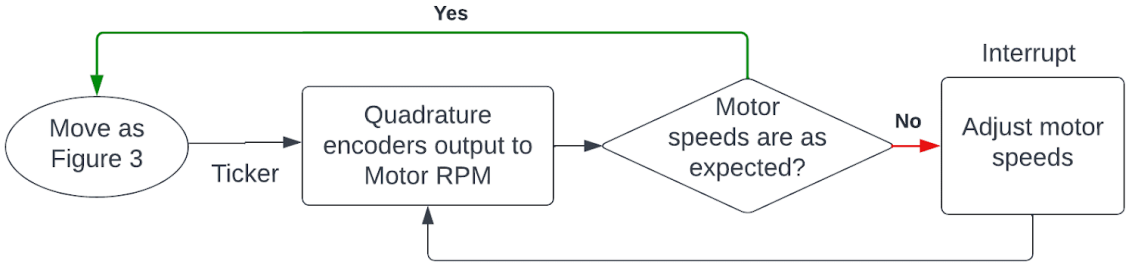

Figure 4 - Speed Detection Use Case Diagram

Use Case 3 - Current Detection

Current detection will be used to prevent overheating and damaging the motors. This case refers to the current function of the buggy and includes externals:

- Input from current sensing circuits provided on the motor drive board
- Output to the motor drive board

The software will be designed so that the current sensing takes place within predefined intervals using the ticker API, it is expected that the current sensing will be done less frequently than speed sensing. The following diagram shows how the software will be implemented:

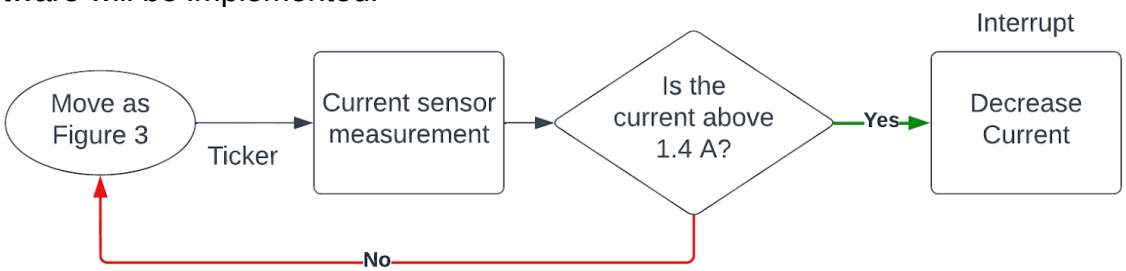

Figure 5 - Current Detection Use Case Diagram

Possible Conflict

It is expected that during inclined planes speed-sensing algorithm will try to increase the speed while the current-sensing algorithm will decrease it. This can be avoided by detecting the inclined planes. Design Report 1 shows that during the inclined planes, the current on the motors significantly drops, which can be used to detect the slopes. When the slope is detected, the buggy might allow exceeding the current limit for a limited time or a different acceptable speed might be used.

Use Case 4 – Manual Interrupt – Turn Around

A Bluetooth signal will be used to interrupt the buggy's regular movement and turn the buggy around. This case refers to the turning around function of the buggy and includes externals:

- Input from Bluetooth sensor
- Output to the motor drive board

The software will be designed so that the Bluetooth input will trigger an interrupt and start the predefined turning around process.

2.7. Object specifications

```
float volatile line_sensor_pos, motor1_Speed, encoder1_reading, current1_reading, current2_reading;
float line_sensor_threshold, min_speed, max_speed, max_current;
Ticker speed_ticker, current_ticker;
void line_detect();                    //Line Detect use case
   void check_line_sensors();
   void power_on_side_LEDs();
   void line_pos_approximate();
   void PID();
void speed_detect();                    //Speed Detect use case
   float encoder1_read();
   float encoder2_read();
void current_detect();                  //Current Detect use case
   float curren1.read();
   float current2.read();
void turn_around();                     //Turn Around use case
```

```
void motor_control();                    //Motor Drive
   void motor1_PWM();
   void motor2_PWM();
   void motor1_control(Speed, Mode, Direction);
   void motor2_control(Speed, Mode, Direction);
int main() {
   InterruptIn BluetoothSignal;                    //Bluetooth as interrupt
   BluetoothSignal.rise(&turn_around);             //Bluetooth triggers turn around
   speed_ticker.attach(&speed_detect, TBD);              //Periodically speed detect
   current_ticker.attach(&current_detect, TBD);    //Periodically current detect
  while(1) {
     line_detect();                          //Motor control is called by all functions
  }
}
```

3. Line Sensor Characterisation

Characterising the line sensors is the basis and the core of the lab. For this project's objectives, choosing the most proper sensor and pairing emitters and detectors are necessary to reject unwanted signals such as ambient light and be sensitive about detecting the edge of the white line.

Table 1 - Characteristics of Components

Components	Wavelength Range/ Peak(nm)	Suggested current for LED (mA)	Beam half angle at half intensity	Type (Emitter/ Detector)
TCRT5000L	940			
TSHG6400 IR LED(a)	850	20	22	Emitter
OVL5521 WHITE LED(d)	470		30	Emitter
TEKT5400S PHOTOTRANSISTOR(g)	920		37	Detector
TSHA4401 IR LED(b)	875	600	20	Emitter
SFH203P PHOTO DIODE (c)	400 to1100		75	Emitter
BPW17N T3/4(f)	825		12	Detector
5M OHM LDR(e)				Detector

Wavelengths, and rejecting unwanted signals such as direct sunlight [5] were the criteria taken into consideration when pairing emitters and detectors. The TCRT5000 includes a sensor with a built-in detector and emitter, a direct daylight-blocking filter, and a wavelength of 940 nm according to the datasheet [10]. Hence, by making the comparison with TCRT5000 sensor, similar wavelengths will be needed for other components. From the data in Table 2, in contrast to the wavelengths of the TCRT 5000 sensor, the emitter must be chosen from (a) TSHG6400 IR LED(850 nm) and (b) TSHA4401 IR LED(875 nm), and the detector must be chosen from (f) BPW17N T3/4 Photo Transistor(825 nm) and (g) TEKT5400S(920 nm). All these chosen emitters and detectors' wavelengths ranges from 800 to 950 nm, according to the spectrum [6], the wavelength belongs to infrared light. It is less affected by the noise from ambient lights [8].

