The University of Manchester

Department of Electrical and Electronic Engineering

Embedded Systems Project 2020-21

DESIGN REPORT #2

Title: Technical Characterisation

Group Number: 1

Group members:	ID Number	I confirm that this is the group's own work.
Ahmed Omara	10344777	☒
Ibrahim Al Brashdi	10513452	☒
Jiangsheng Wang	10471806	☒
Mohammad Ali	10510145	☒
Xavier Millican	10363193	☒
Zhengyi Zhao	10447097	☒

Tutor: Dr Emad Alsusa

Date: 10/12/2020

Contents

1. Introduction

This report contains the technical information necessary to complete the design of the buggy. Using the datasheets, a decision has been made on which line sensor to use after carefully examining their advantages and disadvantages, and through a practical, some of the characteristics of the TCRT5000 sensor have been obtained, by testing the sensors sensitivity across different hights and angles.

Deciding on the structure used for the programming is crucial, as it will dictate how the buggy responds and performs. The aim is to decide on how to program the buggy so it can take the measurements necessary to stay on the white line. Other sensors will also be implemented into the buggy, such as wall sensors and speed sensors.

Deciding on which control algorithm to use between PID and Bang-Bang is crucial, as it will dictate how the buggy behaves. The circuit diagram is one of the most important parts of the design, and should be designed with extreme care, as if it does not work, the whole buggy will not work. Finally, the chassis design will include the details of the positioning of all the parts required and will form the foundation of the buggy.

2. Software

When the buggy is switched on by means of a mechanical switch, the buggy will enter a sleep state until a white line (of width 17mm [1]) is detected by the line sensors. At this point a red LED will be displayed to indicate that the buggy has successfully powered on and the LED screen will display the remaining battery level.

After the line is detected the motors will be switched on and the buggy will move at a predefined speed. When at a turn the motors will move with different driving forces to move the buggy [2]. Once the buggy reaches the base of the slope the motor current will be increased to allow the buggy to ascend the slope and at the end of the slope the speed will be lowered to a sensible level to allow control of the buggy.

At the end of the line, a Bluetooth signal will be sent to the buggy to make the buggy turn back and go back to the start [1]. At the top of the slope, the speed of the buggy will be reduced so that it does not go out of control while descending the slope. The speed will be increased again as the buggy reaches the bottom flat surface to allow for faster speeds.

At the end of the track the buggy will sense that there is no line, at which point the buggy's motors will brake allowing the buggy to come to a halt (within 20 cm of the line [1]). After the buggy comes to a stop, it will remain stationary until the line is detected again or if there is another Bluetooth signal.

The constraints of the buggy include the following:

- Line sensors must be able to handle a break in the white line of up to 6mm [1]
- The buggy must be able to regain control if there is a change in the surface height of the track of maximum 4mm [1]
- The buggy must wake up quickly with minimal delay when the line is detected
- Line sensors must be able to handle a variation in reflectivity of the line [1]
- The buggy must be able to cope with a bend radius of minimum 50mm [1]
- The buggy must be able to navigate without touching the walls of the track which has a minimum width of 280mm [1]

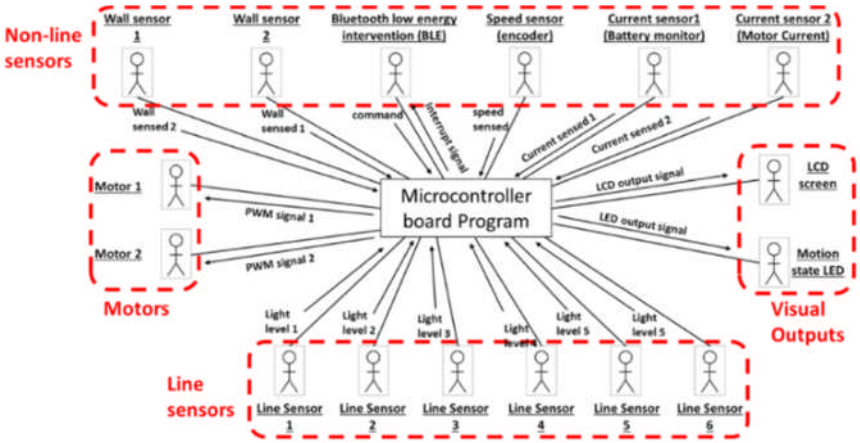

Figure 1 – Context diagram illustrating how the microcontroller program interacts with all of the devices attached, showing the messages being sent either way

Table 1 – Messages received to and from the microcontroller

Name	Description	Source	Destination	Interface	Size in bits	Arrival pattern	Comments
Light level 1	Light level received	Light sensor 1	Microcontroller	Analog In	32	Periodic	None
Light level 2	Light level received	Light sensor 2	Microcontroller	Analog In	32	Periodic	None
Light level 3	Light level received	Light sensor 3	Microcontroller	Analog In	32	Periodic	None
Light level 4	Light level received	Light sensor 4	Microcontroller	Analog In	32	Periodic	None
Light level 5	Light level received	Light sensor 5	Microcontroller	Analog In	32	Periodic	None
Light level 6	Light level received	Light sensor 6	Microcontroller	Analog In	32	Periodic	None
LED output signal	LED indicating current operational state	Motion state LED	Microcontroller	GPIO	1	Asynchronous	None
LCD output signal	Output of the value of the battery level	LCD screen	Microcontroller	GPIO	32	Continuous	None
Wall sensed 1	Wall location relative to buggy	Wall sensor 1	Microcontroller	Analog In	32	Periodic	None
Wall sensed 2	Wall location relative to buggy	Wall sensor 2	Microcontroller	Analog In	32	Periodic	None
Interrupt signal	Signal sent to check if Bluetooth signal received	Microcontroller	Bluetooth low energy intervention (BLE)	Serial	32	Asynchronous	None
Bluetooth command	Bluetooth signal sent to microcontroller for U turn of buggy	Bluetooth low energy intervention (BLE)	Microcontroller	Serial	32	Asynchronous	None
Speed Sensed	Speed of rotation about vertical axis of wheel	Speed sensor (Encoder)	Microcontroller	Analog In	32	Periodic	None
Current Sensed 1	Current sensed from batteries	Current sensor 1 (Battery monitor)	Microcontroller	Analog In	32	Periodic	None
Current Sensed 2	Current sensed from motors	Current sensor 2 (Motor current)	Microcontroller	Analog In	32	Periodic	None
PWM signal 1	PWM signal sent to motors to control speed	Microcontroller	Motor 1	PWM	32	Periodic	None
PWM signal 2	PWM signal sent to motors to control speed	Microcontroller	Motor 2	PWM	32	Periodic	None

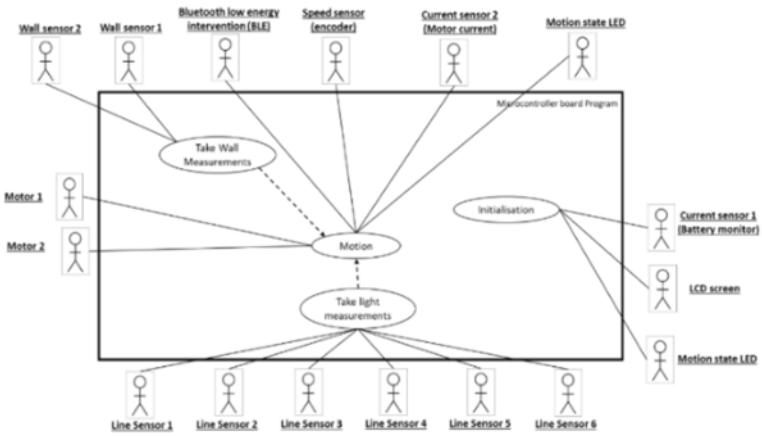

Figure 2 – Case diagram detailing the different use cases that the program will use and how these use cases will interact with the relevant hardware

The case description for the software of the buggy can be described as follows:

Use Case: Initialisation
- IF switch is ON
- Sleep
- Power on red led
- GOTO Take_light_measurements
- ELSE
  - GOTO Initialisation

Use Case: Take_light_measurements
- Emitter sends out light to environment
- Measure current of light sensor from light received
- IF light received
  - GOTO Motion
- ELSE
  - GOTO Take_wall_measurements

Use Case: Take_wall_measurements
- Wall sensor sends signals to environment
- Measure current of signals received back to the sensor
- IF distance between wall and buggy < 1cm
  - Back off from the wall
- GOTO Motion
- ELSE
  - GOTO Motion

