Contents

1. Introduction

This is the second report regarding the design of an autonomous line following a buggy project. This project aims to design a buggy that will follow a white line on a black track in the fastest possible time. Innately, the control system of the buggy will be the most crucial aspect in determining the project's success.

An autonomous line-following buggy integrates a sophisticated combination of line and non-line sensors, control mechanisms, software, and hardware to navigate the track; naturally, with these rather broad topics, different routes can be taken to achieve the same goal, with each route having advantages and disadvantages. This report aims to discuss and determine the most suitable route through research, experimentation, and calculation. On the sensors side, three photo emitter-detector combinations will be tested for their dark currents, ambient light currents, line spread functions and variation with height so that a suitable sensor for detecting the position of the line can be selected. For the control aspect, control algorithms, such as Bang-Bang and PD or PID control, will be discussed, as well as how they will interact with the chosen sensors. This leads to how this control algorithm will be implemented; the software is the crucial intermediary, bridging the inputs the buggy receives with its corresponding outputs. Lastly, the hardware decisions will discuss the chassis regarding material, design, wheel, and component placement. These decisions may be discussed in different sections of the report; however, they are heavily interlinked and will all work harmoniously to allow the buggy to navigate the track successfully.

Line sensors, typically infrared, are strategically placed on the buggy to detect and follow the predetermined path marked by a visible line on the ground. These sensors enable the buggy to make real-time adjustments to its trajectory, ensuring it stays on course. The control algorithm interprets the data from the sensors and executes precise motor commands to steer the buggy and provide appropriate current to the motors according to the situation. The software plays a crucial role in processing sensor data, implementing the control algorithms, and making decisions, such as turning. The hardware components, including motors, wheels, and chassis, form the physical foundation that brings the software's instructions to life. Together, these interconnected elements create a sophisticated and efficient autonomous system capable of navigating a path with precision and adaptability.

2. Software

2.1. Functional Summary

The software section outlines the methods to control the hardware, illustrating multiple functions used in the buggy. This section will include a functional summary, software constraints, a context diagram, a case diagram, and object specifications. The line sensors are critical to the operation of the buggy and are pivotal in accurately determining its position relative to the white line. These sensor outputs feed into the software, shaping the buggy's actions, whether it involves maintaining a straight trajectory or executing a turn, facilitated by a feedback control algorithm embedded in the software.

When the buggy encounters a ramp, the software adjusts the buggy's speed to ensure a seamless ascent or descent, leveraging information from the encoder.

The software computes the buggy's speed based on the encoder output. Line sensors further contribute by detecting line breaks in the track, prompting the system to respond accordingly— by re-engaging with the line on the opposite side of the break.

Furthermore, the software orchestrates the activation of line sensor LEDs in a specific sequence, providing continuous information to the buggy about the white line and ensuring sufficient data for timely turns. Additionally, the software incorporates a provision for the buggy to execute a 180° turn when the track concludes, facilitated by a Bluetooth Low-Energy (BLE) trigger.

In its multifaceted role, the software extends to managing the user interface. Capable of displaying error messages on the LCD screen, it also presents pertinent information such as battery life and speed, enhancing the user's interaction with the autonomous buggy.

2.2. Software Constraints

The software for the autonomous buggy operates within specific constraints imposed by the STM32 microcontroller, which features 512 kB of flash memory and 96 kB of SRAM. Within this limitation, the emphasis on efficient memory management is paramount, contributing to program stability and optimisation.

Given the 512 kB memory limit, the code must exhibit conciseness and robustness to prevent system crashes when inputs are provided. Achieving this involves meticulous error handling and adherence to effective software programming practices.

A critical consideration in software design is the control of current consumption of the motor, capped at 1.4 A. This constraint serves the dual purpose of mitigating overheating and conserving battery life, thereby ensuring smoother overall performance.

Furthermore, the control system's responsiveness is crucial for preventing line detection issues at maximum speed. The software is designed to elicit a response within 12.78 µs, effectively minimising the likelihood of the buggy veering off track and enhancing its overall stability during high-speed manoeuvres.

2.3. Context Diagram and Table of Messages

The context diagram (Fig.1) describes the software system by highlighting the essential components that execute movement in the buggy. The arrows describe the movement of signals between software and Input/Outputs. The table of messages (Table 1) describes each of the peripherals in detail as a complement to the context diagram. These two combined provide a detailed representation of the software system.

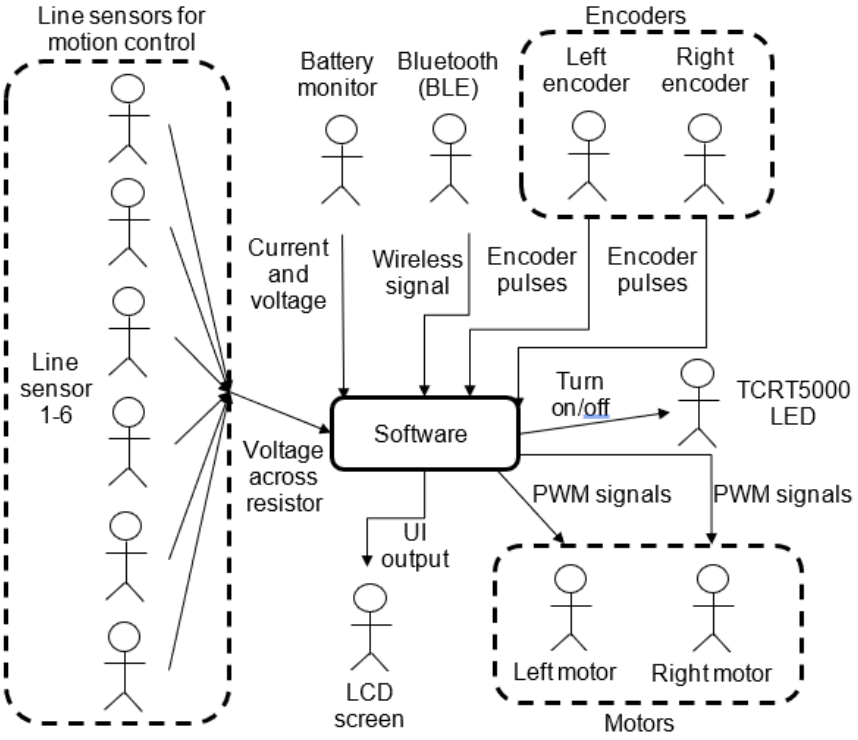

Fig. 1. Context diagram of buggy software system.

