4 Hardware Design

4.1 Introduction

Prior to our team's involvement in the project, an off-the-shelf solution was used for the robot. In particular a modified AlphaBot [16] platform was used, with added on LiDAR and swapped out motors. However, Alphabot costs around £150 and is mainly only used for the chassis. Hence it is not the most efficient solution. Because of this it was decided to spend a little time on a simple custom solution that would be built in-house. The benefits of this approach are mainly:

1. Reduced cost.
2. Ease of manufacturing – with very few additional mechanical parts required it is easy and cheap to build multiple units
3. Improved or customizable options. For example, one of the complaints of Alphabot was poor battery life, which can be increased in design process.

4.2 Electrical Design

4.2.1 Raspberry Pi Power Usage

In order to work out the power budget, power usage of each component must be known. While the current draw for other components of the system can be found within literature, the power draw of the Raspberry Pi was to be measured. Also, in order to maximize battery life of the system the power draw from the Raspberry Pi must be reduced as much as possible. Luckily there are a few options on how to do that when running a headless setup. The following options can be used:

1. Turning off the USB and ETH chipset – since none of the USB ports are used, and the internet connectivity is provided through Wi-Fi.
2. Turning off the HDMI output – video output is not used in headless setup.
3. Turning off the Bluetooth - also not used.

It was decided to measure the power draw of the Raspberry Pi with each of these options individually in order to see how much power can be saved. This was done by powering the Raspberry Pi with a lab bench power supply and measuring the supply current. The tests were performed in two states, first – when Raspberry Pi not doing anything, i.e., idle current, second – when loading the CPU, i.e., stress testing. The stress testing was done by downloading and using "stress" package, which is readily available. The results can be seen in the Table 1.

Configuration	**Value**	**Stress test current, mA**	**Difference, mA**
Normal operation	420	650	-
USB and ETH disabled	190	420	-230
Just HDMI disabled	400	630	-20
Just BT disabled	405	635	-5
Everything disabled	175	405	-245

Table 1: RPi current draw

4.2.2 Power Budget

Now that the maximum current of the Raspberry Pi has been determined a power budget can be constructed. The components of the system and their according supply voltage and maximum current draw can are summarised in the table below. Most of the components are powered from the 5V rail, except for the LIDAR

motor, which is powered straight from the battery at nominal voltage of 7.4V. It can be seen that the largest contributor to the power draw is the Raspberry Pi.

It was determined that the 5V rail must be able to supply at least 1.4A of current, with additional 60mA straight from the battery.

Device	Voltage rail	Max current draw, mA	Power draw, W
Raspberry pi	5V	650	3.25
Servo motors	5V	2x200=400	2
LIDAR	5V	330	1.65
Total	5V	1380	6.90
LIDAR (motor)	7.4V (Vbatt)	60	0.44

Table 2: Power budget

4.2.3 Power supply

5V converter For the 5V rail it was decided to use an off-the-shelf DC-DC buck converter. The one chosen was "DollaTek XL4015", which has adjustable output voltage and output current capability of 5A. This converter is cheap and readily available and provides plenty of headroom for additional current draw of the system.

Battery The system was to be powered from a rechargeable battery. For this a Li-Ion battery was chosen. In particular it was the "RS PRO 7.4V Li-Ion 5.2Ah". The battery stores roughly 38.5Wh of energy.

The running time of the system can be estimated as follows. Total power draw from the 5V rail is 6.9W, or 5.68W if the functions such as USB, HDMI and BT are disabled on the Raspberry Pi. Assuming 90% efficiency of the buck converter the total power draw of the system then is:

- 6.9W/90%+0.44W=8.11W
- 5.68W/90%+0.44W=5.70W

The robot should run 4.7 hours with no power savings and 6.8 hours with USB, HDMI and BT functions disabled when using the selected battery.

4.2.4 LiDAR

The object-sensing is performed by Slamtec's LiDAR (Light Detection and Ranging) RPLIDAR A1M8 [17]. This sensor was recommended by the team supervisor since the low-level setup for the LiDAR had already been successfully built, prior to the commencement of the herein described project. The low-level setup includes instructions on how to connect the LiDAR directly to a RaspberryPi using its in-built serial port, program code in C++ reading the coordinates of the objects detected by the sensor as well as PWM control of the motor, which rotates in clockwise direction the ranging core of the LiDAR and hence enables a 360° scan of the surrounding environment. Figure 5 represents a 360° of a room, where the LiDAR is considered to be positioned at the origin.