Use Case: Motion
- IF Bluetooth signal NOT detected
- IF motion on flat (distance between sensors and floor = 2.5mm)
  - Turn on motors via PWM
  - Turn on green led
  - Turn off red led
- GOTO Take_light_measurements
- ELSIF motion on slope down-up (distance between sensors and floor < 2.5 mm)
  - Increase voltage to motors via PWM
  - GOTO Take_light_measurements
- ELSIF motion on slope up-down (distance between sensors and floor > 2.5 mm)
  - Lower the voltage to motors via PWM
  - GOTO Take_light_measurements
- ELSIF motion at a turn (line shifts to one side)
  - Turn one motor in forward motion
  - Turn one motor in reverse motion
  - GOTO Take_light_measurements
- ELSIF Wall sensed
  - GOTO Take_wall_measurements
- ELSIF line not sensed
  - Turn motors in reverse
  - Turn off motors when speed sensed = 0
  - GOTO Take_light_measurements
- ELSE
  - GOTO Initialisation
- ELSE
  - Turn the buggy by 180°

The object specification of the following use cases can be seen below:

```
class rest {

    private:
        DigitalOut StateLED;
        AnalogueIn CurrentSensedBattery;
        C12832 lcd;
        float battery_level, max_current;

    public:
       rest(PinName lcdpin_1, PinName lcd_pin_2, PinName lcd_pin_3, PinName lcd_pin_4, PinName lcd_pin_5, PinName state_led_pin, PinName current_sensed_battery_pin);
       void get_buggy_state (void);
       void sample_current_sensed_battery (void);
       float getCurrentsensedbattery(void);
       void displaybatterylevel(void);

};
```

Figure 3 – Object specification of the initialisation use case

```
class line_measurments {

    private:
        AnalogueIn LineSensor1
        float linemeasurment;

    public:
        line_measurments(PinName linepin);
        void sample_line_measurment(void);
        float getlinemeasurment(void);


};
```

Figure 4 – Object specification of the line measurments use case

```
class wall_measurments {

    private:
        AnalogueIn WallSensor;
        float wallmeasurment;

    public:
        wall_measurments(PinName wallpin);
        void sample_wall_measurment (void);
        float getwallmeasurment(void);


};
```

Figure 5 – Object specification of the wall measurments use case

```
class motion {

    private:
        DigitalOut StateLED;
        PWMOut Motor_voltage;
        AnalogueIn Speed_sensed, CurrentSensedMotor1;
        Serial HM10_BLE_Module, PC_Control;
        float motorcurrent, speed, direction;

    public:
        motion(PinName motorpin PinName state_led_pin, PinName encoder_pin, PinName current_sensed_motor_pin, PinName BLE_pin_1, PinName BLE_pin_2, PinName PC_pin_1, PinName PC_pin_2
        void sample_motor_current(void);
        float getmotorcurrent(void);
        void sample_speed(void);
        float getspeed(void);
        float getdirection(void);
        void bluetoothsignalcommand(void);
        void bluetoothinterruptsignal(void);
        void get_buggy_state (void);

};
```

Figure 6 – Object specification of the motion use case

3. Line sensor characterisation

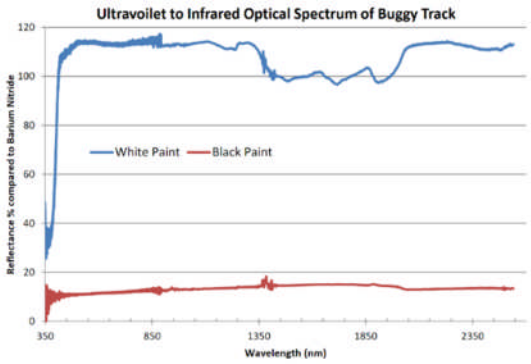

Figure 7 – Reflectance of white paint vs the reflectance of black paint [2]

When deciding which emitter and phototransistor to use, there are multiple factors that should be taken into consideration. Firstly, the wavelength emitted from the emitter has to be within the range of the phototransistor, and preferably close to the peak wavelength sensitivity of the phototransistor. Secondly, the size of the emitter/phototransistor has to be considered. Multiple sensors will be required, which means that if the phototransistor/emitter is too big, it will be difficult to fit multiple sensors into the chassis. Also, The wavelength sensitivity range of the phototransistor is extremely important, as if the range is too big, the phototransistor

will be affected by background light. As seen in figure 7, the wavelength reflected into the phototransistor depends on the surface it is reflecting from, as the reflectance of white paint is higher than that of black paint. If the range is too small, the phototransistor might not detect the reflected wavelength. The forward voltage is important, as it represents the amount of voltage required to get the current across the diode. Also, the angle of half sensitivity/intensity should be considered. If the angle is too narrow, then the placement of the detector/emitter will be limited as the emitted wavelength and the detector will not be as accurate if they are placed incorrectly or inaccurately.

The BPW17N is a phototransistor with a peak wavelength sensitivity of 825 nm [3]. It has a wide range of wavelength sensitivity, from 620 nm to 960 nm [3]. Due to its narrow aperture, it is not sensitive to background light (±12°) [3]. Another phototransistor is the SFH203P. It has a peak wavelength sensitivity of 825 nm, and a very wide range of 400 nm to 1100 nm [4]. It also has a wide angle of half sensitivity, at ±80°, as well as a fast switching time at 5 ns. The wide angle of half sensitivity means that its positioning is more flexible, as it will be able to detect the wavelengths from a very wide angle. However, it also makes it susceptible to detecting background light. The final phototransistor available to select is the TEKT5400S, which has a peak wavelength sensitivity of 920 nm, and a range of 850 nm to 980 nm, as well as a daylight blocking filter, which means it will be insensitive to background light [5]. Furthermore, it has a relatively high angle of sensitivity of ±37°, as well as fast response times, and is paired with the TSKS5400S [5].

The TSKS5400S IR emitter also has many advantages. Firstly, it has a low forward voltage value of 1.3 V. Also, it emits infrared waves of wavelength 950 nm, which is within TEKT5400S' range [6]. Finally, it has a high angle of half intensity of ±30°. Another available emitter is OVL-5521. Its advantage is that it has a low DC forward current of 30 mA [7]. The TSHA6203 is an IR emitter with a peak wavelength of 875 nm [8], which has low forward voltage and is suitable for high pulse current operations. The final emitter is OPE5685. It emits IR waves with a wavelength of 850 nm. One advantage is that it has a low forward voltage. Also, it has a wide beam angle at ±22θ. Furthermore, it has very high speed, as the rise time is 25 ns.

Finally, there is TCRT5000, which is an optical sensor. It has both an emitter and a phototransistor with an emitter wavelength of 950 nm [9]. The phototransistor is coated in order to block out anything outside of the IR spectrum [9]. Furthermore, the package is small compared to the other emitters/phototransistors, which makes it easier to assemble on the buggy.

Table 2 – Comparison of the different parameters of different line sensors

Sensor	Type	Dimensions (L x W x H) (mm)	Sensitivity to sunlight	Peak Wavelength sensitivity (nm)	Wavelength emitted	Wavelength sensitivity range (nm)	Angle of half sensitivity/ Intensity (±°)
BPW17N	Phototransistor	3.3 x 2.4 x 29.8	Low	825	-	620 - 960	12
SFH203P	Phototransistor	5.9 x 5.9 x 34	High	825	-	400 - 1100	80
OPE5685	IR Emitter	5.7 x 5.7 x 32.7	-	-	850	-	22
OVL-5521	IR Emitter	5.9 x 5.9 x 32.7	-	-	?	-	8
TCRT5000	Sensor	10.2 x 5.8 x 7	Low	950	950	?	?
TEKT5400	Phototransistor	5 x 2.65 x 5	Low	920	-	850 - 980	37
TSKS5400	IR Emitter	5 x 2.65 x 5	-	-	950	-	30
TSHA620	IR Emitter	5.8 x 5.8 x 35.9	-	-	875	-	12
VT90N2	IR Emitter	?	-	-	?	-	?

Table 2 above compares all of the relevant parameters of the different phototransistors/emitters. It gives a general overview of the dimensions, sensitivity to sunlight, wavelength ranges etc. This helps with the analysis of the different types of line sensors and provides a general summary of all the benefits and drawbacks.

Looking at the available emitters/phototransistors, VT90N2 can be excluded as the datasheet does not include the required information. OVL-5521 can also be excluded as the emitted wavelength is not available in the datasheet, and it can easily be replaced by OPE5685. Looking at the previous criteria, the possible pairings are TEKT5400 and TSKS5400, OPE5685 and SFH203P, OPE5685 and BPW17N, TSHA620 and BPW17N, TSHA620 and SFH203P, and TCRT5000.