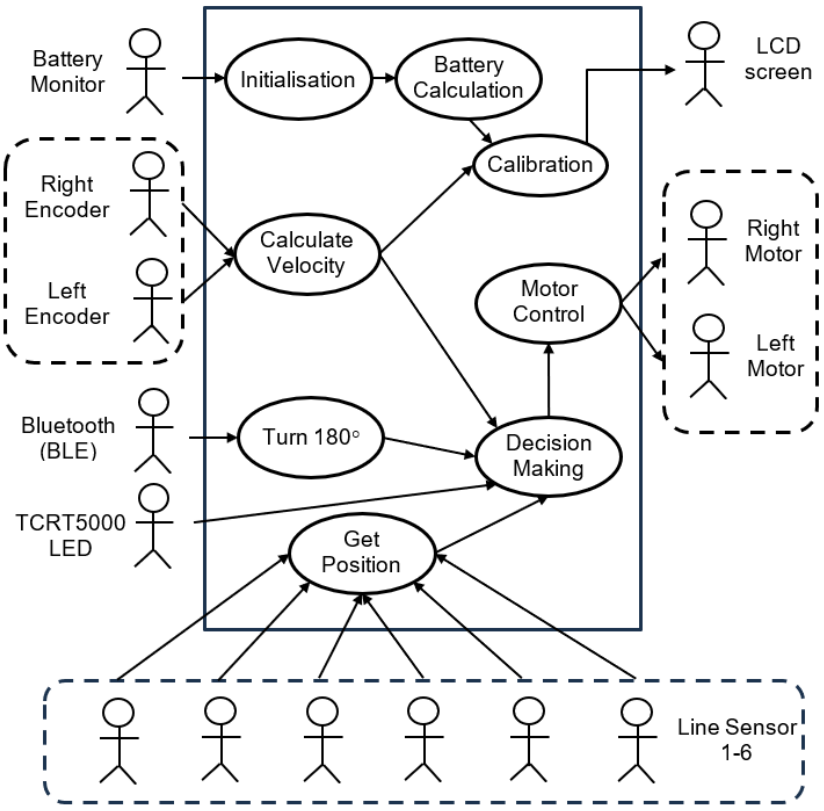

Fig. 2. Case diagram of buggy software system.

Table 1 Table of messages for buggy's context diagram

Name	Description	Source	Target	Interface	Size in bits	Arrival pattern
PWM signals	Control drive board (motor speed)	System	Motors	GPIO	16	Periodic
Battery voltage	Maximum voltage from batteries	Battery monitor	System	GPIO	8	Continuous
Velocity	The speed and direction of buggy when it's moving	Encoders	System	Motor drive board	8	Periodic
Blue-tooth	Wireless communication to let buggy make a 180° turn	BLE	System	Micro-controller	16	Aperiodic
Signal from line sensors	Detect the intensity of white line reflections	Line sensors	System	GPIO	6	Periodic
Voltage of motor	Voltage at the motor ends	System	Motors	GPIO	8	Continuous

2.4. Case Diagram and Descriptions

A case diagram (Figure 2) shows how the software interacts with peripherals and describes the flow of signals and functions. The descriptions table (Table 2) gives each class's full, detailed explanations and pseudo code.

Communications between the control system and peripherals are essential to move the buggy. A robust code should be tailored to produce efficient message exchanges within the use cases.

Table 2 Case Pseudo Code and Description

Case	Pseudo Code	Description
Initialisation	*Procedure initialise():* *turnOnBatteryMonitor();*	The initialisation function ensures the safe activation of the battery monitor, a critical step in the system's start-up. Incorrect initialisation poses a risk of component damage, potentially leading to the failure of the entire buggy.
Update Data	*procedure updateData():* *batteryV = readBatteryV();* *velo = calculateVelocity();* *printOnLCD (batteryV);* *printOnLCD(velo);*	The update data procedure involves reading the battery voltage, calculating velocity, and displaying these values on the screen. This step ensures accurate parameterisation for subsequent operations, contributing to the overall reliability and performance of the system.
Get Position	*procedure getPosition(…):* *intensity = readLS();* *processLSData(intensity);* *NextLEDSequence();*	During the "Get Position" phase, line sensors guide the buggy based on white line intensity. This continuous process informs the motor control system, ensuring the buggy stays on course, then switching to the following LED sequence.
Turn 180°	*Interrupt BTRecieved:* *perform180();*	The "Turn 180°" function orchestrates a controlled turn, triggered by a BLE signal. The buggy then returns to the starting point, employing the same control system used earlier.
Decision Making	*Procedure decisionMaking():* *getPosition();* *if not on line* *return(turningDecision());* *else: return(goStraight());*	The decision-making process continuously assesses the buggy's position and then decides; accordingly, this includes turning or maintaining a straight trajectory.
Motor Control	*procedure motorControl():* *updateData();* *pwmSig= ToPWM(decisionMaking());* *PWMToMotors(pwmSig);*	The motor control function relies on continuous data from line sensors and encoders. A PID controller processes this data, generating PWM signals translated into controlled motor movement, ensuring the buggy's precise navigation.

2.5. Object Specification

Table 3 shows the program classes that will be used in the software.

Table 3 Table of all Classes

Line Sensor	**Battery Monitor**
class LineSensor { *private:* *float intensity;* *public:* *// Get the current intensity value* *float getPosition() {…}* *};*	*class BatteryMonitor {* *private:* *float voltage;* *float current;* *public:* *void calculateBatteryCapacity() {…}* *// Calculate and manage battery capacity* *};*
TCRT5000 LED	**Encoder**
class LED { *private:* *bool ledState;* *int outputSignal;* *public:* *void on() {…} // Turn the LED on* *void off() {…} // Turn the LED off* *void toggle() {…} // Toggle the LED state* *};*	*class Encoder {* *private:* *float time;* *float encoderPulse;* *public:* *float calculateSpeed() {…}* *// Calculate and provide the current speed* *};*
LCD	**Motor**
class LCDScreen { *private:* *SetLCDPins();* *public:* *void DisplayInfo() {…}* *// Displays Velocity and battery capacity on LCD* *};*	*class Motor {* *private:* *float RWheelSpeed;* *float LWheelSpeed;* *public:* *float getVelocity() const {…} // Get the average speed of the motors* *};*

The described autonomous buggy system prioritises precision and adaptability through advanced control mechanisms and sensors. It navigates the track autonomously, considering factors like ramps, line breaks, and turning angle constraints while prioritising reliability. However, software development is an iterative process, subject to refinement and evolution from the initial plan through trial and error.

3. Line Sensor Characterisation

An array of line sensors will be secured to the bottom of the chassis to locate the white line that the buggy is following. These line sensors will each consist of a photoemitter and some form of photodetector, the latter of which will be used to calculate the location of the buggy. Therefore, it is imperative to the project's success that the sensor we choose is fully characterised so that the microcontroller receives consistent and accurate values to process.

3.1. Sensor Selection

Prior to the sensor characterisation lab, the main features of the available sensors were compared, and a few photodetectors were selected for analysis. Comparisons between the sensors are shown in Tables 4 and 5.

Table 4 Photoemitter Characteristics

Photoemitter	Wavelength Range/Peak (nm)	Suggested Current (mA)	Response Time (ns)	Beam Half angle at half intensity (°)
TCRT5000 [7]	950	60	unknown	unknown
OVL-5521 [5]	400-700	30	unknown	15
TSHA4400 [8]	875	100	600	20
TSHG6400 [10]	850	100	20	22

Table 5 Photodetector Characteristics

Photo-detector	Wavelength Range/Peak (nm)	Required Current or Voltage	Expected Output Range	Response Time (µs)	Angle of half sensitivity (°)	Daylight Blocking Filter?
TCRT5000 [7]	950	5 V	0.5-2.1 mA	unknown	unknown	yes
BPW17N [3]	620-960	5 V	0.1-1 mA	4.8	12	no
SFH 203 P [4]	400-1100	1.3V	5-9.5 µA	0.005	75	no
N5AC50108 [6]	650	Max 150 V	1-10 µA	unknown	unknown	no
TEKT5400S [9]	940	5 V	0.2-4 mA	6	37	yes

The main factor to consider when selecting a sensor and emitter pairing is how well it can detect the light from the emitter while not detecting other light sources. If the phototransistor detects light not reflected off the track, it will allow through more current, causing the calculated line position to be incorrect. The infrared sensor and LED pairings are desirable as they would be less affected by the ambient light, mainly in the visible range. The infrared range is 780 to 1100 nm, so the TCRT5000, TEKT5400S and BPW17N phototransistors are strong candidates. The SFH 203 P detects infrared light and light in the visible spectrum. Another aspect to consider is the current required to turn on the LEDs, as higher currents will require more power from the batteries and cause more power loss in the form of heat. The TCRT5000 excels in this aspect as it uses IR light detection, but compared to the other IR LEDs, it has a lower current requirement. A fast response time from the phototransistor would mean a quicker response to a change in line position. The SFH 203 P has by far the fastest response time, but it also has a very wide angle of half sensitivity, causing it to detect more ambient light rays than other options. The TEKT5400S features a daylight-blocking filter, which would significantly reduce the detection of ambient rays; however, the peak wavelength sensitivity is much higher than the wavelength of the

provided IR emitters. This would cause the current response to be decreased and the difference between white and black sections of the track to be decreased. The only other sensor featuring a daylight-blocking filter is the TCRT5000; therefore, considering other previously mentioned benefits, it is the sensor we fully characterised.