4.2.5 RPLIDAR A1M8 Sprecifications

RPLIDAR A1M8 is a 360° omnidirectional laser scanner, based on the triangulation ranging technique that emits modulated laser signal in the near-infrared (NIR) light band (typ. 785nm) and this signal is then reflected by a detected object [17]. The triangulation ranging technique, involves the laser diode emitting a collimated light beam on the surface under test, which is reflected by the surface and is focused onto the image sensor, by means of lens [18]; the position of the light on the module's image sensor enables

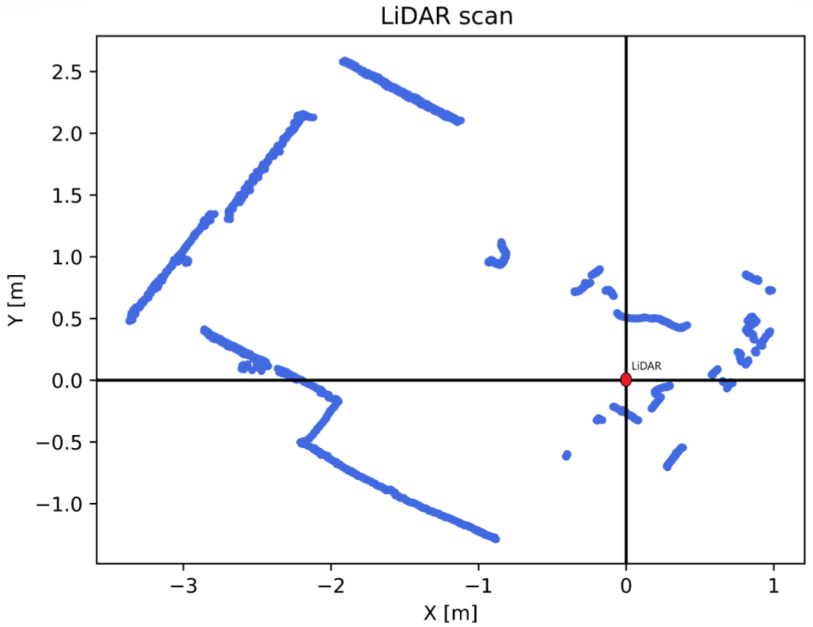

Figure 5: 360 ° scan of a room with 10k sample points. The LiDAR has been centered at the (0,0) origin. X position has positive/negative values when the detected object is in front or behind the robot, respectively. Y position has positive/negative values when detected object in on the left/right of the robot, respectively.

calculation of angle from which distance can be derived by the embedded DSP unit. The LiDAR ultimately outputs distance in meters, x-y coordinates of the detected object with respect to the LiDAR, heading angle measurement and signal strength that are all transmitted through the communication interface.
In this project the UART communication interface was used to transmit LiDAR data to the RaspberryPi rather than the alternatively provided USB, due to the comparative simplicity of the communication protocol and the transmitted data packets, which in turn ensures a quicker data transmission.
The LiDAR's motor is powered directly from the battery (7.4V), while the scanning systems is powered by regulated 5V volts, in order to meet the manufacturer's requirement.s In this project a constant scan rate of 5Hz (i.e. 300 rotation per minute) was set. Yet, if the motor system was under-powered ($V_{supply} < 5V$), then the LiDAR would not be able to achieve the preset scan rate. Regarding the range of the LiDAR, test results showed that the it is able to detect obstacles as close as 10cm and as far as 11m away from it, and with the values provided in the data sheet: it is able to Key specifications of the employed LiDAR are summarised in Table 3.