SFH203P has an extremely high wavelength sensitivity range, at 400-1100 nm, thus making it highly sensitive to background light, which is not ideal. This removes the TSHA620-SFH203P and OPE5685-SFH203P pairings. Looking at the dimensions of BPW17N, OPE5685 and TSHA620, the lengths are high, which would make it hard to fit them into the chassis. This leaves us with TCRT5000 and TEKT5400-TSKS5400. Looking at the TEKT5400 wavelength sensitivity range, which is 850-980 nm, even though the emitter wavelength of TSKS5400 is within that range, when the emitter wavelength gets reflected it might not be within the range, as the range is quite narrow. This could cause problems, as the reflectance of white paint is much higher than that of black paint, as seen in figure 7, so a relatively wide range is required. This leaves us with TCRT5000. TCRT5000 has both an emitter and a phototransistor together in a small package [9]. This makes it easy to fit multiple sensors into the chassis. Furthermore, since both the emitter and phototransistor were designed to be matched together, this eliminates any chance of the emitter and phototransistor being incompatible. The phototransistor is coated, which blocks out any background light not in the IR range [9]. The phototransistor and emitter are separated by a small barrier which makes it so that the phototransistor can only detect the reflected wavelengths [9]. All of these factors make the TCRT5000 the sensor of choice.

There are several constraints related to the sensors that could cause issues in the buggy working. First, if there is a line break on the white track, the reflected wavelength will be different from that which is expected to keep the buggy on the white line, as the emitted wavelength will be absorbed by the black paint, which as seen from figure 7, has much lower reflectance than white paint. This would cause the buggy to stop working as no white line would be detected. As mentioned in the upcoming control section, this could be mitigated by programming the sensors in a way where if no white line is detected for a short period of time, the buggy continues in the previous motion and direction.

The second constraint is how direct sunlight/background light will affect the sensors. The sunlight can saturate the detector and make the sensor think it is above the white line when it is not. The TCRT5000 has a built in barrier placed between the emitter and detector, which blocks the emitted IR wavelengths from directly entering into the detector, as well as a coating on the detector that blocks out every wavelength outside of the infrared spectrum. As observed during the practical, ambient light will not affect the sensor due to the coating on the detector. Also, the coating will block out approximately 47% of sunlight, as at the earth’s surface, sunlight consists of approximately 53% infrared, 44% visible, and 3% ultraviolet light [10]. The sensors will be aimed towards the track, not the sun, as the purpose of the

sensors is to detect where the white line is. This will block a lot of the sunlight. Furthermore, the external light level could be measured constantly in order to adjust the sensors. Finally, the buggy could be primarily used indoors in order to eliminate this constraint.

A practical was carried out in order to find out the optimal height and the response of TCRT5000 as it moves across the white line. The sensor is soldered into a mini PCB, and four pins of the TCRT5000 are mapped into the eight pin DIL socket [1]. The emitter of the TCRT5000 uses pins 2 and 7, while the phototransistor uses pins 3 and 6. The orientation of the sensor is extremely important, and the wiring diagram shows where pin 1 should go. The circuit is created on a stripboard and connected to a prototyping board, with the following two equations used to obtain the required resistor values [1]:

$$R_{LED} = \frac{V_S - V_{LED}}{I_{LED}} \tag{1}$$

$$R_{Collector} = \frac{V_S - V_{CESAT}}{I_C} \tag{2}$$

Fluorescent light was shined on the sensor to show that it is not sensitive to background light. An incandescent light was also used to show that the sensor is sensitive to sunlight. The dark current is the minimum value obtainable from the sensor and is measured by using a voltmeter with high input impedance and covering the sensor to block out all light [1]. First, the sensor was held above the white line at different heights in order to obtain the sensitivity at different heights. The measurements were obtained, as shown in figure 8, showed that the sensor is most effective at a distance between 0 mm and 1 mm, with a huge drop-off between 1 mm and 2 mm.

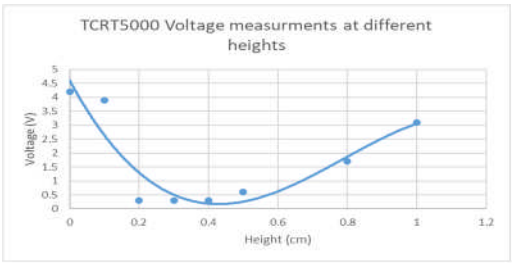

Figure 8 – A graph showing how the voltage varies with different heights above the ground for the TCRT5000 line sensor

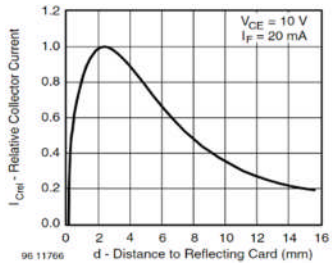

Figure 9 – A graph showing the current characteristics of the TCRT5000 while varying the height from the ground [9]

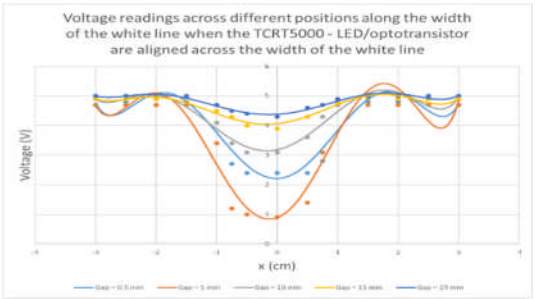

Figure 10 – A graph showing how the voltage varies as the sensor is moved while the TCRT5000 – LED/phototransistor are aligned along the white line

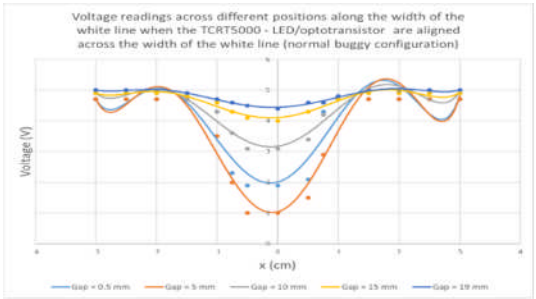

Figure 11 – A graph showing how the voltage varies as the sensor is moved while the TCRT5000 – LED/phototransistor are aligned across the width of the white line

Next, a test rig was used to find out the sensor's response as it is moved across the white line for different heights [1]. Figures 10 and 11 show the results obtained. The angular response of the sensor determines the width of the curves, and the higher the sensor is from the surface, the less accurate the values are, as seen in figures 10 and 11.

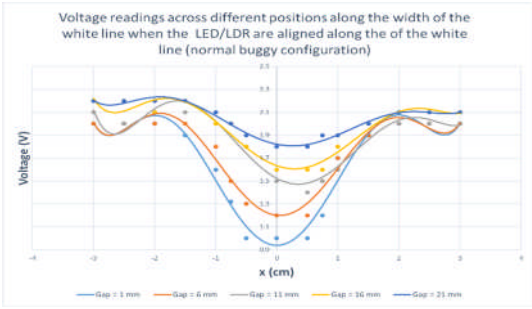

Figure 12 - A graph showing how the voltage varies as the sensor is moved while the LED/LDR are aligned across the width of the white line

This is repeated using an LED/LDR instead of the TCRT5000, and the results are shown in figure 12.

As previously stated, the measurements obtained show that the sensor is most effective at a distance between 0 mm and 1 mm from the surface, as can be seen in figure 8. However, the height measurement error is ±1.5 mm, which is extremely high based upon the small heights used. This makes the results obtained not very reliable, so the graph obtained from the datasheet, figure 9, should be more accurate in determining the preferred distance above the ground. The graph shows that the sensor is most effective 2.5 mm above ground. This tells us that the height measurement error had an effect on the results and made them unreliable, as they show that the sensor is not very effective 2 mm or higher above ground. This error can be overcome by using the graph from the datasheet as the main source of information for the optimal height above ground, by repeating the experiment multiple times, or by using a more accurate method to measure the height.