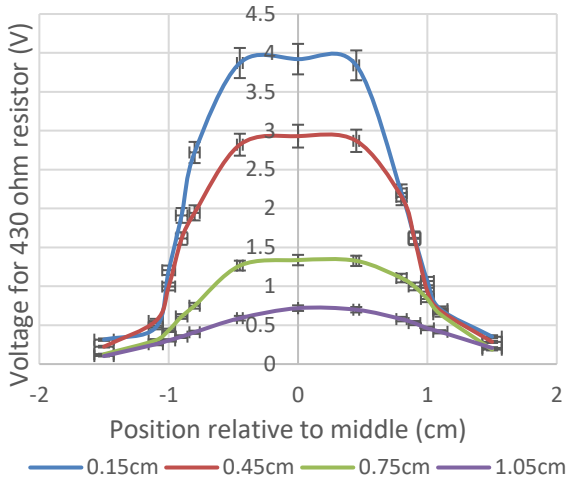

Figure 6 - Voltage against Distance for Sensor 1 (TCRT5000)

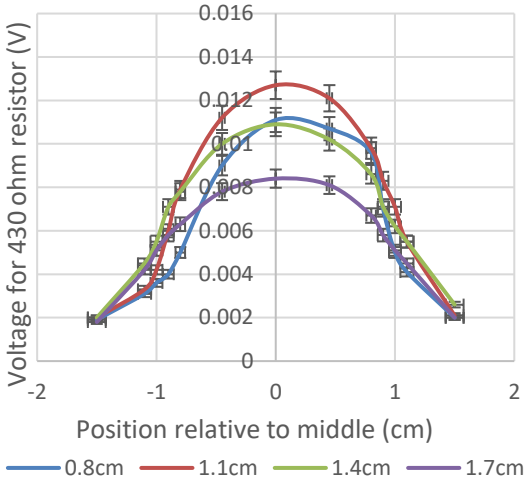

Figure 7 - Voltage against Distance for Sensor 2

(TSHG6400 IR LED & BPW17NT3/4)

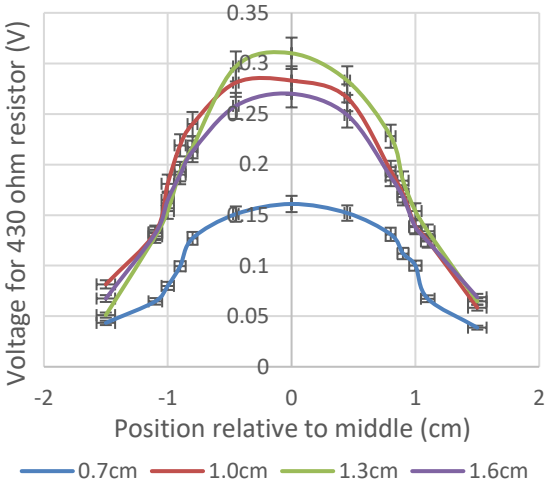

Figure 8 - Voltage against Distance for Sensor 3

(TSHA4401 IR LED & TEKT5400SPHOTOTRANSISTOR)

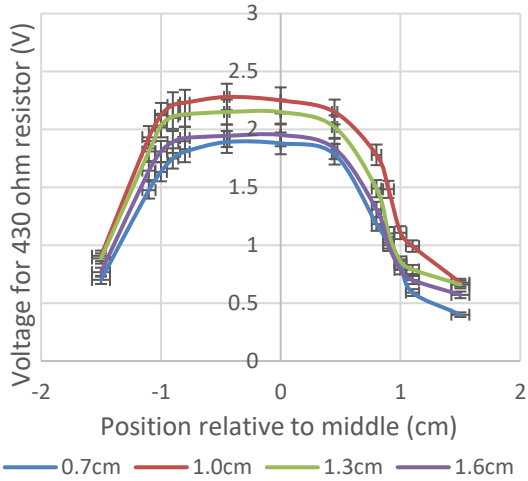

Figure 9 - Voltage against Distance for Sensor 4

(OVL5521 WHITE LED & 5M OHM LDR)

To find how line sensors could detect the white line and choose the most appropriate height for sensors, the line spread function will be used. The line spread function plot shows how much the sensor blurs the edge of the white line. The further the sensor is away from the line, the more it will blur the edge of the line. When moving the PCB at different heights, the sensor position changes from being over the black track area to being over the white line and then on to the black area again. There might be unintentional errors happened during the lab or other conditions which might influence the overall results. With the help of the percentage error bar, those graphs are given a signal of how much uncertainty is in the data under the default of 5% error. The reason for using a percentage error bar is that it includes uncertainty in practical conditions. 5% is acceptable for TCRT5000 sensors to track the white line.

In comparison with the 4 figures above, the Line Spread Function for TCRT5000 is much more sensitive than other sensors. When TCRT5000 sensor is within 0.6 cm

from the middle of the white line, the voltages for the resistor are all close to 4 V, with negligible changes. When the position is more than 0.6 cm from the middle which is close to the edge of the white line (0.7 cm), the voltage drops dramatically. Hence it can precisely detect that the buggy is at the edge of the white line and needs to turn left or right to follow the line again. Therefore, it allows the buggy to keep tracking the white line continuously. However, for other sensors, the voltages for resistor 91 Ω are too low which are all close to 0 V, and change slowly the position of those sensors are changed. If other sensors are used rather than TCRT5000, it would render the buggy unable to check the edge of the white line when there are some changes in the position relative to the middle of the white line such as turns. From Figure 6, TCRT5000 is most sensitive when placed exactly above the white line within 0.15 cm. Therefore, based on these considerations and comparisons, the TCRT5000 sensors should be the sensor used for the buggy and should be put 0.15 cm above the white line.