Feature/Specification	Value
Scan rate [Hz]	2 - 10
Sampling frequency [Hz]	> 2000
Range[m]	12
Distance Resolution [mm]	< 0.5
Angular field of view [°]	0 - 360
Sampling Duration [ns]	500
Peak LASER power [mW]	3 - 5
Wavelength [nm]	775 - 795
Communication Interface	UART/USB
Supply Voltage Motor System [V]	5 - 10
Supply Voltage Scanner System [V]	4.9 - 5.5
Total Current Consumption(= scanner + motor systems) [mA]	400 (= 300 + 100)

Table 3: Characteristics of RPLIDAR A1M8 [17]

4.3 Continuous Rotation Servo Motors

To make the robot movable, two Parallax continuous-rotation(CR) servo motors [19] were used. Their specifications have been summarised in Table 4.

Feature/Specification	Value
Supply Voltage [V]	4 - 6
Control Signal [V]	3.3 - 6.2
Operating current (no load - stall current) [A]	0.15 - 1
Torque [N.m]	0.27

Table 4: Characteristics of Parallax Continuous Rotation Servo [19]

CR servos are modified servos that offer open-loop speed control instead of the usual servos' closed-loop position control. This modification turns them into DC motors with integrated motor driver, which in turn enables the bidirectional rotation. Both the rotation speed and direction of the servo shaft of each motor is controlled by a pulse width modulated (PWM) signal. Yet, the lack of motor encoder or speed sensor to provide speed feedback makes the CR servos prone to differences between desired and actual speeds (more details in Section 4.3.1). Both motors have been powered by a 5V regulated signal generated by the buck converter. The control signals of the motors are 50Hz PWM signals, generated by two RaspberryPi GPIO pins, where the higher the PWM duty cycle, the higher the supplied powered to the motors and the faster the motor rotation would be.
When tested, the motors exhibited non-linear response of speed to applied PWM duty cycle, as shown in Figure 6. Second-order polynomial equation has been therefore derived to describe their relationship (see Equation 1). In program code the PWM duty cycle needed to achieve a required speed is found by calculating the positive root of Equation 1.

$$Speed\left[\frac{m}{s}\right] = -0.0773(PWM)^2 + 0.1787(PWM) - 0.0075 \tag{1}$$

where PWM duty cycle $\in [0, 1]$ (with 1 corresponding to 100% duty cycle, resulting in maximum speed).

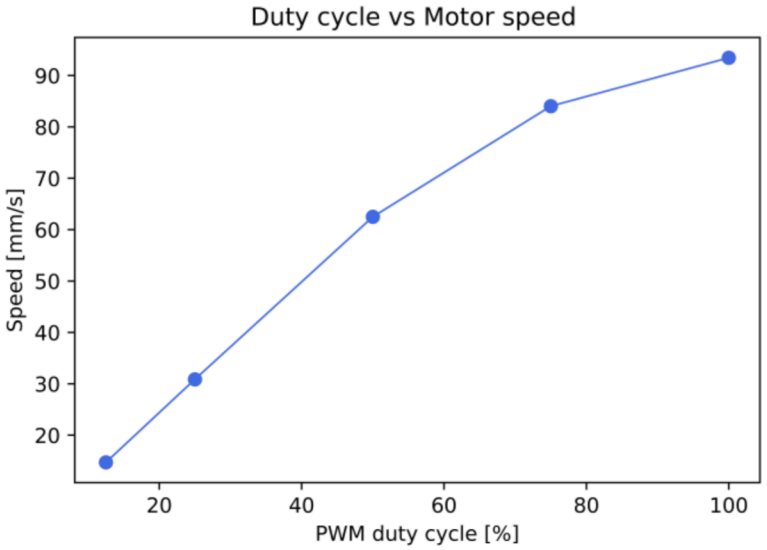

Figure 6: Motor speed as a function of PWM duty cycle; non-linear response to PWM

4.3.1 Synchronisation of servos

In the initial stages of building the robot, the center position of each motor was calibrated in order to define the pulse width value at which the servo holds still. If both motors are identical, have been calibrated and equal-duty-cycle PWM signals (and hence equal power) have been applied to them, then the robot would ideally move in a straight line. In spite of the calibration, the CR servos used in this project exhibited slight differences in speed when the same power was applied to them, which in turn resulted in the robot drifting from the straight path. Potential reasons for this are motor coil resistance variation and uneven surface. Three methods were considered to compensate for the differences:

1. Implement rotary/motor encoders (based on hall-effect sensors) + feedback control loop; the encoders would be mounted to each motor/wheel and the speed of each motor would be determined from encoder rotation count. A feedback loop could be modelled to compare the computed speed (from the encoder) with the desired speed, and increase or decrease the PWM duty cycle according to how far it is off.