4. Circuit diagram for proposed line sensors

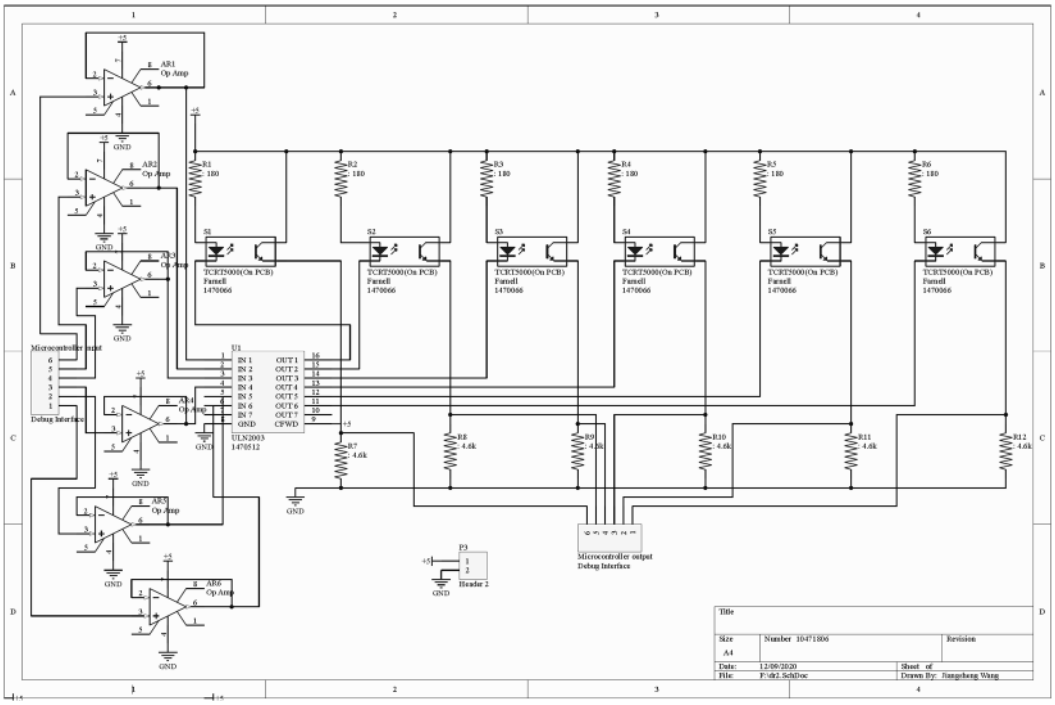

Figure 13 – Schematic diagram for the line sensor circuit detailing how the circuit will connect to the microcontroller

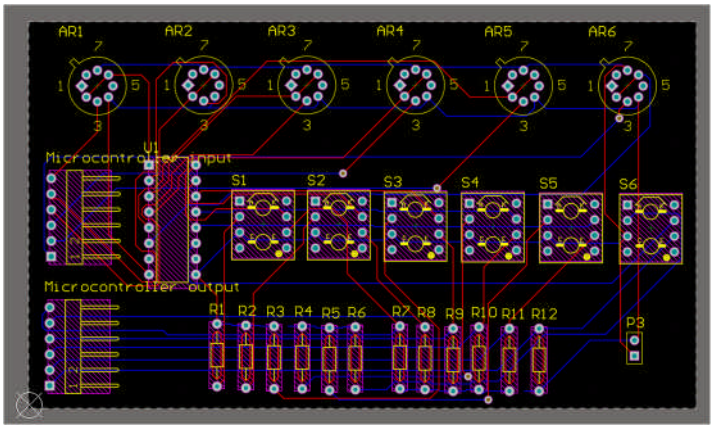

Figure 14 – PCB routing diagram showing the placement of the relevant components and how the tracks will be etched on the PCB

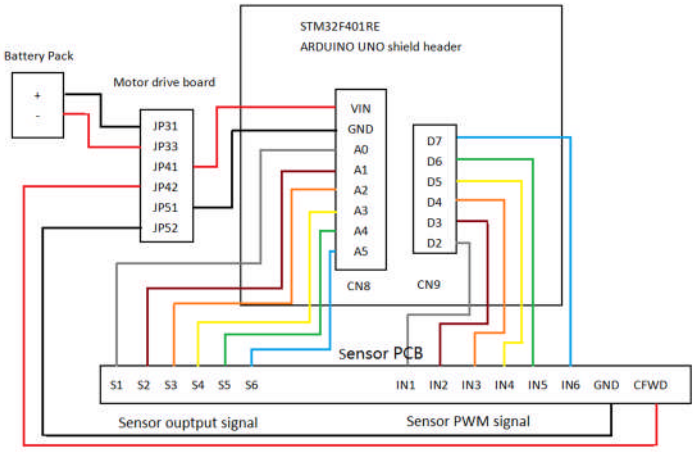

Figure 15 – Wiring diagram showing how the STM microcontroller pins connect to the Line sensor PCB pins

Figure 13 shows the circuit that will be connected so as to use the line sensors in the buggy to take the relevant line measurements. The values of the of the biasing resistors have been calculated using equations 1 and 2, where:

$$R_{collector} = \frac{5 - 1.4}{0.02} = 180\ \Omega$$

$$R_{LED} = \frac{5 - 0.4}{0.001} = 4.6\ k\Omega$$

The use of the ULN2003 IC is there to add a buffer to the input by means of a Darlington pair, so that the input can be switched at will. This allows for the sensors to be switched on independently from each other as there is not enough current supplied to turn all of the sensors on at the same time.

Figure 14 shows the design of the PCB. This shows how the components will actually be placed on the circuit. Here it can be seen that the sensors are placed in a horizontal arrangement; this is done so as to maximise the range of where the sensors scan the line.

Figure 15 shows how the pins of the microcontroller will be connected to the other PCBs and peripherals in the buggy. This helps when assembling the buggy when all of the parts have been fabricated.

5. Non line-sensors

There are five non-line sensors that will be used in the buggy, which are encoders for speed sensing, Bluetooth low energy (BLE) for the turn at the end of the track, current sensors for sensing the current at each motor and a current sensor to detect the battery level. In addition to the above given sensors, wall sensors will be used as they can be used to help avoid walls and allow for easier navigation when the buggy is turning in a tight space.

The speed of each wheel in the buggy is measured using an encoder disk (as seen in figure 16) in addition to a phototransistor [2]. In the buggy a quadrature encoder (as seen in figure 17) sensor will also be used to track the direction.

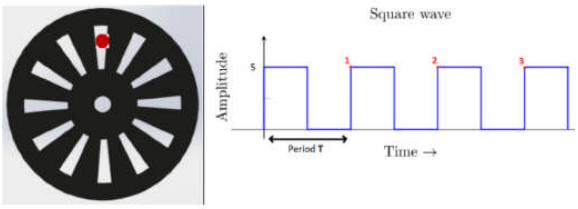

Figure 16 – Representation of an encoder and its output waveform [2]

The measurement setup is as follows [2]:

Step 1: initialise the timer

Step 2: Clear pulse counts

Step 3: read the change of pulse count after dt

Step 4: Reset and restart the timer and repeat steps 2 – 4

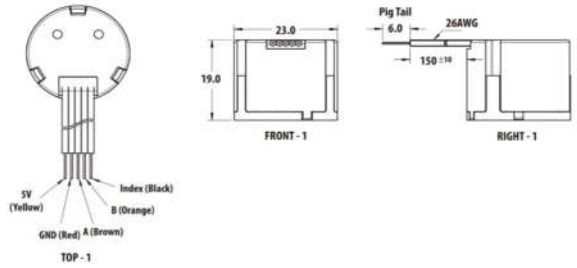

Figure 17 – Representation of the quadrature encoder used in the buggy [2]

The interfaces that can be used in the software are the Quadrature Encoder Interface (QEI) library and the GPIO InterruptIn library [2].

The 2nd type of non-line sensor used is the Bluetooth sensor. This is the standard for data exchange over short distances, with an operating frequency of 2.4 GHz [2] and a range of 100 m [13].

The BLE module that is going to be used for the buggy connects to the STM32 over the UART port. To configure them, a text command, called “AT commands” should be sent [2]. Bluetooth Low Energy (BLE) is a subset of Bluetooth (1st version was Bluetooth 4.0 [2]) that is designed to establish quick wireless links without consuming much power [2]. This is mostly applied to wearable devices in addition to the internet of things (IoT) and can be set up as a central-peripheral link for 2-way communication or as a beacon for 1-way communication [2]. In the buggy, 1-way communication is needed to control the turn by giving an external interrupt signal.

Having a fixed time interval between sending out advertising packets can have a great impact on the power consumption of the BLE [2]. If the BLE advertises the identity infrequently then it consumes very little energy, enabling the device to live for a very long time (years on end on a coin cell battery) [2].

The 3rd type of sensor that will be used in the buggy is a current sensor (configuration can be observed in figure 18). This sensor will be used to monitor the motor current and will be used to monitor the battery level [2].

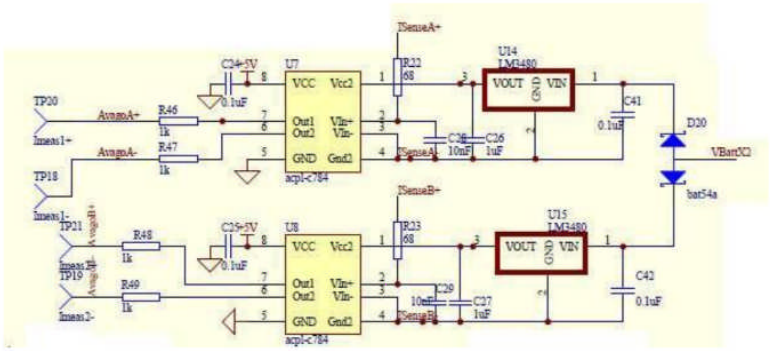

Figure 18 – Current amplifier circuit on the motor drive board

For current sensing, the current sensors are on the motor drive board and are in the form of two isolated current sensor circuits (one for each motor). These signals are connected to an isolation amplifier, pins 2 and 3 to create amplified outputs, from which the voltage readings are sent to pins 6 and 7 of the isolation amplifier [2]. The voltage measurements between TP18 and TP20 (TP19 and TP21 for the 2nd motor) can be calculated using the isolation amplifier datasheet [2].