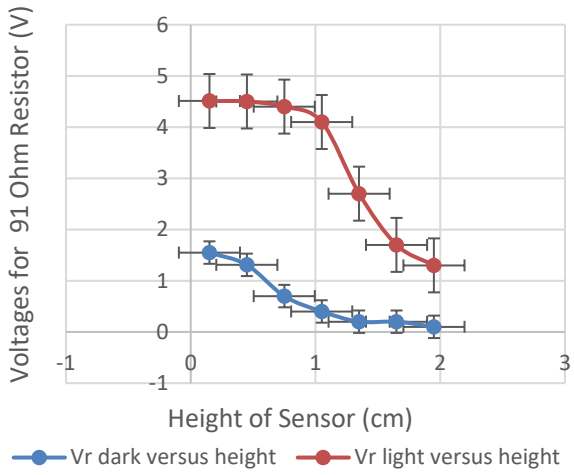

Figure 10 - Voltage against Height for Sensor 1

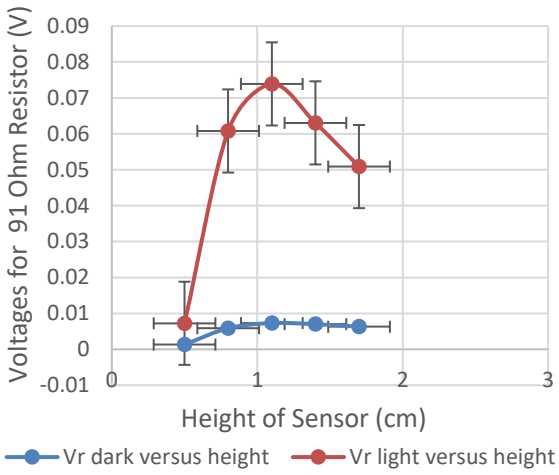

Figure 11 - Voltage against Height for Sensor 2

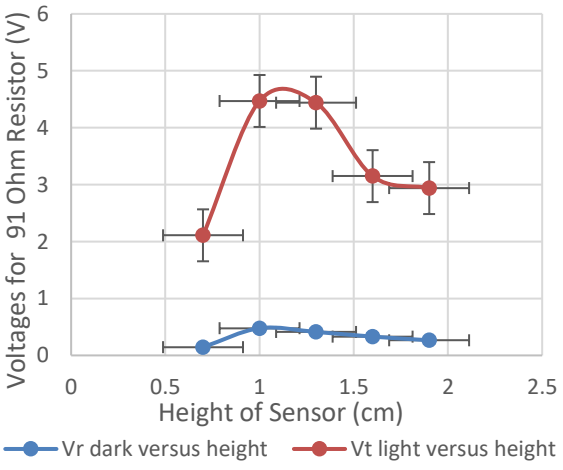

Figure 12 - Voltage against Height for Sensor 3

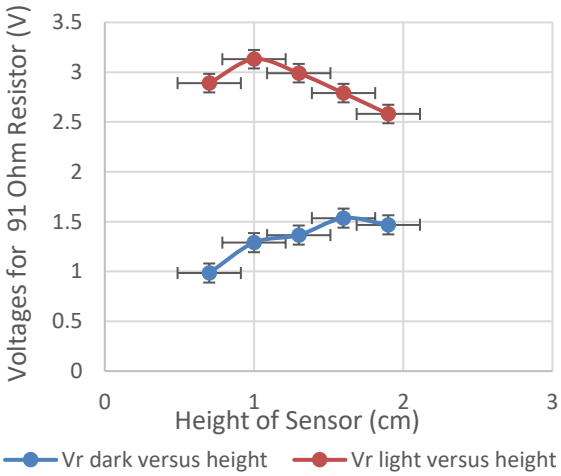

Figure 13 - Voltage against Height for Sensor 4

According to the 4 figures above, there are much higher voltage differences across the TCRT5000 under dark and light conditions at low heights from 0.15 cm to 1.0 cm. When the TCRT5000 sensor is over the white line at low heights, the voltages for 91

Ω resistors are close to 5 V, and less than 1.5 V under the black area. For Sensor 2 (TSHG6400 IR LED & BPW17N T3/4), the voltage differences between dark and light conditions are too small to test the white line and track it. For sensor 3 (TSHA4401 IR LED & TEKT5400SPHOTOTRANSISTOR), even though the voltage differences look significant at heights between 1 cm and 1.3 cm, it faces more ambient light interruption so the sensor might be affected when tracking the white line. But if we use TCRT5000 sensor with a height of 0.15 cm, it means the distance between the sensor and the white line is close enough so that there might be an ignorable effect of ambient light on the TCRT 5000 sensor to track the white line.

In order to make sure whether ambient light would affect the TCRT5000 Sensor, the background illumination will be tested when the LED of TCRT5000 sensor is off. The dark current should also be measured which is the "offset" of a photodetector measured when there is completely no light entering. By measuring the dark current of the TCRT5000's phototransistor by turning its LED off and blocking any light from entering the phototransistor of TCRT5000 sensor, a large resistance (10 $K\Omega$) will be used. As the value of current is too small to be measured accurately, we use the voltage measurement across the LED and then calculate the current by dividing the 10 $K\Omega$ resistance. The voltage for the LED in TCRT5000 sensor is 3.363 V. Hence the dark current will be 33.63 mA.

$$I = \frac{V}{R} = \frac{3.363\ \text{V}}{10\ K\Omega} = 33.63\ \text{mA} \qquad (7)$$

If the TCRT5000 sensor is put under direct sunlight, the voltage will be 0.234 V. Also, if the TCRT5000 is pointed at a bright flashlight that is approximately 1 m away, the voltage will be 0.332 V. As the flashlight belongs to visible light, the same as the sunlight, therefore TCRT5000 is sensitive to ambient light. To overcome this problem and allow the sensor to follow the white line track, the TCRT5000 sensor must be placed on the bottom chassis of the buggy to filter the undesired ambient light which has a significant effect on the signal reception of TCRT sensors.

In practical conditions, there may be some irregularity in the width and reflectivity of the line and track such as line break. From the procedure handbook [4], there is a line break of up to 6 mm. To cope with it, a time *t* is set up assuming the speed of the buggy is 3 m/s.

$$t = \frac{S}{V} = \frac{6\ \text{mm}}{3\ \text{m/s}} = 2\ \text{ms} \qquad (8)$$

Therefore, when TCRT sensors cannot detect the white line, the line break should be ignored for time *t* = 2 ms. As the time is too small, inertia is enough to overcome the short distance. If no other line is detected after time *t* = 2 ms, the buggy will read it as the turn-around point or the start of the track.

4. Circuit Diagram

The circuit diagram implements how the sensors chosen will be connected between one another and to the NUCLEO-F401RE board. The line sensors will be crucial as they will be the eyes of the buggy. Readings from the sensors will be used as inputs to the control algorithm piloting the buggy, hence the readings should be as efficient as possible.