3.2. Lab Procedure

To measure the output current of the chosen sensors, they were soldered onto a small PCB that fits into a DIL socket, which was, in turn, soldered onto Veroboard. The piece of Veroboard could slide in and out of a test rig that held it at different heights and positions above a piece of the test track. The copper tracks on the Veroboard corresponding to the different pins on the sensor were then connected to a breadboard. This allowed the LED current to be changed using a resistor and the output current to be measured by taking the voltage across a second resistor in series with the phototransistor. The dark current was measured by disconnecting the LED and covering the sensor, and the TCRT5000 was measured to be 2.5 nA. This value is small compared to the output when the LED is on; therefore, an amplifier to remove it would not be necessary in the sensor circuit. The background illumination was tested by leaving the sensor face up on the desk 1 m from the desk lamp. The current was measured to be 2.5 µA, which could affect line sensing accuracy. However, in practice, the sensor will face the track, which will immediately reduce the ambient light, and additional steps can be taken, such as covers around the sensors. The variation in current with respect to height and the line spread functions for the sensor were then measured. These were taken at LED current values of 60 mA and 13 mA. Three repeats were taken for each data point. The error bars in all the graphs were taken to be the maximum and minimum values, representing the random error occurring when taking the measurement.

3.3. Lab Results

Any height change will significantly impact the current value produced if the sensors are positioned closer to the track, which could occur when the buggy starts on an incline. However, the software could mitigate this effect by comparing the sensor values to the left and right of the buggy. Regardless, this could introduce errors in the positioning of the line. The results from measuring output current with respect to height at 13 mA show less variation, and considering the benefit of reducing power consumption, this is a more appropriate LED current. Therefore, for the rest of the values for the TCRT5000, the LED current is set to 13 mA.

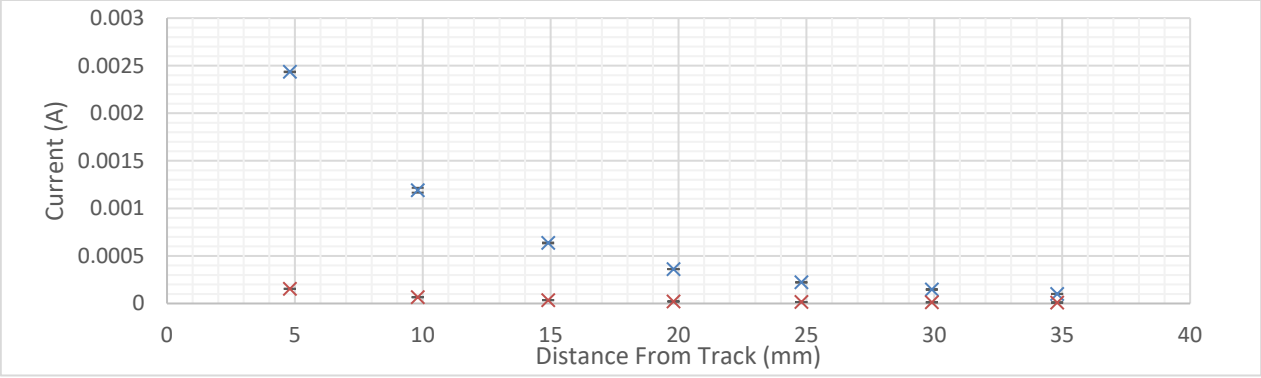

Fig. 3. Phototransistor output varying with height at LED current 60 mA.

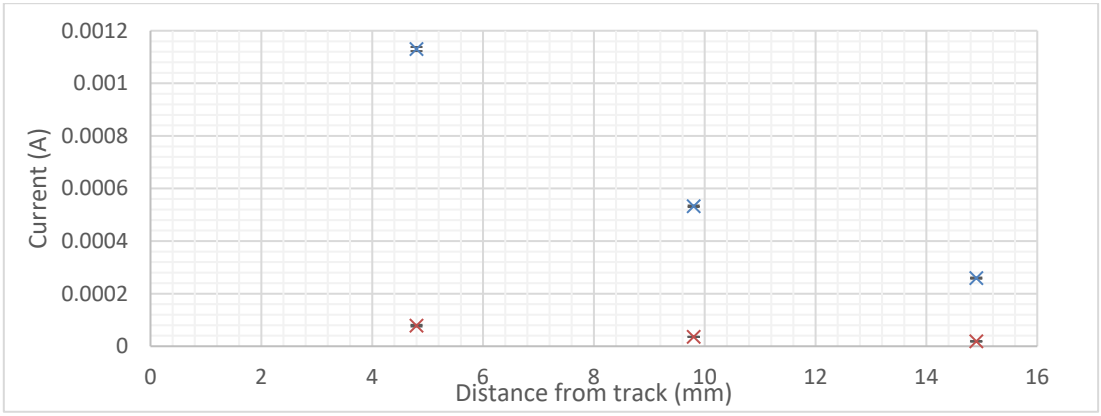

Fig. 4. Phototransistor output varying with height at LED current 13 mA.

To measure the line spread function, a piece of tape was applied to the Veroboard; this provided a clear reference point to measure the position – as the see-through test rig was marked with vertical line markings every centimetre. The piece of tape was marked in millimetre increments for the more granular line spread functions.

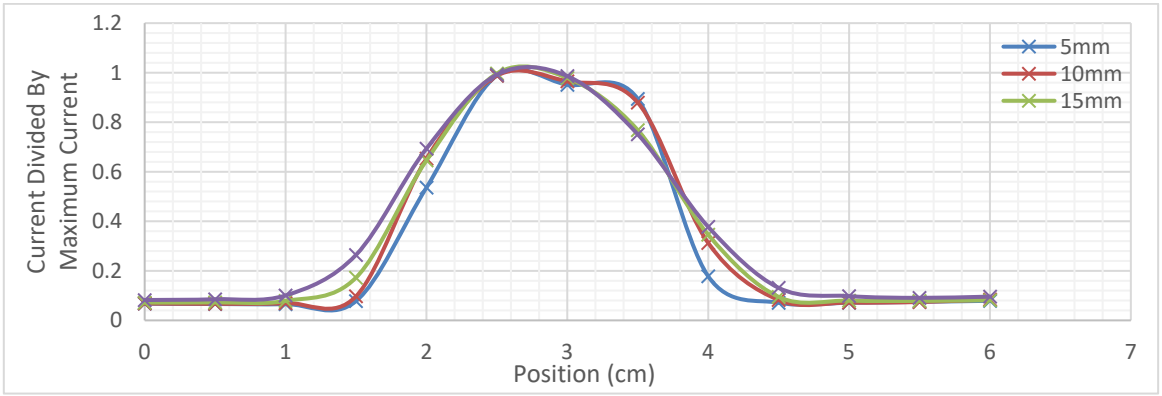

Fig. 5. Line spread function varying with heights.