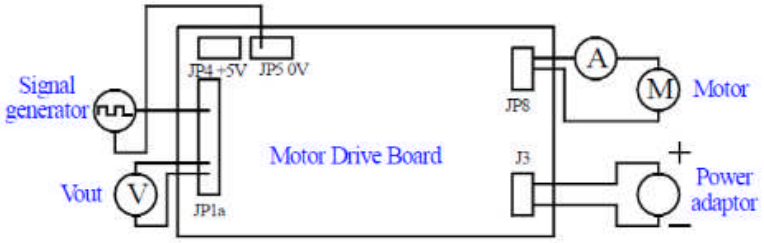

Figure 19 – Connections to the drive board for current sensor callibration

To calibrate the current sensor, the following procedure is done (using the circuit in figure 19) [2]:

Step 1. Supply the motor drive board with a known constant PWM input
Step 2. Add a voltmeter across the current sensor outputs while also measuring the current across the motor by using an ammeter
Step 3. Apply tensions across the motor and record the V-I characteristics of the motor

The 2nd type of current sensing that will be required is for the battery level. This uses the Dallas DS2781 integrated circuit to measure the voltage and the accumulated current of the battery [2]. Samples of the current i_k are made at regular intervals Δt, which are summed together [2].

The final type of sensor being used is the wall sensor HC-SR04. This was not one of the given sensors, however it holds the advantage of making sure that the buggy would not hit the wall and makes sure that the buggy is aware of the distance between the buggy and the wall [11]. This is to avoid any collisions of the buggy with the wall when the buggy is moving at high speeds. In essence, this is an insurance policy to make sure that the probability of the buggy colliding with the wall is even lower.

The wall sensor consists of two parts, which are the ultrasonic transmitter and the ultrasonic receiver [11]. The ultrasonic transmitter transmits an ultrasonic wave, travelling in air and when it gets obstructed by any material, it gets reflected back to the sensor [11].

At this point the critical distance can be set in the program to make sure the buggy can maintain more than that set distance away from the wall. For the buggy, 2cm seems to be a safe distance between the sensor and the wall.

6. Control

When the buggy is moving on the track, the track contains a kind of special white line and black surface, from which the direction of motion is dependent on the orientation of the white line [2]. The line sensors scan the white line and, transfer the signal to the controller of buggy so that that the wheel speed and direction can be changed by using a set algorithm [2]. Furthermore, if the buggy deviates from the white line, the controller will alter the motor torque enabling the buggy to return to the white line [2].

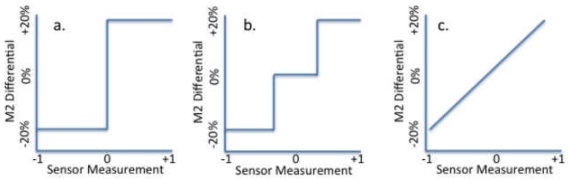

Figure 20 – Representation of Bang-Bang with a dead zone [2]

In Bang-bang control (step control), the range of the sensor measurements are between -1 and +1 (indicating if you are to the right or to the left of the line), and this will affect M2 differential (the difference between the sensor measurements). Bang-bang control causes the vehicle to zigzag across the white line which causes the M2 differential to change between the different motors (that has a range from -20% to +20% as seen in figure 20). This causes instability in the buggy when the change in the M2 differential is not too large. However, when Bang-bang is used with a dead zone, this issue can be avoided so that the stability of the buggy is increased; this is done by making sure that the buggy does not respond if the change in the M2 differential is small. However, when the curvature radius is small, the sensors cannot receive this information quickly enough so that the motors will alter their speed accordingly to make the turn (i.e., the sharper the turn, the harder to implement Bang-Bang as there are massive variations in the M2 differential). So, in general, this algorithm is best for straight lines where there are no turns where there need to be rapid response times for the algorithm to make the turn.

In Proportional(P)-Continuous Control, the range of the sensor measurement is the same as Bang-bang. The equation for the M2 differential in relation to error is given by:

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

Where $u(t)$ is the M2 differential, K_p is the proportional gain and $e(t)$ is the error

K_p is equal to the slope of figure 20(c) which directly dictates the stability of buggy's motion. The advantage is that each value of the sensor measurements corresponds to a different value of $u(t)$, so this method is efficient, and the buggy hardly deviates from the white line, where the motors will supply suitable torque. But the disadvantage is that the exact value of K_p needs to be determined and it is very hard to find a suitable value. A small value will lead to the buggy having to be move even further away from the line for corrections to be made, and a high value will cause large changes even when only small changes are needed.

In proportional control, it may be observed that the buggy will not be directly above the white line and will be shifted more over to one side of the line. This is due to the steady state offset. This can be eliminated by using PI (Proportional - Integral) control. As the name suggests, here proportional control is used as usual with the addition of an integral part, modifying equation 3 to become:

$$u(t) = K_p e(t) + K_i \int_0^t e(\tau)\, d\tau = K_p \left(e(t) + \frac{1}{T_i} \int_0^t e(\tau)\, d\tau \right) \quad (4)$$

Where K_i is the integral gain and T_i is the time required for the integral action to produce the same control action as the proportional action subject to $e(t) = 1$

Which in digital form can be approximated as:

$$u(t) = K_p e(t_k) + K_i \sum_{i=1}^{t} e(t_i)\, T_s \quad (5)$$

Where T_s is the sample time and k is the current sample number.

To avoid having to store the value of the integral, equation 5 can be re-written as:

$$u(t_k) = u(t_{k-1}) + e(t_k)\left[K_p + K_i T_s\right] - K_p e(t_{k-1}) \quad (6)$$

It is very important that T_s remains constant and that $e(t)$ is limited to sensible values to avoid issues in the operation of the control of the buggy.

This ensures that the algorithm continuously increments the output based on the previous output value.

However, the above algorithm is not necessarily smooth and to compensate for this differential control is used. To increase the smoothness of the operation of the control algorithm it is best to use a PID (Proportional – integral – Differential) control algorithm. This changes equation 4 one step further to become:

$$u(t) = K_p e(t) + K_i \int_0^t e(t^*)\, dt^* + K_d \frac{de}{dt} \quad (7)$$

Just as with PI control, equation 7 can be re-written as:

$$u(t_k) = \left[K_p e(t_k)\right] + \left[u_i(t_k - 1) + K_i e(t_k) T_s\right] + \left[K_d \frac{e(t_k) - e(t_{k-1})}{T_s}\right] \quad (8)$$

Equation 8 describes how the algorithm will be achieved in the buggy. Where k is incremented in a loop up to a maximum value.

However, a big problem with differential control is that it can also amplify the noise coming from the line measurements and so the effects are overwritten and no real improvements to the response occur.

All of the above control algorithms provide tangible benefits to the system and will in turn make sure that the buggy stays on the line without zigzagging across, the buggy will be cantered along the white line throughout and the control operation will occur as smoothly as possible. PID is the most suitable algorithm to use in the buggy as the factors that need to be controlled are the stability of the buggy due to the integral control of PID bringing the buggy to the centre of the line, and the proportional control provided a control to minimise zigzagging maximising stability and making the margin of error minimal unlike with big-bang where the error can be as great as +/- 20%.

To obtain a properly functional algorithm, manual tuning may be necessary. This involves the following series of steps (Zeigler-Nichols Tuning):

Step 1. Start with a low K_p where $K_i = 0$ and $K_d = 0$

Step 2. Increase K_p until the most persistent oscillations are observed and fix this K_p value (this becomes ultimate gain K_u)

Step 3. Increase K_i until steady state is appropriate.

Step 4. Slightly increase K_d (this should provide improvements to the system, but noise to the system is critical at this stage)

Step 5. If this does not satisfy the needs of the control algorithm then repeat steps 1 through 4 with a smaller selection of K_p

Step 6. Measure the period of the oscillations when K_u is determined and hence this period is called the ultimate period P_u

Step 7. Determine the values of the controller from figure 21

Ziegler-Nichols Tuning Method

Control Type	K_p	K_i	K_d
P	$\frac{K_u}{2}$	-	-
PI	$\frac{K_u}{2.2}$	$\frac{1.2K_p}{P_u}$	-
PID	$\frac{K_u}{1.67}$	$\frac{2K_p}{P_u}$	$\frac{K_p P_u}{8}$

Figure 21 – Values of K_u and P_u for the controller

From the above conclusions, PID is the better option for the control algorithm as bang-bang with dead zone can have errors if the curvature radius is small. If the scanning area of centre sensors are not parallel with the white line, the side sensors can revise the route to be corrected immediately. It is necessary to ensure that the buggy is stable and is scanning the white line periodically, so two or more sensors are needed. Also, the working voltage for these sensors should be the same, to ensure all the sensors obtain the same feedback to avoid unnecessary errors.