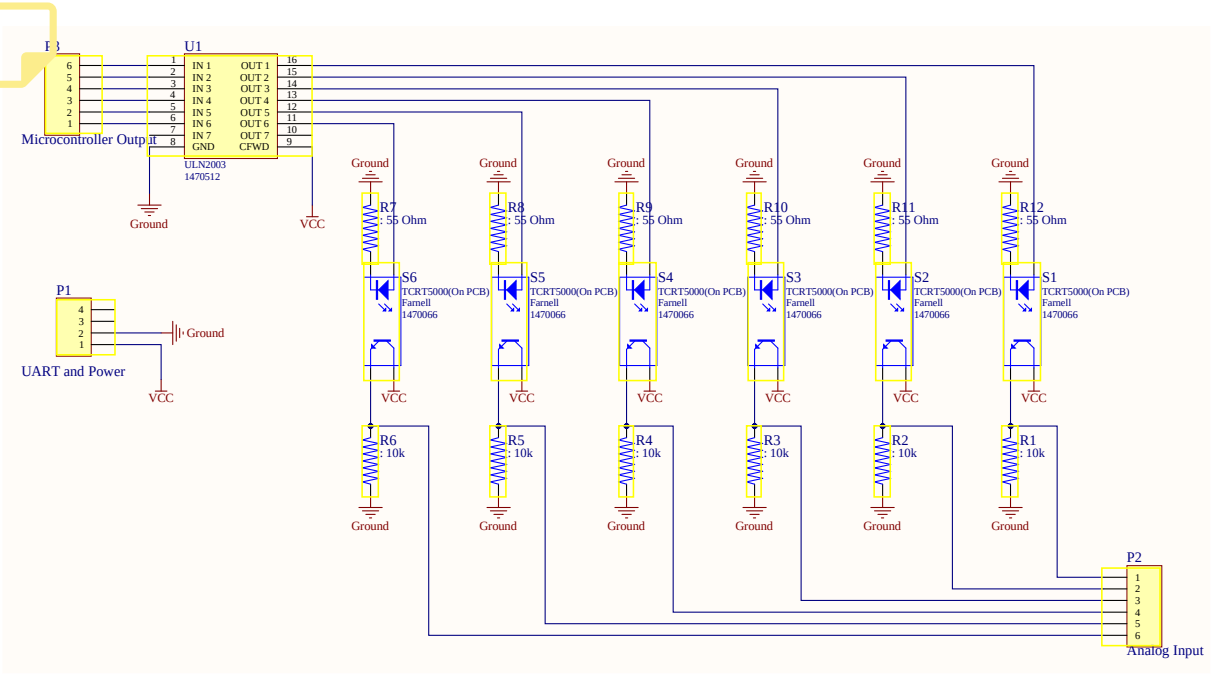

Figure 14 - Schematic Diagram for Line Sensors

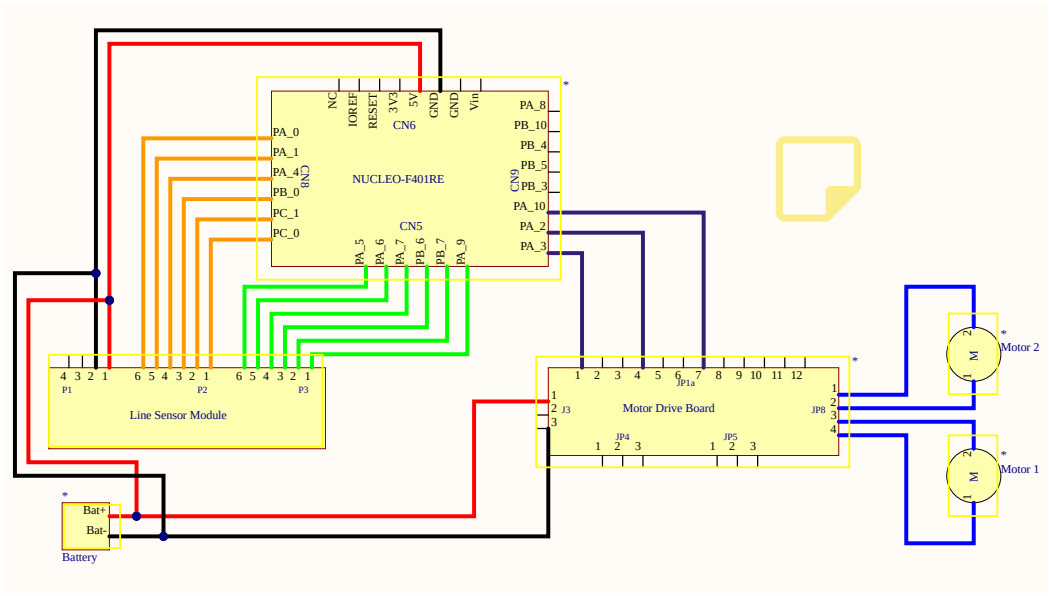

Figure 15 - Wiring Diagram for NUCLEO-F401RE and modules

From figure 14, a Darlington Buffer is connected from the microcontroller output to the LEDs in the sensors, which were then paired with 55 Ω resistors to the ground. This configuration allows for maximum current to flow through the LEDs, so less noise will be picked up by the transistor. Referring to figure 15, it shows how the NUCLEO-F401RE, the motor drive board, and the line sensor PCB board are connected. As the NUCLEO-F401RE contains many connecters [3], only the ones used are shown above. The red and black wires indicate the supply voltage from the battery. Green wires are signals from the microcontroller to the LEDs, and the orange wires are readings from the sensors. The purple wires will carry data to the motor drive board, which will turn the motors through the blue wires.

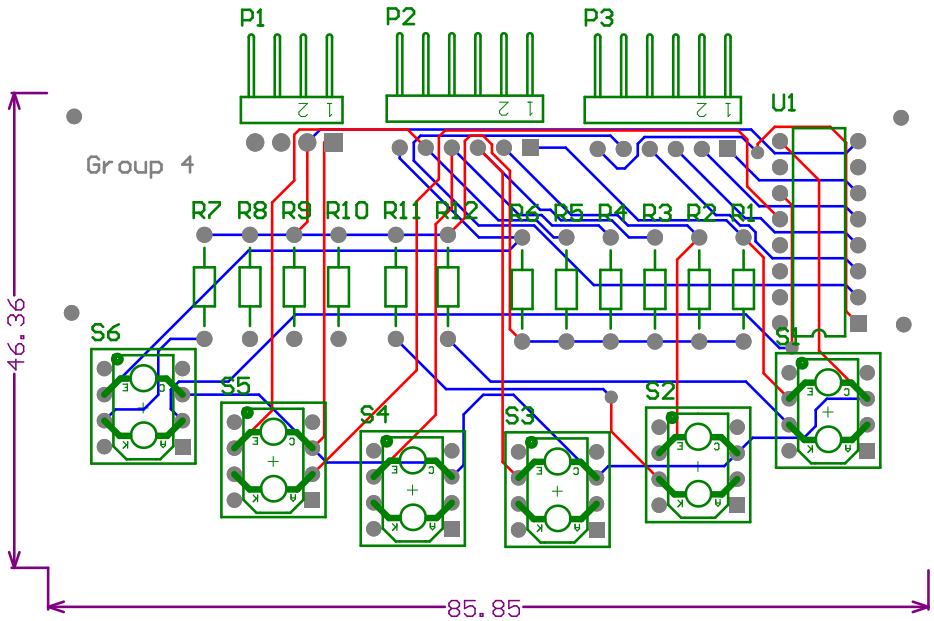

Figure 16 - PCB Layout for Line Sensor Arrangement

The PCB layout above is a reference to how the actual line sensors are positioned. The finalised PCB will have a dimension of 46.36 mm by 85.85 mm. The sensors will be placed at the front of the buggy, with the two middle sensors placed 1.4 cm apart, just shy of the track lines which are 1.8 cm. The sensors in the second and third rows are also distanced 1.4 cm from the sensors in the previous row but at an angle of 0.9°. This is referring to the technical handbook [5], which states that “the white line will not change direction by more than 45° every 50 mm for an angle bend”. This sensor arrangement will allow the curvature of the sensors to be similar to this stipulated track design.

5. Non-Line sensors

Line sensors will detect the white line, but non-linear sensors will aid the control algorithm to successfully follow the line.
Such sensors are encoders, Bluetooth, current sensing sensors and battery monitor. Each of these will be used and implemented with the control software of the buggy. Encoders are sensors that can provide rotational speed of an object that is rotating on a fixed axis. In this case, the encoder will be used to measure the rotational speed of the wheels on the buggy.

The encoders chosen for this purpose are the AEAT-601B-F06 Hall Effect Encoder. These encoders are fitted with the gearbox provided to drive the wheels. Normal encoders such as incremental encoders output just the rotational speed. On the other hand, the AEAT-601B-F06 encoder can also provide the direction in which the rotating wheel is moving towards. This will be needed because the buggy is required to complete the track once and to return to the starting point. The control software needs this characteristic to complete the task described before.

The AEAT-601B-F06 encoder has three outputs: Two Channels (A and B) and one Index. The two channels provide 256 count per revolution and the index provides a single pulse output per revolution [7]. The two channels lead one another meaning that one of the channels is 90° offset of the other depending on the direction of rotation: If channel B is leading then the direction is clockwise, otherwise if channel A

is leading, then the direction is anti-clockwise.