The further the sensor was from the line, the more the edge of the line was blurred. At first, this seems like a disadvantage; however, by using multiple sensors next to each other and spacing them by the distance over which this blurring occurs, the position of the edges of the line can be measured accurately. The line spread function was repeated but with more minor changes in the position of the Veroboard.

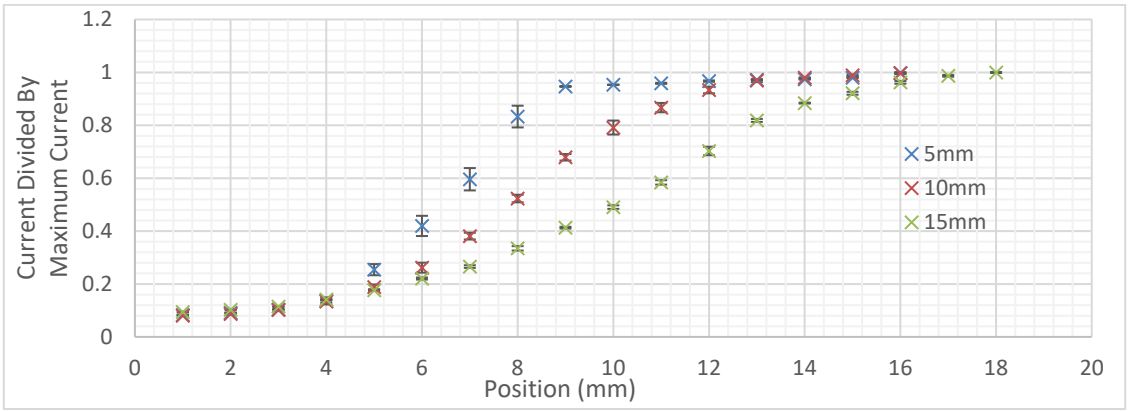

Fig. 6. Line spread function for varying heights.

The random error was much more significant when the sensor was close to the track. However, this could be because it is more sensitive to the position and, therefore, any error in the position measurement of the Veroboard.

A wider line spread would allow the buggy to differentiate between line breaks and the end of the line. If the line spread is larger than the line break, some light will be reflected onto the sensor, whereas the end of the line will have no light reflected. The width of the line spread should be considered when positioning the sensors on their printed circuit board, as the width of the line blurring should match how far apart the sensors are spaced. This ensures that the positions of the edges of the line are constantly being measured. Therefore, the sensors will be positioned 15 mm away from the track, allowing for a wide enough line spread while still being close to the track and, therefore, less vulnerable to interference from ambient light. This means that the sensors will be spaced by 15 mm, as this is the width of the line spread.

Two other sensors were characterised to compare the TCRT5000 to other options. One sensor pair was a visible light LED and a light-dependent resistor. This sensor had major flaws; the ambient light test made more current flow through the circuit than when the sensor was directly above the white line (1.6 mA compared to 0.8 mA), and the line spread function was so wide that crosstalk between each sensor on an array would be inevitable. Additionally, shining a torch near the sensor test rig caused the current output to change, and the random error on the data points was large compared to the TCRT5000. These factors made this sensor unusable for the precise measurements required.

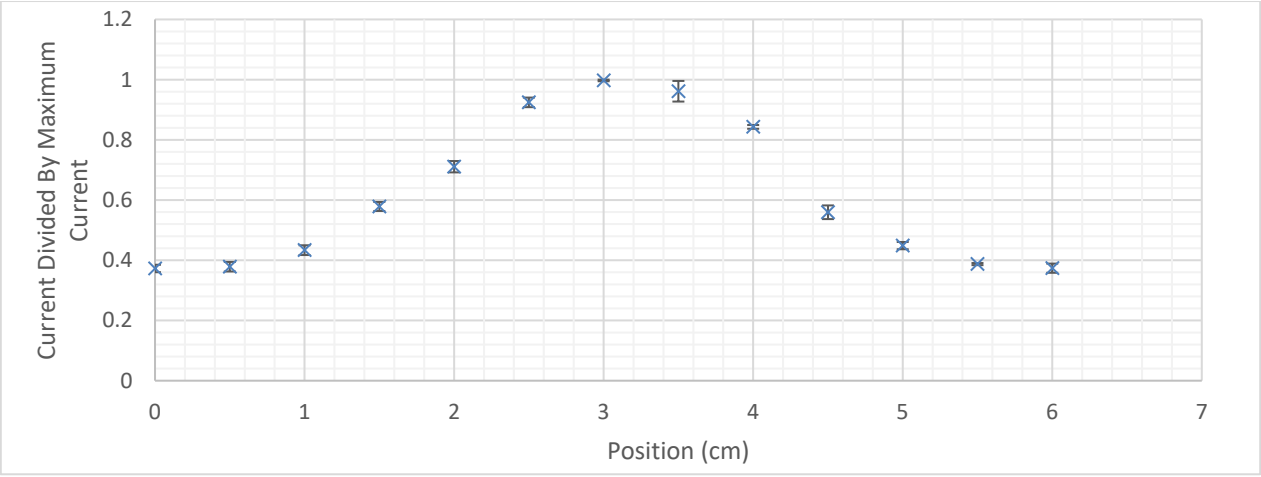

Fig. 7. Line spread function for LDR sensor.

The final sensor pairing to be compared was the TSHG6400 LED and the BPW17N phototransistor. This pairing was chosen as the LED's emitted wavelength closely matched the phototransistor's peak wavelength sensitivity. While this pairing was much more resistant to error caused by ambient light than the LDR sensor, the current through the phototransistor circuit was very low despite 60 mA powering the LED. The maximum current over the white line on this sensor was 18 µA compared to 2 mA on the TCRT5000 while running at the same LED current. The low current output would be more challenging to measure and would result in errors in calculating the position of the line. The high current draw of the LED would drain the battery life of the buggy much quicker, overall making this sensor less suited to our application.

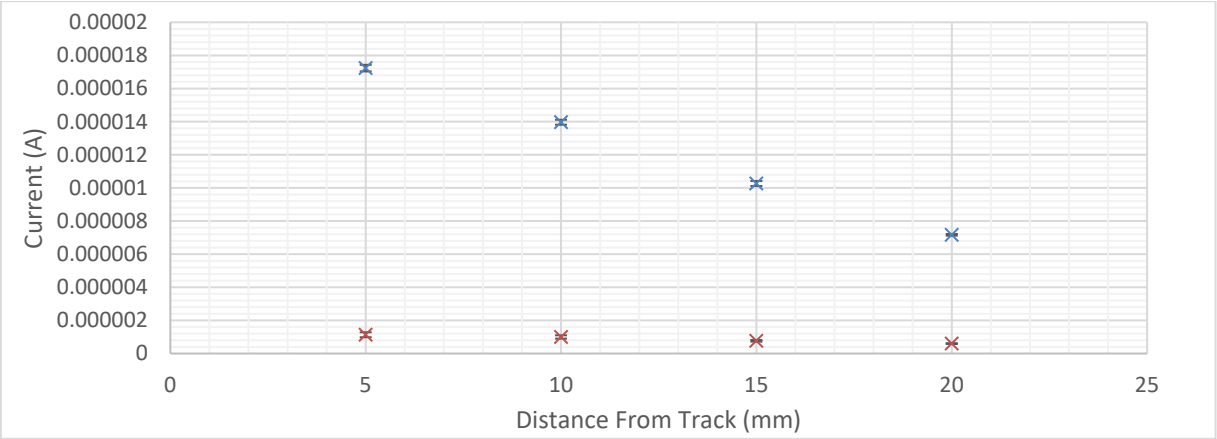

Fig. 8. Graph of current against distance from track for final sensor pairing.