From section 2, the proposed sensors used in the buggy will be the TCRT5000. There will be six on the buggy for a wider range for measurement. These six line sensors should be aligned horizontally (as observed from figure 14) and should be 2.5 mm above the surface, this is where the maximum operation is achieved.

In the buggy, there is no gap required between the successive line sensors, seeing as a buffer (as shown in section 4) is used to control the sensors to be operated one at a time. This eliminates the concern of sensors interfering with each other.

These choices will make it so that the PCB design has to contain a gap of between each sensor, which will affect the overall size of the PCB. These sensors will be connected to the microcontroller via the ADC pins to ensure that the signals are optimised then will interface with the motor drive board for motion. However, this consideration can be avoided by the use of the buffer as seen in section 4.

There are two motors on buggy, and unipolar and bipolar are two configurations of the motors, when the buggy is moving forward/reverse or is breaking, (slowly or fast), the motors need change working state always. When turning left/right, the two motors are in different states, therefore a controller using the predefined PID algorithm is needed to calculate how much torque needed. Also, seeing as the motors may need to turn in reverse, the bipolar mode of configuration will be used.

When Sw1a and Sw2b or Sw1b and Sw2a (as seen in figure 22) are closed this is termed bipolar operation, where the direction of motion is determined by the direction of the voltage and the buggy will keep accelerating. Furthermore, changing the direction of motor movement will cause the motor brake regeneratively, so the magnitude of brake is more efficient and more effective. When Sw1b and Sw2b or Sw1a and Sw2a are closed this is termed unipolar operation, where the direction of motor movement can only go in one direction during operation. The buggy can brake by shorting the motor. In both cases the buggy turns by varying the force on the wheels, where bipolar can simply make a wheel go in reverse. In the buggy, bipolar operation will be used to ensure an easier turn and regenerative braking. These switches are controlled by PWM and varying them varies the voltage and hence the speed and direction of the buggy.

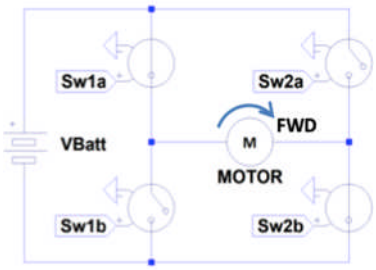

Figure 22 – Representation of switches on the motor drive board

Finally, there are some considerations in the buggy, which are mentioned in Section 2. For the buggy to stay on the line, an instruction can be set so that, when the sensor cannot scan any surface (white or black) in a small range of time (10 ms —100 ms), the buggy will continue with the last predefined motion until a new command is received. Furthermore, if the sensor detects infrared light for a long period of time (over 1 s), the buggy will decelerate and stop. To mitigate the effect of direct sunlight light, insulation can be used to shield the sensor to prevent detection. Another effective method is doing the experiment indoors so as to avoid direct sunlight and in this case, a light insulation shield is not needed.