The STM-Nucleo-F401RE has quadrature interface mode on pins TIM2 and TIM5 that count up, count down and direction reading without polling the CPU [5]. To count rotation speed, the software will set up a timer where it starts counting (encoder ticks) and stops the timer when the index outputs signal to indicate a complete revolution of the wheel. This is set up as a loop. The rotation speed will be calculated as follows:

$$Encoder\ Tick\ Rate\ = \frac{(Current\ Encoder\ Ticks - Previous\ Encoder\ Ticks)}{Sample\ Time}$$

$$Wheel\ Velocity = Encoder\ Tick\ Rate\ x\ f(wheel\ diameter,\ gear\ ratio, CPR)$$

Where $f(x, y, z)$ represents a function of the variables x, y and z [5].
Two of these AEAT-601B-F06 encoders will be used for each wheel and their output will be used to compare the expected wheel velocity. The error will be used in the PID control to adjust speed of the baggy.

Bluetooth module (HM-10) will be used to ease setting up the PID controller for the buggy for the following reason: mitigating the problem of connecting the microcontroller via a USB interface to analyse the output and modify the control software. For this project, a Bluetooth Low Energy (BLE) will be used which will link the microcontroller and a computer. The BLE will be configured as slave and the computer will be the master in this link. This configuration will allow the computer to receive and send data to the microcontroller as serial communication. The microcontroller and BLE will have a UART serial port communication where the TX (Transmitter pin) of the microcontroller is connected to the RX (Receiver pin) of the BLE and the RX of the microcontroller is connected to TX of the BLE. This particular BLE has a really low power consumption: in active mode it needs just 8.5 mA and in sleep mode it needs just 50~200 uA. It has a range of operation that goes up to 100 meters [1]. Working frequency is 2.4 GHz which is more than enough for this project.

The current sensing sensor (ACPL-C784) will be used in this project to monitor the output current from the motor drive board to the motor. Two sensors are already integrated on the motor drive board. One for each motor. These two sensors allow the control software to not go over 1.4 A on the motors input thus preventing overheating the motors or damaging them if the current is higher than 1.4 A for a prolonged time. The microcontroller will read the output of these sensors as analogue. This sensor has a reading range of $\pm$200 mV. Which will be amplified and converted in current [5].

The battery sensor (DS2781) will be used to monitor the voltage and accumulated current of the battery . This prevents any inconvenience during testing or technical demonstrations where the project will be marked: if the current level inside the battery is low, it means that the battery needs to be charged. DS2781 will be connected as 1-wire bus communication. This sensor has internal nonvolatile storage which helps reduce the polling required from the software. The sensors can measure from 0 V to 9.9902 V with resolution of 9.76 mV and the voltage register is updated every 440 ms. For current sensing, it measures the voltage drop across the battery which has the range of $\pm$51.2 mV and the current register is updated every 3.515 s [5].

6. Control

The control algorithm is driven by line sensors with the aid of other sensors such as encoders and current sensors.
The white line is detected by line-sensors which are read by the microcontroller as Analogue Inputs.

The measurements from the sensors will determine whether the buggy is above the line or far away in the following way: if the measurements over time are decreasing, it means that the buggy is moving away from the line and if the measurements are constant around 3.9 V then the buggy is above the line.
If the buggy is moving away from the line, then decrease the speed of a wheel to a certain extent till the buggy is above the line again based on the reading from sensors. When the buggy is above the line the speed of both the wheels should be same.
The above procedure resembles a bang-bang control algorithm. It will successfully complete the objectives of the project and it's very easy to implement. At the same time this algorithm is not smooth and will take long time to complete the objectives as it moves from side to side or zig-zag motion.
A proportional (P) algorithm will smooth the process of following the line and decrease the finish time of the objectives. Unfortunately, the proportional algorithm will take a lot of testing and guess work to find out the constant K_p(Proportional gain).
The relationship between the output to the motors and K_p is as follows:

$$u(t) = K_p e(t)$$

where $e(t)$ is the error measurement (i.e., around 3.9 V is set point measurement and error means that a reading is different than the setpoint) [3].
If the proportional gain constant is very high, then the buggy will be very sensitive to the smallest changes in sensor measurements. If the proportional gain is very low, then the buggy will react late to the turns and will end up colliding with the rails of the track thus disqualifying the group.
With just the proportional control, the buggy will end up following the line but with a slight off-set error and not above the line.
I (integral gain) control can solve the problem of steady state off-set but this control requires more parameters thus making it rather slow and the relationship between the output to the motors and T_i is:

$$u(t) = \frac{1}{T_i}\int_0^t e(\tau)d\tau$$

where T_i is the integral time [3].
The derivate control helps with damping the signal of the system. Thus, preventing oscillation in output. Downside is that this control can amplify the noise from the sensor reading such as ambient light noise. The relationship between the output to the motors and T_d is:

$$u(t) = \frac{de}{dt}$$

where T_d is time derivate constant [3].
For this project, PID controller will be used to control the buggy. P control is the fastest control but provides a steady state-offset. The I control solves this steady state-offset but requires a lot of parameters to work with thus more time required. The D control will decrease the time required to damp the signal and prevents big oscillations. These three controls interact with each other to create a control algorithm that can smoothly follow the white line. For the reasons stated above, PID will be used for this project.

The choice of the PID control determined the microcontroller reading set-up of the sensors: analogue in this case. Since the Integral control requires a lot of parameters, analogue output from the sensors would be necessary. And sudden ambient light noise can be prevented with this set up because the reading will jump from low voltage (i.e., 0.9 V) to high voltage (i.e., 3.9 V) in case of noise instead of gradual increase or decrease of the measurement. This will increase the stability of the line-following task, thus increasing the chances of successfully qualifying for the final race day.

All six line-sensors were set up in an arch that follow the maximum curve radius of the track (45°). This can mitigate both the cross talk and some of the ambient light noise because each sensor is on a different position so the ambient light noise will affect them on a different time *t.* The ambient light noise can be prevented by pulling a software interrupt that ignores all readings from the sensors momentarily. The encoders will determine the direction of the buggy and will help determine how much to speed up or slow down on each wheel, thus helping in following the line on turns. The PID control will work as follows:
From figure 3, the sensors 3 and 4 have priority 1 on the PID controller. These two sensors will mainly be used to follow slight curves, thus can make only slight adjustments on the motors and cannot take turns. The sensors 2 and 5 will have priority 2 on the PID controller and can adjust for turns that have radius of turn not greater than 30°.
The sensors 1 and 6 will have priority 3 on the PID controller and can take on turns that have right angle turns. The default output value to the motors is 9 V from the motor drive. Each sensors reading will be multiplied by a negative constant value depending on their priority (i.e., S3 and S4 will have -0.5 constant. S5 and S2 will have -1.5 constant. S1 and S6 will have -2.5 constant). Using the following equation [3]:

$$u(t) = K_p e(t) + K_i \int_0^t e(t^*)dt^* + K_d \frac{de}{dt}$$

The $u(t)$ will be negative thus if we subtract $u(t)$ from default output parameters, then the buggy will be able to have a smooth line following operation.
Line breaks will not be greater than 6 mm, and with the help of an ISR timer and speed calculations, the algorithm will determine whether that is a line break or the end of the track.
Direct sunlight is dealt as follows:
All sensors other than S3 and S4 are not reading. When sensors S3 or S4 outputs are decreasing, only then, depending on which sensor S3 and S4 is decreasing, sensors S5 and S2 will start reading. As soon as sensors S5 and S2 start picking up signals of the line, sensors S1 and S6 will also start reading as well. This process will be gradual and will help mitigate direct sunlight. This will not completely mitigate the threat of such noise but placing the line sensors in such configuration will certainly help.