2. OpenCV (for accurate robot position determination) + feedback control loop to adjust the PWM signals of the motors in order to get to a reference position.

3. Compensation in program code by keeping the control signal of one of the motors constant, while adjusting the signal of the second motor until straight movement is achieved.

Method 3. was implemented in this project due to small computation time and not requiring additional hardware. Since the differences were not linearly related, offset values for five PWM duty cycles were found and then extrapolation between these points was done (see Figure 7); then a second-order polynomial equation was derived from their relationship:

$$offset = 0.0976(PWM)^2 - 0.0297(PWM) - 0.0387 \tag{2}$$

The added offset values allowed for keeping a straight path for $\geq 2.5m$, which was deemed acceptable. Yet, if a more precise and stable performance is required, either method 1. or 2. should be implemented instead, since they would involve feedback loop to continuously adjust the servos' control signals as required.

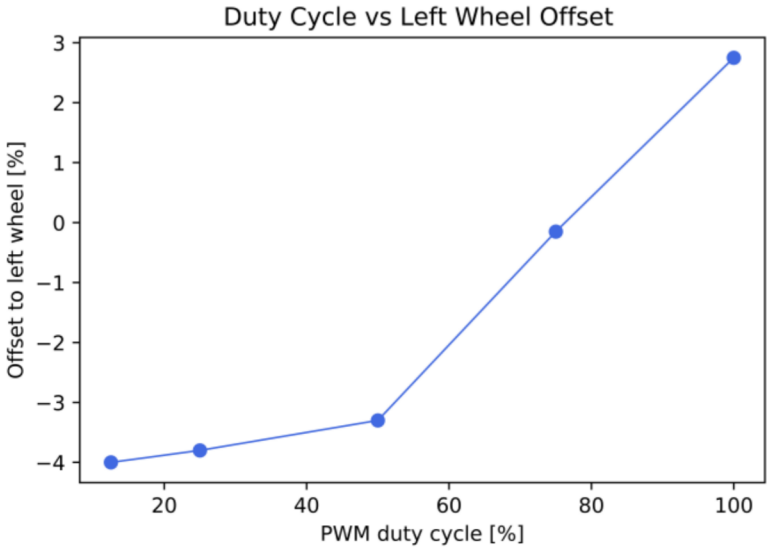

Figure 7: Offset to left wheel as a function of PWM duty cycle; non-linear response to PWM

4.3.2 Full Schematic

Apart from the power supply, connectors for the LiDAR and motors had to be implemented and wired to the Raspberry Pi. The full schematic can be seen in Figure 8.