7. Hardware overview

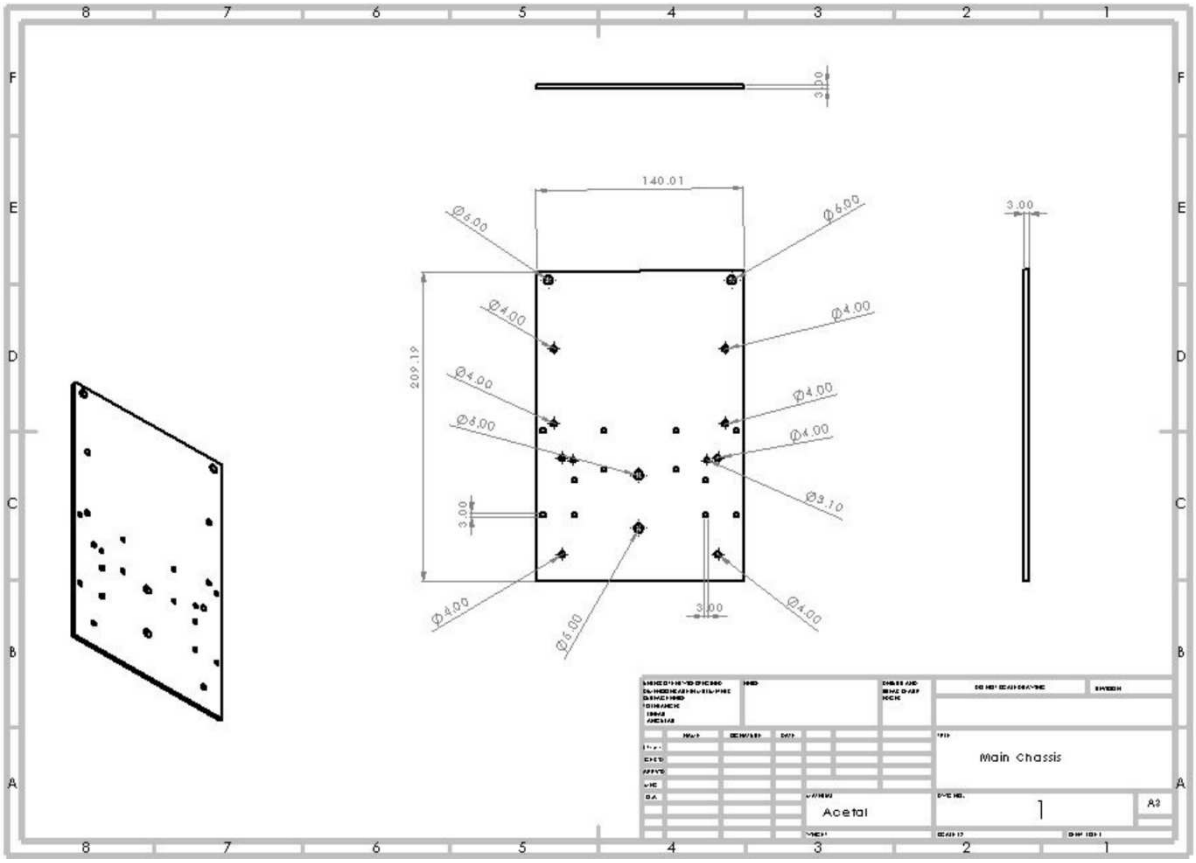

Figure 23 – Engineering drawing of main chassis

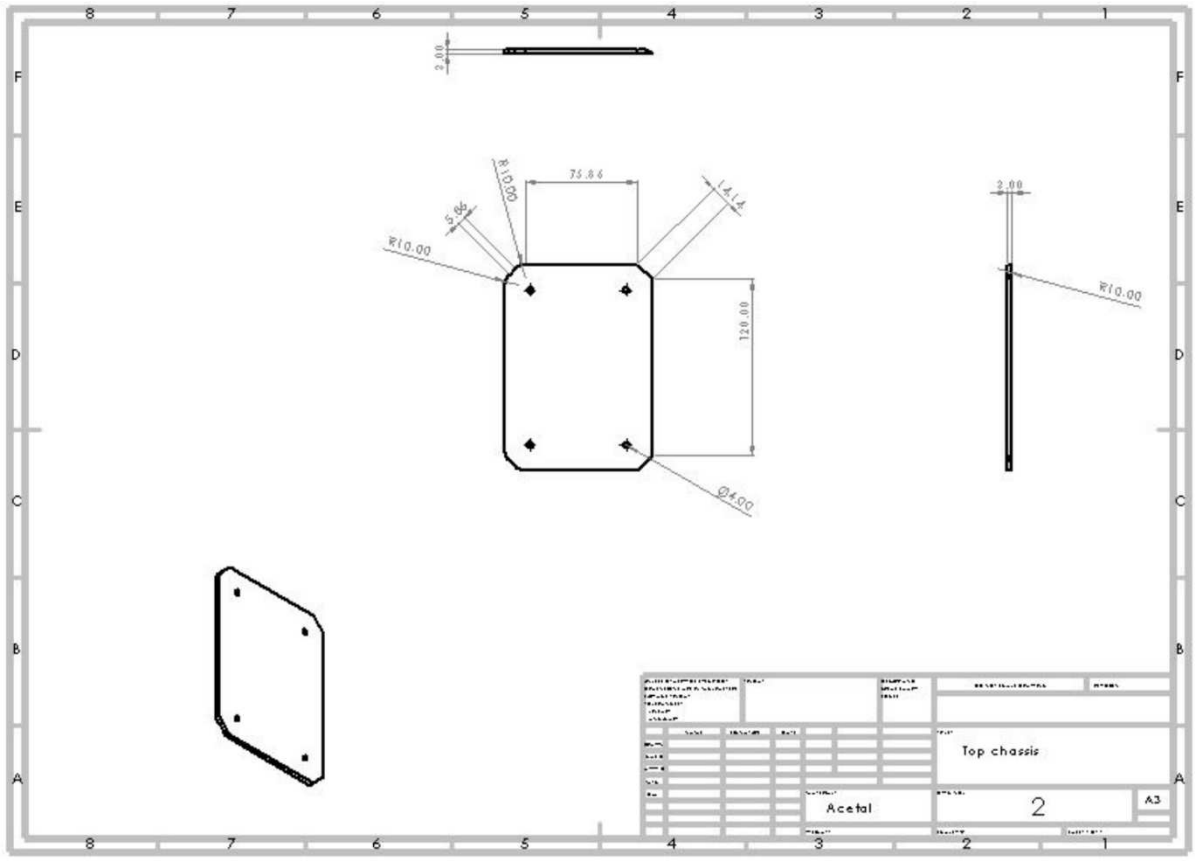

Figure 24 – Engineering drawing of top chassis

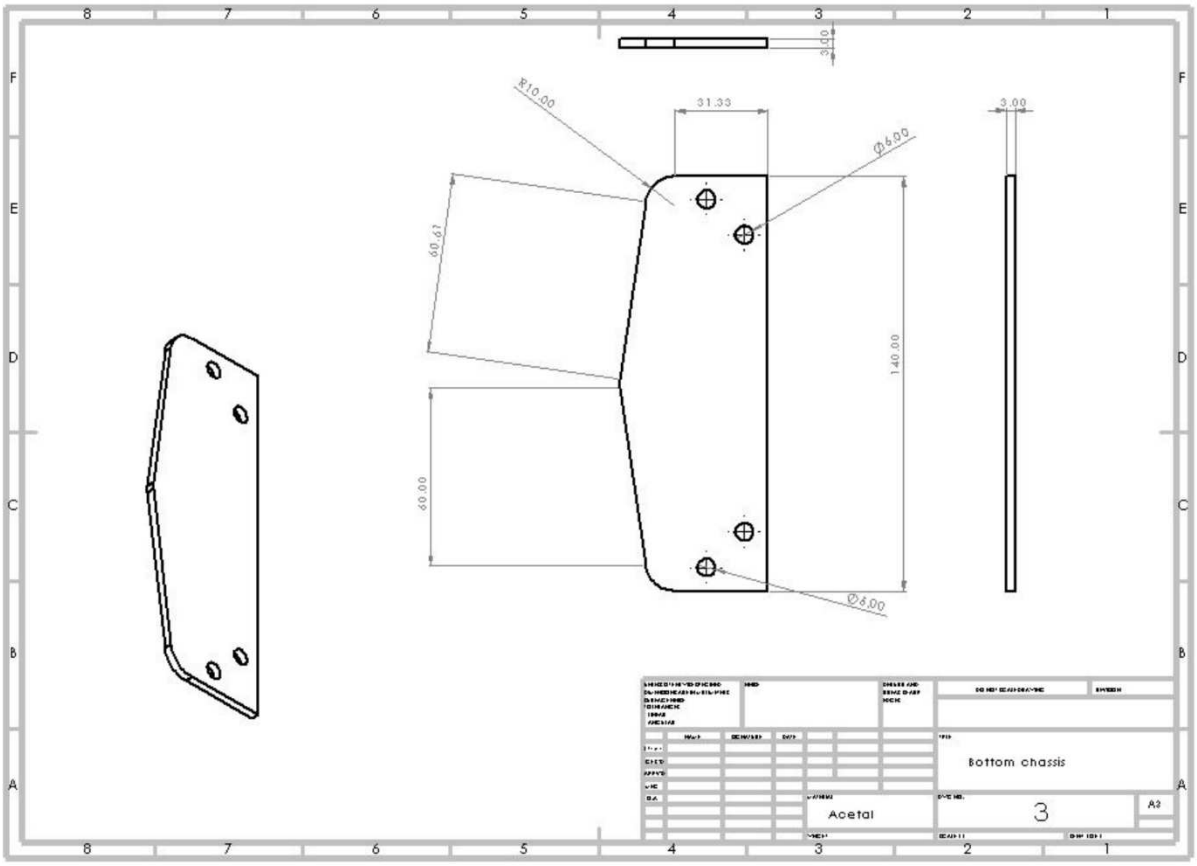

Figure 25 – Engineering drawing of bottom chassis

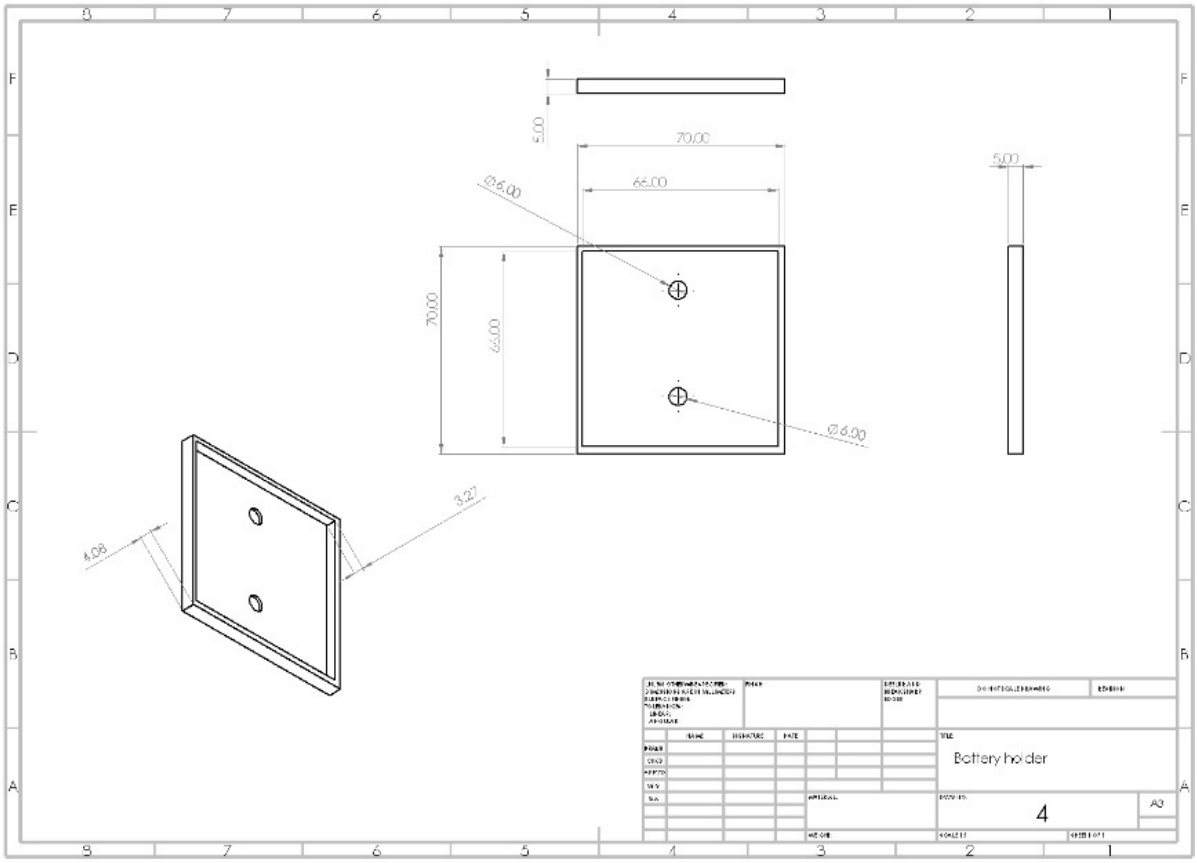

Figure 26 – Engineering drawing of battery pack

The chassis used for the buggy can be characterised into four layers, which are: the main chassis (figure 23), the top chassis (figure 24), the bottom chassis (figure 25), and the battery pack holder (figure 26). There are a few reasons for splitting up the chassis into the following parts. The main chassis is the core of the buggy and is what every part of the buggy connects to and is hence the largest piece. The top piece is there to hold the Nucleo board without extending the size of the buggy, so that it can manoeuvre the track in an orderly fashion where the buggy is more compact. The bottom piece is needed as the main chassis is very high up off the ground and so for the sensors to remain close to the ground at the required 2.5 mm, an additional part was needed. Finally, the battery holder is there to make sure that the battery pack stays in position securely and doesn't fall off the buggy.