7. Hardware Overview

Designing a chassis for a buggy is critical in the Embedded Systems Project because it holds all the components together. Consideration is given to several

materials such as Acetal, Glass Reinforced Plastic (GRP) laminate, Aluminium, and Mild Steel for the chassis of our buggy.

The necessity to maintain the buggy weight as little as possible might result in increased battery life and velocity. The chassis must be sturdy enough to withstand the load exerted by all components acting on it.

Stress Analysis

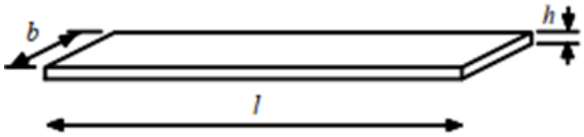

Figure 17 - A Simply Beam [5]

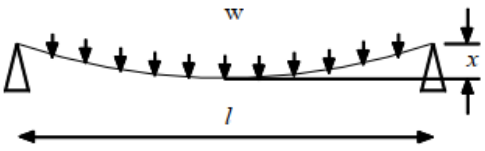

Figure 18 - Distributed Load [5]

A simply supported beam with a distributed load of w (Nm^{-1}), length l, height h and breadth b has a deflection shown in figure 18. Distances are in m [5].

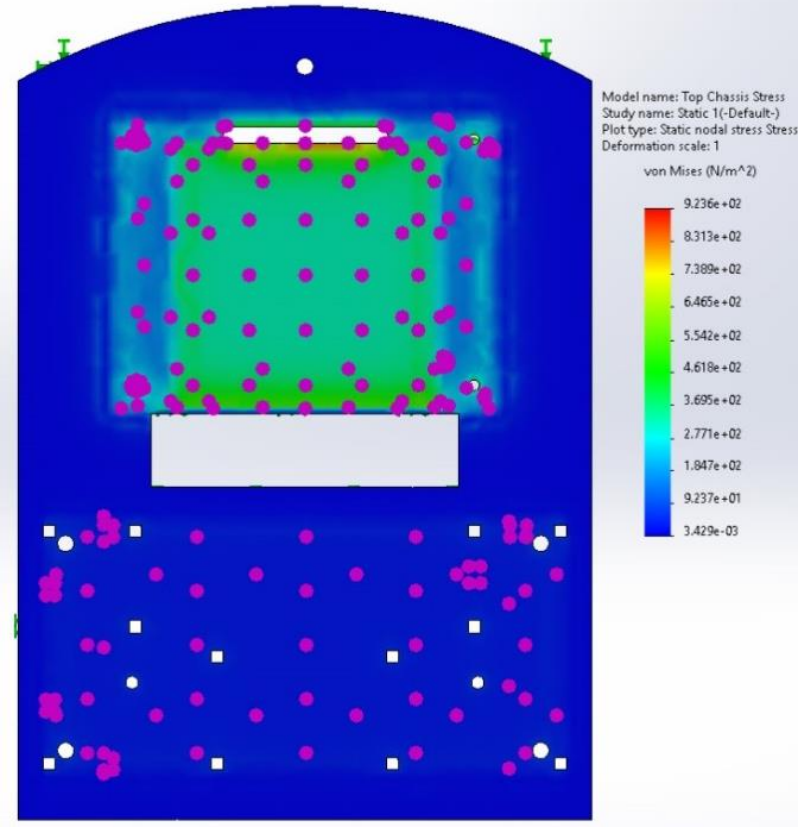

Figure 19 - Stress Analysis for Top Chassis

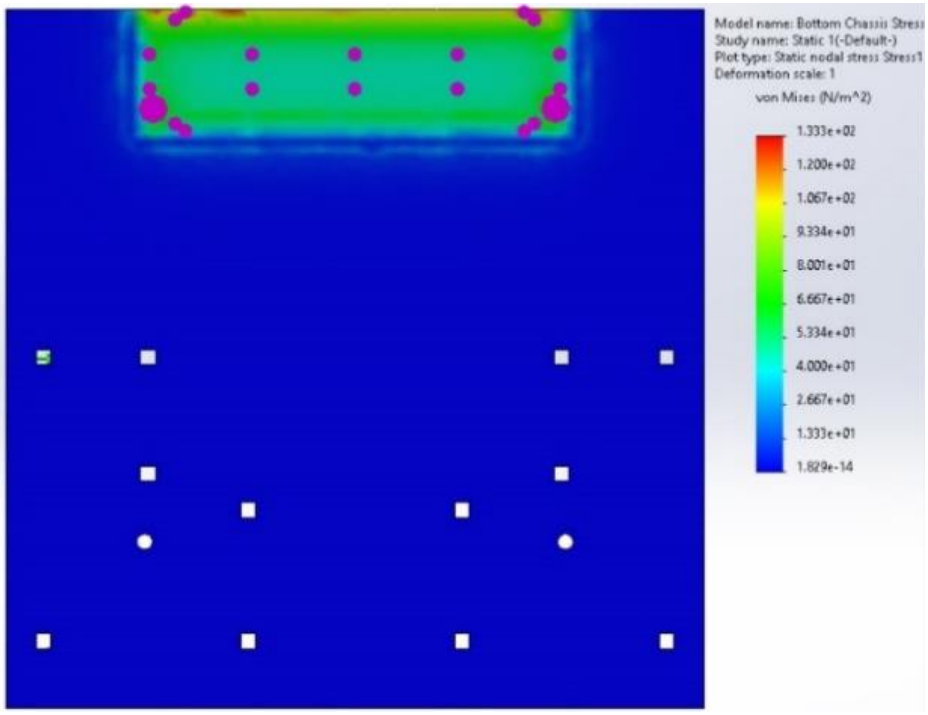

Figure 20 - Stress Analysis for Bottom Chassis

The stress analysis of the top and bottom chassis is shown in Figures 19 and 20, respectively. Attached components exert static force on the chassis. The von Mises scale's stress analysis provides a thorough perspective of the chassis. Nonetheless, the force is so little that the simulation shows no deformation of the chassis. The maximum vertical deflection of the chassis when loaded with components shall not exceed 1.096 mm. This can be estimated by dividing the chassis length by 180 [4].

Ease of Manufacture

Submission of the document in the form of a DXF file is required for simplicity of fabrication. In the DXF file, just the outlines and cutting lines of the chassis are required, which must be in 'Normal To' view rather than isometric [4]. By looking at the laser cutting workshop specifications, one may estimate the tolerance of the cut and determine the inaccuracy.