In conclusion, the sensor package to detect the line will be the TCRT5000, with an LED current of 13 mA. This will be achieved by placing a 390 Ω resistor in series with the LED circuit. The height of the sensors off the track will be 15 mm, and they will be separated by 15 mm from the centre of each sensor. At this height, the current signal for line position will vary from 8 µA over the black line to 90 µA over the white line. With this data, the circuit diagram for the sensor can be constructed, and algorithms for the control of the buggy can be considered.

4. Circuit Diagram for Proposed Line Sensors

This section describes the layout of the sensors, the integration of these sensors onto a PCB and how it will be wired to the microcontroller. The PCB layout of the report is vital as the positioning and control of the buggy relies on the sensors' arrangement and the signals received from them.

4.1. Schematic Diagram

Figure 9 shows the circuit diagram of the line sensor PCB. P1 is the emitter power connected to the ULN2003, which controls the toggling of the sensor LEDs. P2 is the detector output connected to the phototransistor side of the line sensor, and P3 is used to provide the pin for 5 V and ground.

4.2. PCB Layout Diagram

The length of the PCB is 105.41 mm, and the width is 45.72 mm. The diameter of the screw holes are 4 mm. The board shows the six TCRT5000 sensors positioned in parallel and spread out evenly across the width of the PCB and the track layout and positioning of the other components.

4.3. Wiring Diagram

The wiring diagram shows the connections between the PCB and the microcontroller pins. The emitter power P1 is connected to the digital output pins on the microcontroller. The detector output P2 is connected to the microcontroller's analogue inputs. The microcontroller can convert the analogue signal to a digital signal and use it in the motor control algorithm. P3 is connected to the 5 V and ground, which is used to provide the power for sensors.

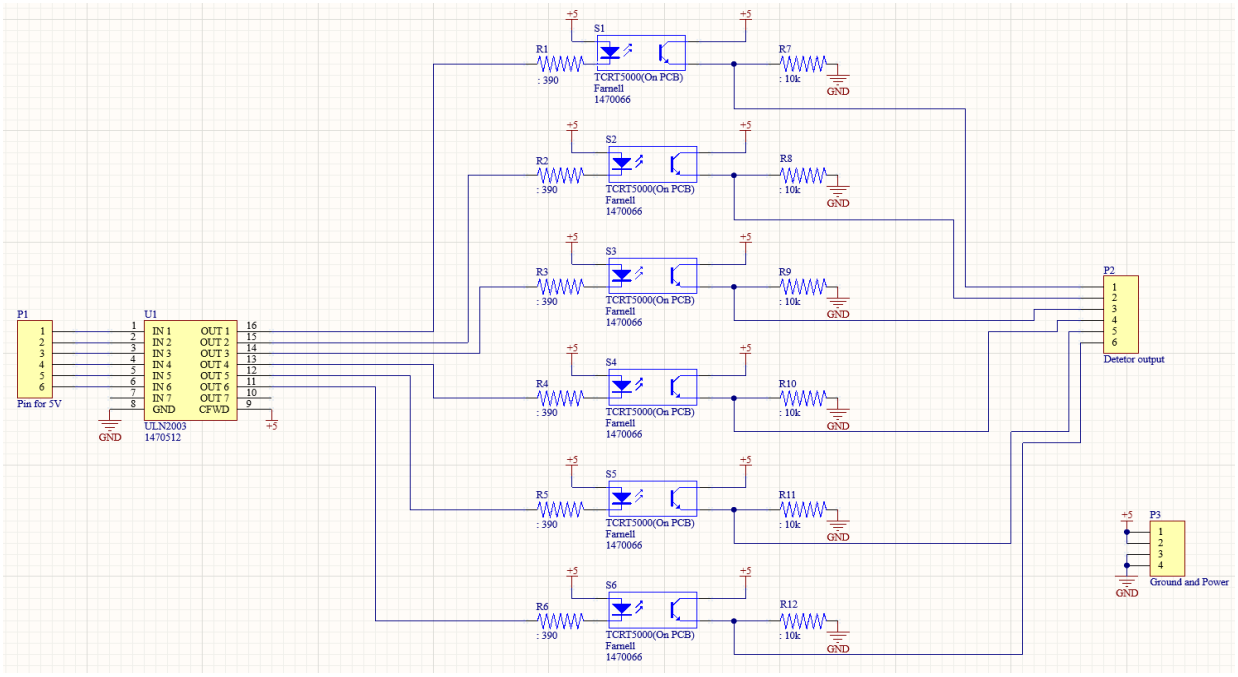

Fig. 9. Schematic diagram of line sensors.

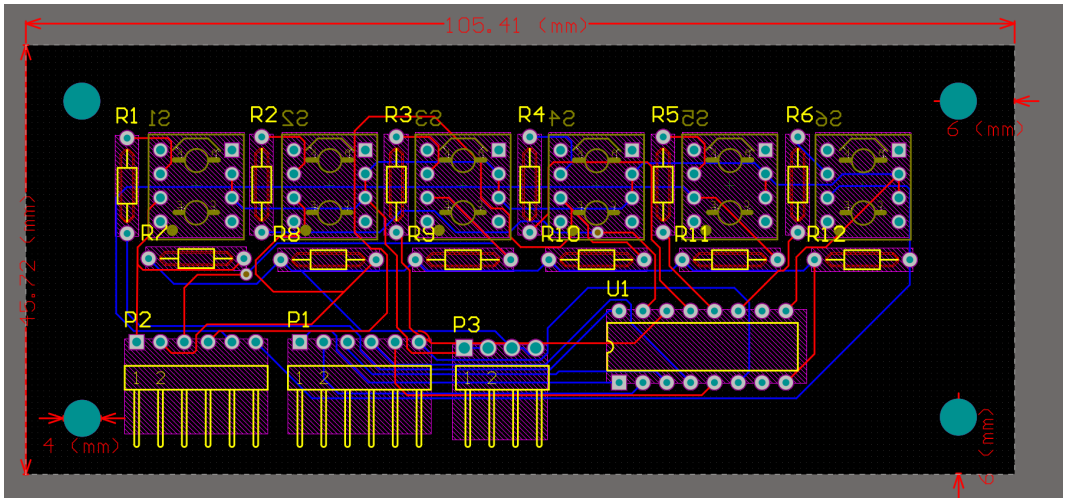

Fig. 10. PCB diagram of line sensors.