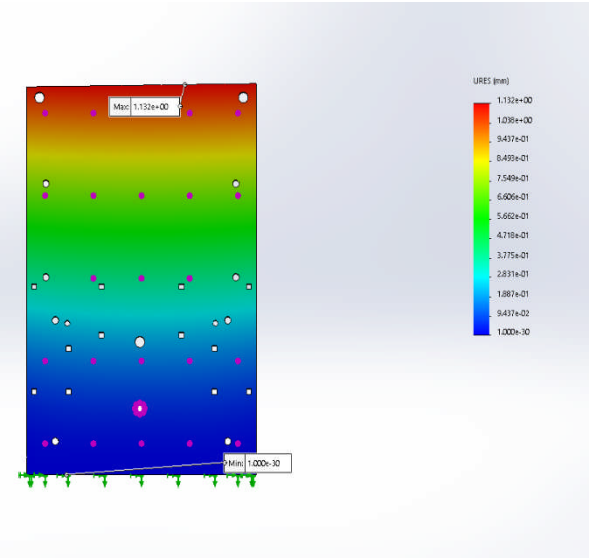

Figure 27 – Stress analysis of the main chassis

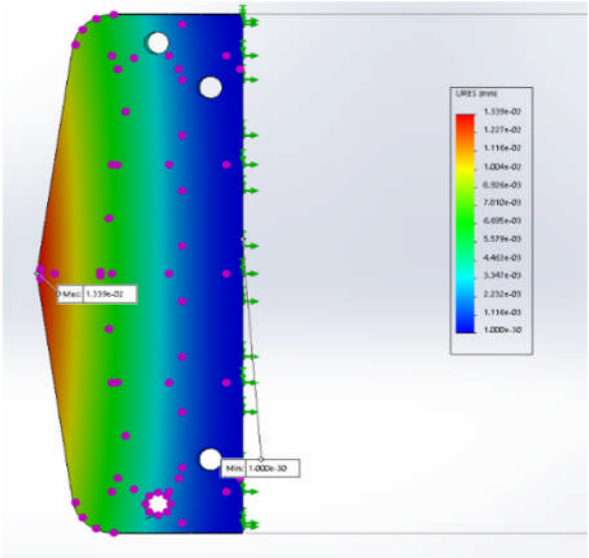

Figure 28 – Stress analysis of the bottom chassis

The following stress analysis shows that the chassis will have a very low deflection for acetyl, where the maximum value is at 1.132 mm for the main chassis (as seen in figure 27) and 0.01339 mm for the bottom chassis (as seen in figure 28). The other parts of the chassis are fixed from all four sides and so their deflections are negligible.

The manufacture of the corresponding parts should be easy. The main chassis is a rectangular piece with holes of the correct size drilled into it, the top chassis is a rounded rectangle with only 8 holes. The rounded edges may make be more difficult as they need to be of the correct curved radius. The bottom chassis only has four holes but has a rounded and pointy shape so may be harder to manufacture, while the battery holder only has two holes with a square shape, so should be easy to manufacture.

Table 3 – Comparison of characteristics of different materials

Material Characteristic	Acetyl	Glass-Reinforced 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

Acetyl was chosen as the material of choice because it has the lowest parameters (as observed in table 3), meaning that it could withstand more stress than the other materials and is less likely to bend as it is less elastic. It has the lowest density do is the lightest and will make sure that the buggy is light during operation. The thickness used is 3 mm which is shown to be acceptable from the stress calculations shown in figure 27 and figure 28.

The rear wheels are put in parallel and are put at the back to ensure that the centre of gravity is placed at the back, to ensure that the buggy can maintain control and is less likely to break. The front wheel is a castor ball which is light and compact. It is placed behind the sensors so that the sensors can track the line before motion. The castor ball is centralised and hence balances the buggy.

Figure 29 details how the final product will look like. The gearboxes will fit into the bottom of the main chassis, which will connect to the wheels.
The wheel spacing is 140 mm, so as to ensure that the weight is distributed over a bigger distance to make the force distributed over a larger space.

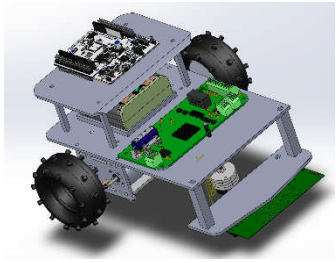

Figure 29 – Final construction of the buggy

8. Summary

In conclusion, in this report the experimental results were used to decide on using the TCRT5000 as the line sensor of choice, as well as design the circuit diagram and chassis, and decide on how to control and program the buggy.

TCRT5000 was the sensor of choice as it has both an emitter and a phototransistor in a small package of size 10.2x5.8x7 mm, as well as a coating on the phototransistor that blocks anything outside the infrared spectrum, and a barrier between the emitter and phototransistor to block the emitted waves from entering the phototransistor directly. Finally, since the emitter and phototransistor are in the same package, there are no concerns over whether they match together or not. The experimental results obtained show the angular response of the sensor at different heights, and that the TCRT5000 works optimally at a height of 2.5mm.

The PCB was created and equations 1 and 2 were used to calculate the resistor values required and a Darlington pair was added to the PCB to control which sensors turn on. A wiring diagram was created to show how the microcontroller is connected to other peripherals in the buggy, as well as schematic diagram was created detailing how the sensor circuits connect to the microcontroller.

There are five types of sensors in addition to the line sensors, encoders for speed sensing, Bluetooth for receiving signals telling the buggy to make to make a U-turn, current sensors for measuring the motors current, battery level sensor, and a wall sensor for walls detection.

The function of the buggy was summarised in four stages, there are seven possible constraints for this. The messages from the microcontroller to the peripherals and vice versa as the light level received and the Bluetooth signal has been included in a context diagram and detailed in Table 1. Initialisation, take light measurements, take wall measurements, and motion are the four are the main functions of the program, a function prototype has structured for each of them.

PID was the control algorithm of choice, as Bang-Bang is not suitable. Bang-Bang makes the buggy move in a zig zag motion, while PID will make it centred above the white line. Furthermore, PID will increase stability and will result in more stable turns.

A chassis of dimensions 209 by 140 mm was created, with a smaller second layer above the battery pack of size 120 by 76 mm. This is to ensure that the buggy can rotate in the track provided. The sensors are located in front of the castor ball, as it was decided that the sensors should detect the white line before the castor wheel moves over it.

9. References

[1] L. Marsh. Procedures Handbook. (2020 Winter). EEEN21000 Embedded Systems Project. University of Manchester. United Kingdom

[2] L. Marsh. Technical Handbook. (2020 Winter). EEEN21000 Embedded Systems Project. University of Manchester. United Kingdom

[3] Vishay, “Silicon NPN Phototransistor,” BPW17N datasheet Mar. 8, 2008 [Last revision Apr. 8, 2008]

[4] Osram, “Silicon PIN Photodiode with Very Short Switching Time,” SFH203P datasheet Feb. 22, 2001 [Last revision Feb. 22, 2001]

[5] Vishay, “Silicon NPN Phototransistor,” TEKT5400S datasheet Aug. 23, 2011 [Last revision Oct. 2, 2012]

[6] Vishay, “Infrared Emitting Diode, 950 nm, GaAs,” TSKS5400S datasheet Oct. 2, 2012 [Last revision Sep. 3, 2013]

[7] Multicomp, “5.0mm Round LED Lamp,” OVL-5521 datasheet May. 25, 2012 [Last revision May. 25, 2012]

[8] Vishay, “Infrared Emitting Diode, 875 nm, GaAlAs,” TSHA6200 datasheet Aug. 24, 2011 [Last revision Oct. 2, 2012]

[9] Vishay, “Reflective Optical Sensor with Transistor Output,” TCRT5000 datasheet Jul. 2, 2009 [Last revision Oct. 2, 2012]

[10] “Solar and Sustainable Energy, Sunlight.” Ag.tennessee.edu. Available: https://ag.tennessee.edu/solar/Pages/What%20Is%20Solar%20Energy/Sunlight.aspx#:~:text=Much%20of%20the%20energy%20from,%2C%20and%2010%25%20ultraviolet%20light (accessed: Dec. 6, 2020)

[11] E. J. Morgan, “HC-SR04 Ultrasonic Sensor,” HC - SR04 datasheet Nov. 16, 2014 [Last revision Nov. 16, 2014]

[12] A.S. Falcon “HM10 Guide”, Mbed, 23 August 2017. Accessed: 05 December 2020. [online]. Available: https://os.mbed.com/users/alexsaadfalcon/notebook/hm10-guide/

[13] HM-10 Bluetooth Module, Components 101, 25 September 2018. [Online]. Available: https://components101.com/wireless/hm-10-bluetooth-module#:~:text=The%20module%20is%20used%20for,very%20less%20power%20to%20function. (Accessed: 09 December 2020)