Choice of Materials

Table 3 – Comparison of Materials [5]

	Acetal	Glass Reinforced Plastic (GRP) laminate	Aluminium	Mild Steel
Cost (£)	55.68	122.00	19.50	18.00
Dimension (mm)	500 x 500 x 3	500 x 500 x 3	500 x 500 x 3	500 x 500 x 3
Density (Mg/m^3)	1.8	1.9	2.7	7.8
Young's modulus (GPa)	2.8	11.5	69	207
Ultimate Tensile Stress (MPa)	67	175	310	414

According to table 3, mild steel is the least expensive when compared to the others with the same dimensions. Mild steel has the greatest Young's modulus and barely minor changes form under elastic strain. Furthermore, mild steel has the greatest ultimate tensile stress. This indicates that mild steel can sustain severe stretching stresses without breaking.

Acetal, on the other hand, has the lowest density of all. The chassis must be as light as appropriate for the specific purpose of this project. It is also easy to laser cut Acetal into the correct form. As a result, the chassis construction process will be sped up.

Layout of Components

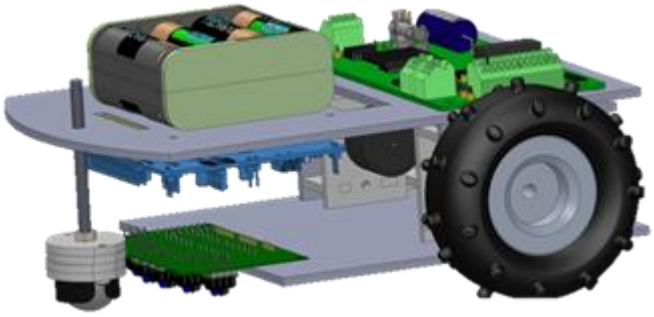

Figure 21 - Layout of Buggy

The arrangement of the component on the buggy is depicted in Figure 21. All the associated components will put strain on the chassis. The placement of the component can have an impact on the buggy's stability.

Table 24 - Component Mass and Placement

Component	Mass (g)	Placement on Buggy
Battery	265	At the front end of top chassis
Nucleo-F401RE + UM1601	108	At the front end of top chassis
Castor	30	At the front end of top chassis
Motor + Gearbox + Encoder + Wheel	456	At the rear end of both chassis
Motor Drive Board	53	At the rear end of top chassis
Line Sensor PCB	40	At the front end of bottom chassis

The mass and arrangement of components are presented in table 4. The mass of the components in the front (453 g) and rear (509 g) end should be the same. It must be placed at the opposite end of the chassis so that the load distribution is more balanced. The weights of the components may be found in the technical handbook [3].