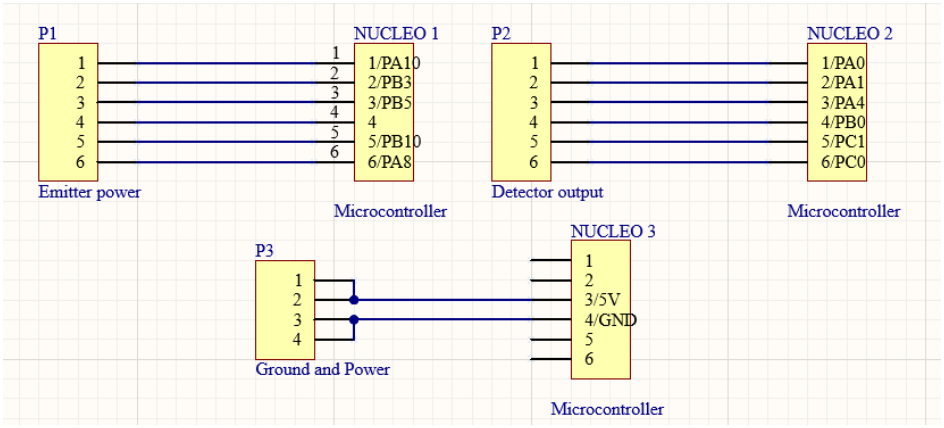

Fig. 11. Wiring diagram of PCB and microcontroller pins connections.

5. Non-Line Sensors

A buggy is only as good as the data it collects, and the position of the line is not the only thing the buggy needs to measure. For the smoothest and fastest operation, the buggy also needs to monitor the rotational speed of the wheels, the motor current, and the battery voltage. This also ensures that the buggy is operating within safe limits.

The buggy's speed is critical to control on inclines as it will require more torque to climb a slope and less to descend. To maintain a constant velocity, the rotational speed of the motors can be measured, and the motor voltage adjusted to vary the torque according to the angle of the incline. Buggy speed is also essential when active braking is used on the motors, as the buggy needs to know when it is stationary to stop applying reverse polarity to the motors and avoid going into reverse.

The speed and direction measurements will be calculated using quadrature encoders in each gearbox. The microcontroller can use counters and GPIO pins to count the number of ticks registered by these encoders in a time period. This gives the tick rate of the encoders. As the wheel velocity is a function of the tick rate, the wheel diameter and the gear ratio, the buggy's speed can be calculated using (1).

$$Wheel\ Velocity\ = ETR \times \left(\frac{\pi \times Wheel\ Diameter \times CPR}{Gear\ Ratio}\right) \quad (1)$$

Where *ETR* is the estimated tick rate and *CPR* is the number of “counts per revolution” of the encoder, the *CPR* of the encoders on the gearbox is 256.

The motor current can be measured using a 0.1 Ω resistor in series with each motor. These resistors are already on the motor drive board and are used to convert the motor current into a voltage. This voltage can be measured using an analogue to digital converter on the microcontroller board. The peak safe voltage would be 0.14 V but in continuous operation the voltage should be lower to minimise the risk of overheating and extend the battery life.

The remaining battery level can be approximately calculated using the motor drive board’s Dallas DS2781 circuit [1] which sums the battery voltage at regular intervals. This gives the battery level as:

$$E \approx V_{BATT} \sum_k i_k \Delta T \quad (2)$$

An analogue to digital converter on the microcontroller can be used to measure the voltage on pin 8 of the motor drive board which is connected the DS2781.

Measurements from these sensors will be critical to the success of the project.

6. Control

The project's success relies not just on the optimisation of the individual components but also on how the measurements recorded by the sensors are used in software to guide the buggy around the track. If the buggy follows the line but only loosely, the time the buggy takes to go around the track will be increased as it oscillates across the line and travels further than is necessary. Unless otherwise stated, all equations in this section are taken from [1].

6.1. Line Following

From the characterisation of the TCRT5000, the output voltage is higher when the sensor is closer to the white line; hence, if there are line sensors on both the left and right sides of the

buggy, the output voltages of the sensors on the left can be subtracted from the voltages of the sensors on the right, to give the error. If the output voltage on both sides is approximately the same, then the white line is between those two line sensors, and the error is zero. When the buggy veers to the right, the sensor output voltage on the left will become higher, giving a negative error, so the buggy will start turning to the left to bring the difference between the output voltages back towards zero. When the buggy veers to the left, the sensor output voltage on the right will increase, giving a positive error so the buggy turns to the right. The sign of the error is used to control the direction by changing the wheels' speed - for example, turning to the right requires the left wheel to rotate faster than the right and vice versa.

6.2. Proportional vs Bang-Bang controllers

Bang-Bang controllers are very simple. They have high accuracy and only require a little complex functionality or memory. The logic is straightforward – when the buggy reaches a certain distance beyond the line, the controller instructs the buggy to turn by a set amount until the buggy overshoots the line on the other side. A serious disadvantage is that this would lead to the buggy oscillating across the line in a sawtooth manner, which would create a lot of vibrations, leading to more problems (such as damage to the buggy and increased errors). Furthermore, the constant turning and overshooting will slow down the buggy. Bang-Bang control is viable if the desired application only requires accuracy without considering other issues.

Proportional (P) controllers are smoother than Bang-Bang controllers. The output of the P controller varies with the error value and performs better in some cases. However, its most significant problem is that its steady-state error can be large, so the buggy may not drive along the centre of the track; this is because when the error is small, the reaction of the P can be negligible, especially in the case of small values of K_p or significant external drag influence. This phenomenon can be alleviated when the value of K_p is increased; however, if K_p is too high then the system will become unstable. Therefore, the P controller is suitable in applications, requiring only high-speed response, low turbulence, and low accuracy.

6.3. P, I, and D in PID controllers

Proportional(P) control:

$$u(t) = K_p e(t) \tag{3}$$

Where $u(t)$ is the control output with time, Kp is the proportional constant, $e(t)$ is the error which is the difference between the setpoint (SP) and the actual process variable (PV). For P Control, the output is proportional to the error. The benefit is this system has a rapid response to changes. However, it easy to inadvertently cause an over-active controller causing the system to be unstable.

Integral(I) control:

$$u(t) = K_i \int_0^t e(t)\, dt \tag{4}$$

Where $u(t)$ is the control output with time, K_i is the integral constant, $e(t)$ is the error which is the difference between the setpoint (SP) and the actual process variable (PV). For Integral control, the output is integrating the cumulative sum of past errors over time. The benefit is enhancing stability by predicting and counteracting potential future errors. However, because the result is an accumulation, PI controllers will have a slower response in dynamic systems.

Derivative(D) control:

$$u(t) = K_d \frac{de(t)}{dt} \quad (5)$$

Where $u(t)$ is the control output with time, K_d is the derivative constant, $e(t)$ is the error which is the difference between the setpoint (SP) and the actual process variable (PV). For D control, the output differentiates the error which helps to add damping to the controller. On the other hand, the introduction of a derivative term can exacerbate noise in the sensors which makes D controllers unsuitable for noisy applications.

6.4. PID Controller

PID control:

$$u(t) = K_d \frac{de(t)}{dt} + K_i \int_0^t e(t)\,dt + K_p e(t) \quad (6)$$

The whole PID control is formed by adding the three kinds of error and letting the control system make the right decision to reach the setpoint. The PID control loop continuously calculates this process. Once it senses a change, it will adjust and repeat the process. The proportional method generates output directly proportional to the instantaneous error. The integral method accumulates the error to make it become zero, while the differential method considers the rate of change in the process and predicts future displacement behaviour, enhancing system response speed. Compared to PI control, to overcome an overactive controller, a derivative response is applied. It calculates the derivative of the error and multiplies it by a positive constant K_d, which can significantly suppress an overactive PI controller.