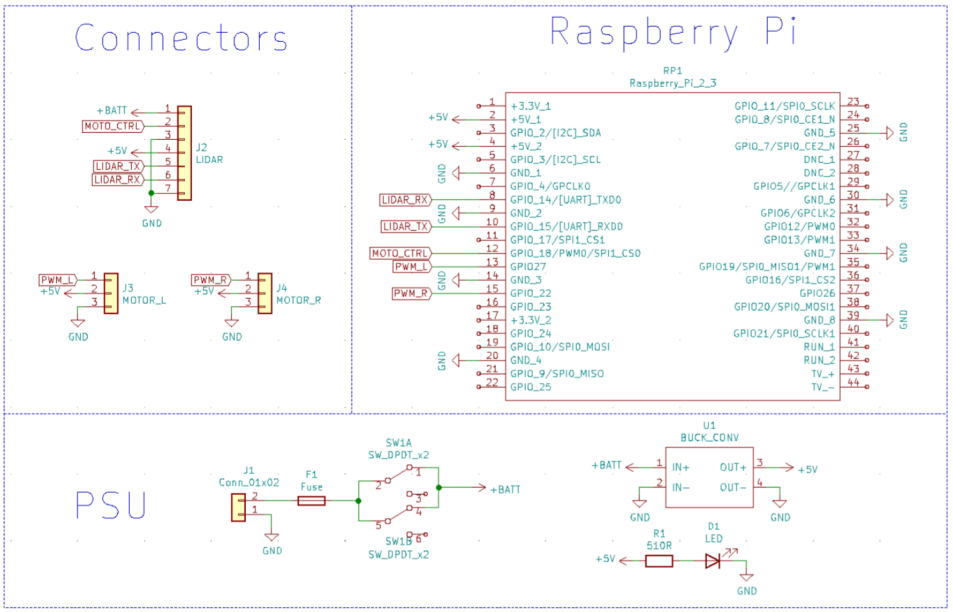

Figure 8: Full Schematic

4.4 Mechanical Design

The idea of the mechanical design was to make something simple and easily and cheaply manufacturable. It was decided to use the main PCB not only for electrical connections but also as the chassis of the robot. With that in mind the PCB was laid out while keeping to the similar dimensions of the Alphabot. A 3D render of the PCB can be seen down in Figure 9.

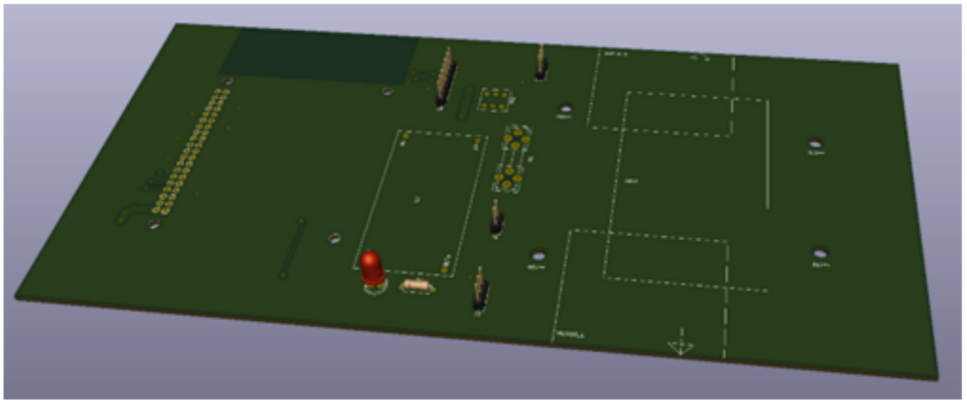

Figure 9: PCB layout

4.4.1 Other Mechanical Parts

There are only few other mechanical parts required. Namely the wheels, the ball castor, and some custom 3D-printed fittings. The chosen wheels were Pololu 1090 Wheels with dimensions of 42x19mm. In order to fit these wheels onto the servo motor shafts a custom wheel adaptor was 3D-printed which can be seen in Figure 10. Mounting brackets for the servo motors were also 3D printed.

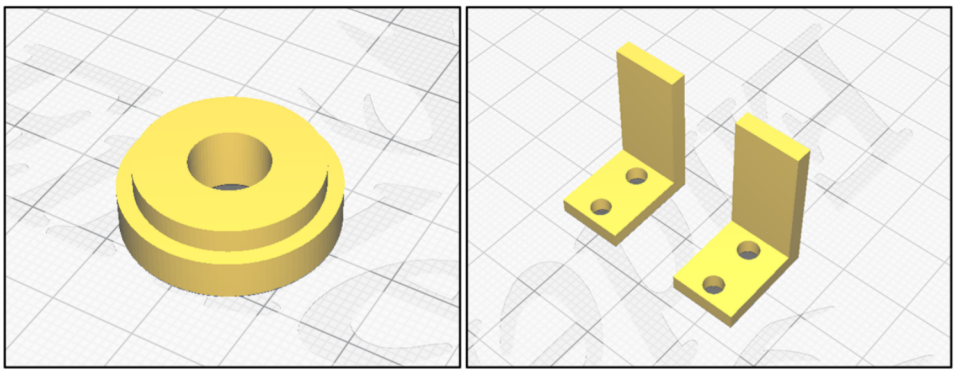

Figure 10: Wheel adaptor (left), servo brackets (right)