Dimensioned 2D Drawings

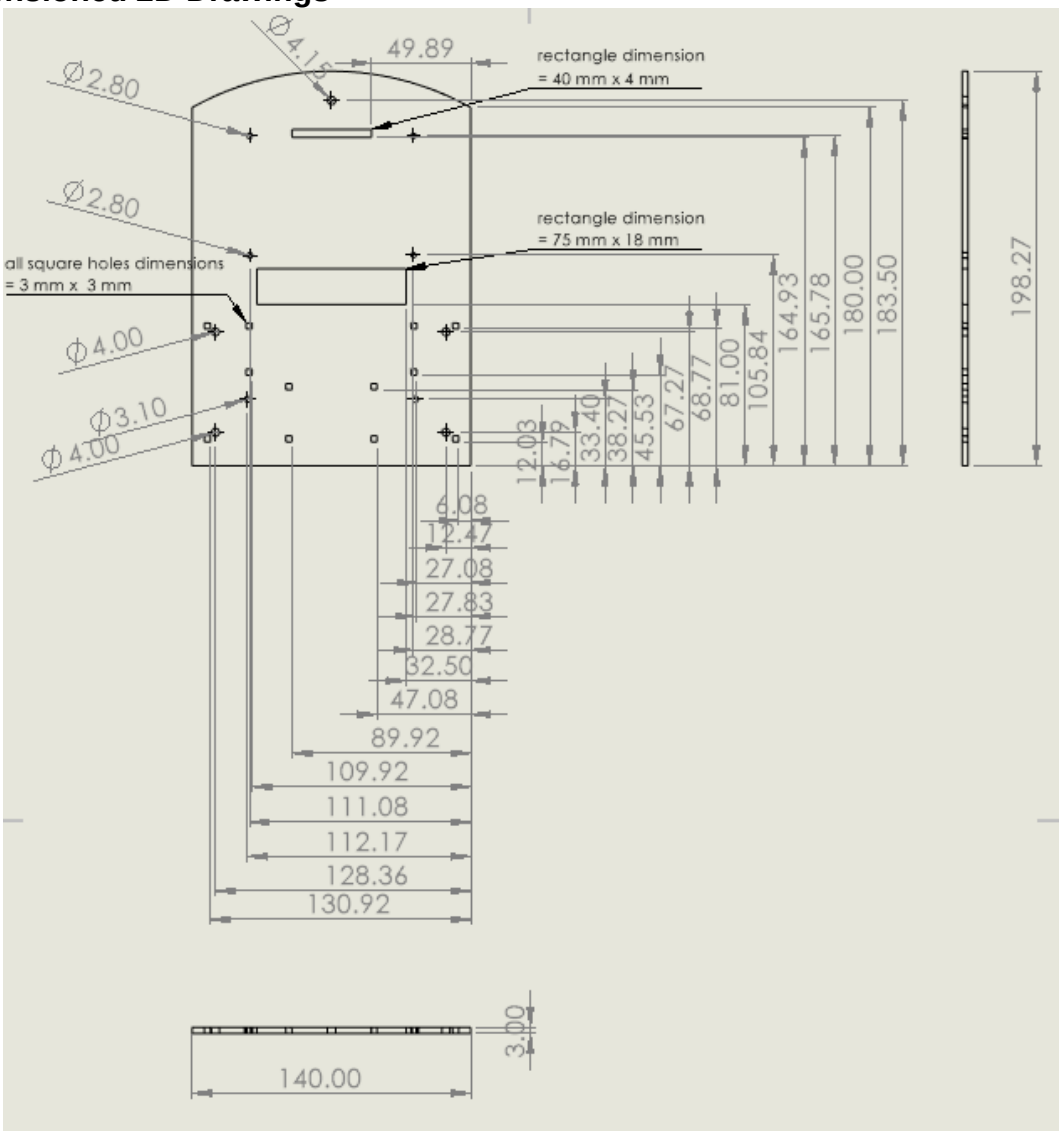

Figure 22 - 2D Drawing of Top Chassis

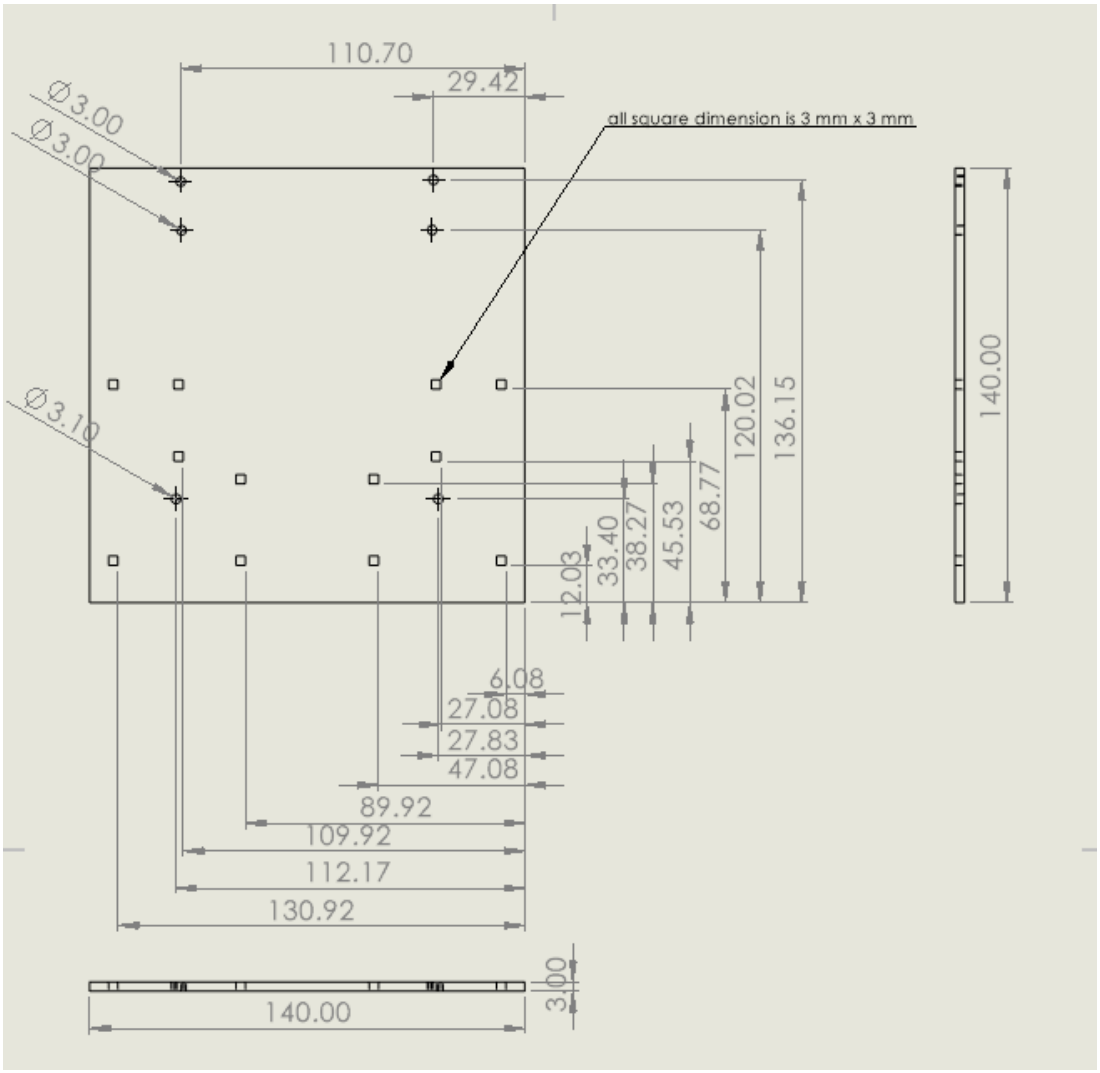

Figure 23 - 2D Drawing of Bottom Chassis

Figures 22 and 23 illustrate the dimensioned 2D drawings of the chassis in relation to the top and bottom chassis, respectively. The axle of the rear wheels should be positioned 55 mm from the rear end of the chassis, so that the buggy may rotate when there is no line without hitting the track wall. The centre of the castor hole is located 183.5 mm from the rear end of the chassis.

Gearboxes would be installed by putting the section of the gearbox that protrudes into the square cuts on both the top and bottom chassis and screwing them together. The curve component of the chassis has a radius of 143.27 mm and a centre location of 55 mm from the rear end.

Response of Buggy

Wheel location is important with our buggy. Wheel positions must be parallel to each other and at the rear end of the chassis. When both wheels are near together, the buggy is more likely to lose its balance than when the wheels are spaced apart appropriately. In contrast, if the wheels are too widely apart, the chassis will easily deform.

The spacing between the centre points of two consecutive TCRT500 sensors on the line sensor printed circuit board (PCB) is uniformly spaced at 14 mm. Because it is most efficient at reflecting white lines, the sensor is also positioned at a height of 15 mm above ground level.

8. Summary

This study provides an overview of software, line sensor characterization, non-line sensors, control, and hardware. Software is required to produce code that allows the system to work. Several line sensor data are acquired during line characterisation to find the best sensor for detecting the track's white line. Non-line sensors were carefully reviewed in order to establish whether they are helpful and can be incorporated into the design. Different control methods are examined and discussed to determine the best control for the buggy. In the hardware overview, the buggy chassis was built, and the stress analysis was performed to check if the chassis could support the component.

A printed circuit board with six line sensors was created to detect the track's white line. The TCRT5000 sensor was chosen because it has a built-in daylight blocking filter, is less impacted by ambient light due to its infrared wavelength and is great at detecting the white line's edge.

Finally, the chassis design and material choices were evaluated. The most lightweight and easy fabrication technique are significant criteria in material selection. Acetal will be utilised as the material because of its capacity to offer sufficient support to keep the components together and since it is the lightest of the materials. This can result in a buggy that is both speedy and stable.

9. References

[1] "*HM-10 bluetooth module*," HM-10 Bluetooth Module, 25-Sep-2020. [Online]. Available: https://components101.com/wireless/hm-10-bluetooth-module. [accessed Nov. 16, 2022].

[2] "Microcontroller Engineering II Combined Lecture Notes 2022-23." University of Manchester Department of Electrical and Electronic Engineering, Manchester.

[3] "*Nucleo-F401RE*," NUCLEO-F401RE. [Online]. Available: https://os.mbed.com/platforms/ST-Nucleo-F401RE/. [accessed Nov. 27, 2022].

[4] "Procedure Handbook 2022-23." University of Manchester Department of Electrical and Electronic Engineering, Manchester.

[5] "Technical Handbook 2022-2023." University of Manchester Department of Electrical and Electronic Engineering, Manchester.

[6] Admin, "*Infrared radiation in the electromagnetic spectrum - byju's*," Infrared Radiation In The Electromagnetic Spectrum, 08-Jun-2022. [Online]. Available: https://byjus.com/physics/electromagnetic-spectrum-infrared-rays/. [accessed Nov. 19, 2022].

[7] Avago Technologies, "AEAT-601B Incremental Magnetic Encoder," AEAT-601B datasheet

[8] C. S. Baird, "*What is the color of the sun?,"* Science Questions with Surprising Answers, 03-Jul-2013. [Online]. Available: https://www.wtamu.edu/~cbaird/sq/2013/07/03/what-is-the-color-of-the-sun/. [accessed Nov. 25, 2022].

[9] E. Sammy Bongiorno, "*How to deal with Pallet Rack Deflection*," Damotech, 05-Oct-2022. [Online]. Available: https://www.damotech.com/blog/beam-deflection-limits. [accessed: Dec. 1, 2022].

[10] Vishay Semiconductor, "Reflective Optical Sensor with Transistor Output," TCRT5000 datasheet, [Revised Aug. 2009].