6.5. Controller Tuning

Choosing the values of the constants K_p, K_d and K_i is obviously a key consideration with enormous ramifications for the project's success. Several methods were considered, but manual tuning was chosen as the project was being optimised for performance. Another critical factor will be the sampling time. If data is sampled too frequently, the buggy's power draw will be unnecessarily high, and the software will become needlessly resource intensive. Alternatively, if the sample time is too low, the buggy will not respond in time to the change in relative position of the white line and crash into the walls of the track. Exact sample times will depend on real-world trials of the buggy to fine-tune these values and will be dependent on software response and buggy speed.

6.6. Algorithm and Sensor Design Integration

The derivative part of PID control amplifies noise on the sensors, so it is imperative that noise is controlled. One of the control measures is that the buggy chassis will shield the sensors from ambient light. Placing the sensors closer to the line also leads to clearer measurements, and one of the reasons for selecting the TCRT5000 is their daylight-blocking filter, which will further mitigate the effects of sunlight.

Rotational speed will also be controlled using PID control. The quadrature encoders will measure the rotational speed, and the voltage to the motors will be increased to increase the buggy's speed. The error here will be the difference between the target rotational speed and the measured rotational speed, where the sign of the error will indicate whether the buggy needs to accelerate or decelerate.

The spacing of the sensors also helps the buggy cope with breaks in the line. Because the arrangement of sensors is symmetrical about the line, if there is a break in the line, then the error should not change significantly. Also, as the size of the line breaks is not larger than

the line spread function, the buggy should know it is still on the track.

6.7. Motor Control

As the microcontroller only has binary output, but the motors are an analogue component, there must be a way to interface these components. Pulse Width Modulation (PWM) varies the time the microcontroller pin is high (the duty cycle) to vary the average output voltage. If the duty cycle is 50%, then the pin would be high 50% of the time and low 50% of the time. A low-pass filter can then be used to stabilise the output at 50% of the voltage.

The direction of the motors will be controlled using an H-bridge configuration, which is already a part of the motor drive board and is shown in Figure 12.

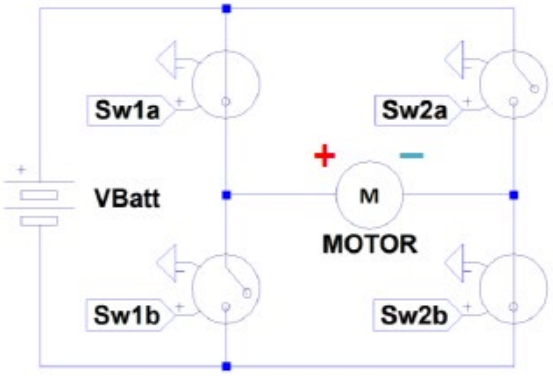

Fig. 12. H-bridge configuration [1]

If the buggy needs to go forward, switches 1a and 2b will be closed, creating a forward voltage across the motor. If switches 2a and 1b are closed, the polarity and direction of the motor is reversed.

If the polarity of the voltage across the motor is suddenly reversed while the buggy is moving, the motors oppose the motion of the buggy, reducing the speed through active braking.

7. Hardware Overview

The chassis is a crucial element of the autonomous line-following buggy, impacting stability, functionality, and overall performance. A well-designed chassis ensures optimal weight distribution for balanced line-following manoeuvres. Considerations for chassis design include material, ease of manufacture, shape, dimensions, and component placement.

The chassis is the platform for mounting various components, such as sensors, batteries, and the microcontroller. Proper positioning and secure attachment are essential for accurate and reliable autonomous operation. Component placement also affects weight distribution, influencing stability and manoeuvrability. Maintaining a low and well-balanced centre of gravity is crucial to prevent tipping, especially during inclines or high-speed turns.

While the material choice influences buggy weight, other material characteristics, such as flexural strength, ultimate tensile strength, and Young's modulus, impact stress-handling abilities. Table 6 [1] provides relevant material properties for potential materials. An effective chassis should withstand stress, preventing significant buckling or bending under component loads.

7.1. Material selection

When evaluating materials for designing the chassis, it is essential to consider factors such as ease of manufacture, structural properties, and overall suitability for the application.

Acetyl is a thermoplastic that can be easily moulded and machined into intricate shapes via laser cutting, making it a favourable choice for the chassis. Furthermore, acetyl has a high stiffness and high dimensional stability. Additionally, being a thermoplastic, it is cost-effective regarding both material and manufacturing costs.

Other potential chassis materials include glass-reinforced laminate, aluminium, and mild steel. While glass-reinforced laminate has many favourable properties and a comparable density to acetyl, its complex and costly manufacturing process makes it unfeasible in this application. Aluminium has a wide range of mature manufacturing processes, which would add significant cost and complexity. Furthermore, aluminium has one and a half times the density of acetyl, which makes it unsuitable. Mild steel has significantly better material characteristics in all areas except density, four times that of acetyl, making it inappropriate for the chassis.

Acetyl offers a balanced combination of ease of manufacture, favourable mechanical properties, and durability. The material's ease of machining allows for the creation of intricate chassis designs without compromising structural integrity. Acetyl's lightweight nature contributes to improved overall vehicle efficiency and manoeuvrability.

Table 6 Comparison of material properties [1]

Material Characteristic	Acetyl	Glass-Reinforces Laminate	Aluminium	Mild Steel
Density (Mg/m^3)	1.8	1.9	2.7	7.8
Flexural Strength (MPa)	91	255	310	414
Ultimate Tensile Stress (MPa)	67	175	310	414
Young's Modulus (GPa)	2.8	11.5	69	207

7.2. Chassis layout and component placement

With the identification of an appropriate material, the chassis design can now be formulated to integrate all components in a space-efficient manner. The preference for a compact buggy facilitates smoother turns, particularly 180° rotations. To achieve this compact design, a 2-level chassis is ideal.

The topmost layer of the chassis is equipped with motor modules which are positioned for a rear-wheel-drive configuration. The microcontroller is at the rear, while the front section accommodates the battery pack. A pill-shaped slit near the centre facilitates efficient cable management. Conversely, the lower layer secures the motor drive board on its upper side and the line sensor array PCB on its underside. As depicted in Figures 2 and 3, these components will be securely fastened using screws. This arrangement offers an advantage in adjusting the height of the line sensors by modifying the screws.

Towards the front of the buggy, a ball castor threads through both chassis layers. Placing the ball castor as close to the front as possible minimizes the risk of the chassis colliding with the track when encountering an inclined surface.

Moreover, strategic placement of the heaviest components, the battery pack, and the motor modules at the front and rear of the buggy ensure optimal weight distribution. While having the battery pack on the lower tier would lower the centre of gravity, it would compromise the flexibility in the height of the line sensors.

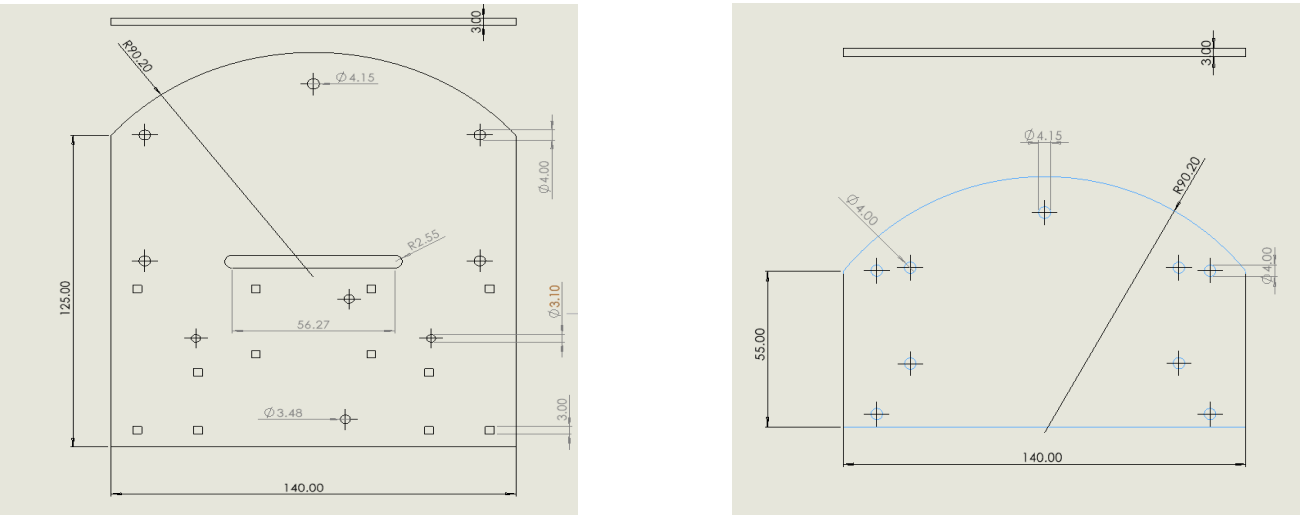

Fig. 13. Chassis drawing with dimensions.

7.3. Chassis Stress Analysis

The practicality of the chassis is contingent not only on weight distribution but also on its robustness. Bending or deflection in the chassis can introduce unpredictability, impacting the line sensors, weight distribution, and the responsiveness of the control system. Therefore, designing a chassis with minimal bending is highly desirable to ensure the system's reliability. Additionally, the chassis weight plays a vital role.

The volume of each chassis layer can be determined using the SolidWorks mass properties feature. By applying the density of acetyl from Table 6, the weight of each piece can be calculated. Furthermore, the weight borne by each layer, as presented in Table 7, can be estimated using component weights mentioned in [1].

Table 7 Shows the total weight and dimensions of the chassis, and each of its layers.

Chassis Layer	Volume of layer (cm^3)	Weight of layer (g)	Component weight held by layer (g)	Total Weight (g)
Upper	60.69	109.24	484.50	593.74
Lower	32.50	58.50	103.00	161.50
Total	93.19	167.74	587.50	755.24

The lower layer, housing the motor drive board and line sensor array PCB, supports a significant portion of the upper layer due to their attachment. Assuming the upper layer bears the entire weight of the lower level, the total weight carried by the upper layer is estimated at 484.50 g. Calculating these values, as shown in Table 7, underscores the advantages of using acetyl as the material, with the chassis weight constituting a minor proportion of the buggy's total weight.

A stress analysis can be systematically conducted by leveraging the data provided in Table 7 to evaluate the chassis' vulnerability to significant bending or, in extreme scenarios, breaking. Both layers exhibit identical breadth (b) and height (h) dimensions of 14 cm and

0.3 cm, respectively. Employing Equation (7), the area moment of inertia (I) can be computed as 3.150×10^{-10} m^4. Subsequently, utilising the calculated value of I, along with the length (l), load (W), and Young's modulus (E), the deflection for a point load (x_1) and a distributed load (x_2,) can be determined using Equations (8) and (9) from [1]. The outcomes of these calculations are presented in Table 8.

$$I = \frac{bh^3}{12} \quad (7)$$

$$x_1 = \frac{Wl^3}{48EI} \quad (8)$$

$$x_2 = \frac{Wl^4}{384EI} \quad (9)$$

Table 8 Calculations of point load and distributed load deflection (to 3 d.p.)

Chassis Layer	W (N)	l (m)	Deflection with point load, x_1 (m)	Deflection with distributed load, x_2 (m)
1	4.753	0.158	4.454×10^{-5}	8.814×10^{-6}
2	1.010	0.088	1.643×10^{-5}	1.643×10^{-5}

The findings in Table 8 demonstrate that the even maximum deflection of 0.445 mm is negligible concerning its impact on the buggy. Consequently, the chosen material and chassis design are well-suited for the intended application.

7.4. Impact of wheel placement

The placement of the wheel axle is set at a predetermined distance from the rear of the chassis. Having the wheels at the precise centre would enhance the efficiency of executing a 180° turn, but practical design constraints render such a configuration suboptimal. The separation between the wheels plays a critical role in influencing the turning capability of the buggy—opting for a minor separation yields a correspondingly diminished turning radius, promoting agility. However, this separation is limited by the need to accommodate the motor drive board while adhering to the specified width of the buggy, as dictated by a range of other essential design considerations. Thus, the wheel placement becomes a delicate balance between optimizing turning performance and accommodating the necessary components within the defined spatial constraints.

8. Summary

This report has investigated some key design decisions in creating an autonomous line following buggy, specifically those relating to software, control, sensors and chassis design.

Important software considerations include the case diagrams, the table of messages and the object specifications. Memory requirements will need to be carefully managed as there are limitations imposed by the microcontroller. The response time of the microcontroller will be 12.78 µs.

The algorithm for the line position will use PID control to ensure the smoothest following of the line and the fastest operation.

The sensor which will be used for following the line is the TCRT5000 as its response for light and dark current was superior to the other sensors tested. PID control would require the line sensors to have as little noise as possible, achieved through the daylight-blocking

filter built into the TCRT5000 and by shielding the sensors with the buggy's chassis. The sensors will be at the front of the buggy to facilitate a fast response and to allow more time for steering and will be 15 mm above the track. The symmetrical layout with 15 mm spacing between the sensors enables the buggy to gracefully manage line breaks and other track irregularities. The expected sensor output current will range from 8 μA to 90 μA.

For hardware, a compact 2-layer chassis design was chosen for stability and manoeuvrability. Due to its favourable density, machinability, and cost, Acetyl was selected as a chassis material. The maximum deflection for the upper and lower chassis was calculated as 45 μm and 16 μm, respectively. A limitation of this analysis was that the calculations assumed the chassis was a perfect rectangle, which does not account for the holes or the irregular shape.

9. References

[1] University of Manchester, *Embedded Systems Project EEEN21000 Technical Handbook*, 2023/24 ed.

[2] University of Manchester, *Embedded Systems Project EEEN21000 Procedures Handbook*, Version 2023.1 (September 2023).

[3] Vishay Semiconductors, “Silicon NPN Phototransistor”, BPW17N Datasheet, Rev. 1.3, 08-Mar-05

[4] Opto Semiconductors, “Silizium-PIN-Fotodiode mit sehr kurzer Schaltzeit”, SFH 203 P Datasheet, Version 2001-02-22

[5] Multicomp, “5.0mm Round LED Lamp”, OVL-5521 Datasheet, 25/05/12 V1.1

[6] LPRS, “LDR CdS PHOTO CELL”, N5AC-50108 Datasheet, 2012

[7] Vishay Semiconductors, “Reflective Optical Sensor with Transistor Output”, TCRT5000 Datasheet, Rev. 1.7, 17-Aug-09

[8] Vishay Semiconductors, “Infrared Emitting Diode, 875 nm, GaAlAs”, TSHA4400 Datasheet, Rev. 1.8, 15-Sep-14

[9] Vishay Semiconductors, “Silicon NPN Phototransistor”, TEKT5400S Datasheet, Rev. 1.6, 23-Aug-11

[10] Vishay Semiconductors, “High Speed Infrared Emitting Diode, 850 nm, GaAlAs Double Hetero”, TSHG6400 Datasheet, Rev. 1.2, 23-Aug-11