Wrocław University
of Science and Technology

Line follower robot

Rafał Cielenkiewicz (263561)
Aleksander Pauch (264352)

Supervisor : Prof. Janusz Jakubiak

Wrocław 2023

Contents

1 Project Description

To begin, our project revolves around the development of a line follower robot. The main objective is to design a robot that can navigate through a predefined path by following a line using some sort of sensors. In our particular case, we opted for IR (Infrared) sensors with an Arduino Uno being the main micro-controller to process signals from our sensors.

We got inspired to take this challenge due to the ongoing development of self automation as well as advancements in AI. The world is at a stage where we see constant progress in things like self-driving vehicles using sensors like LIDAR etc. This technology is a lot more advanced that our project, but we believe our creation can be seen as a prototype or the fundamentals to more advanced self driving vehicles.

An inspiration closer to our project is the HBFS Line follower. It is a line follower that pushes the potential of using IR sensors, as the purpose of this robot is to finish time trials around a predefined track. This is achieved by choosing the best of the best equipment to maximise efficiency, whilst avoiding time loss. Since the HBFS line follower regularly competes in events around the world, it's purpose is different to ours. Our project is deemed successful when the line follower robot can follow the path without going off course, whilst the HBFS's objective is to do it as quickly as possible. Although there is a difference between the complexity of these projects, our project is a building block in case we ever did want to compete.

This report contains the necessary information on the development cycle of our product, as well as the description on why a component was chosen. Each element of our robot must work seamlessly together to ensure precision when following the track. To do this, we need to correctly calibrate components and test them so that any problems are minimised.

2 Theoretical Introduction

The mechanical theory of a line follower robot involves principles to ensure efficient navigation along a predefined path. Kinematics, focus on motion without considering forces, explore how the robot's wheels move in response to motor commands. Dynamics consider the impact of forces and torques on the robot's motion.

One critical aspect of the robot's mechanical design is its wheel configuration. Different configurations of wheels allow for different turns and maneuvers. This ranges from the type of wheels used, the number of them, and where they are placed on the design.

Material selection is crucial, with a preference for lightweight and durable materials. These choices not only reduce the overall weight of the robot but also enhance energy efficiency.

Balancing the robot involves understanding its center of mass, with a well balanced design contributing to stability, particularly during turns or sudden changes in direction. These mechanical principles converge to create a reliable and effective line follower robot capable of accurate navigation along its designated path.

Our project is grounded in the fundamental operation of infrared (IR) sensors, specifically an emitter and collector circuit, commonly referred to as an optocoupler. The emitter composes of an IR LED, and the collector houses an IR photodiode. The interaction between these components forms the basis of our path detection mechanism. When the photodiode detects IR light, its resistance and voltage change proportionally to the received amount.

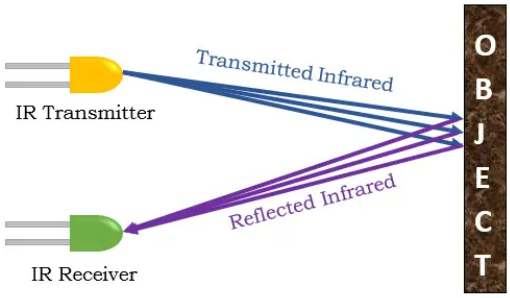

Figure 2.1: Optocoupler configuration with IR LED and photodiode.

The phenomenon exploited for path detection with IR sensors rely on the varying reflectivity of different colors and surfaces. The contrast in reflectivity, analogous to the distinction between black and white surfaces, facilitates discrimination. Dark surfaces may absorb more IR light, resulting in reduced reflection, while white surfaces tend to reflect more IR light. This contrast in sensor readings enables the discrimination between dark and light surfaces.

The calibration process for IR sensors involves determining the threshold values that distinguish between dark and light surfaces. This process ensures that the robot responds appropriately to the variations in surface reflectivity. It is crucial to conduct testing to tune these thresholds, taking into account potential challenges in different scenarios.

After the sensor and path type has been chosen, the next step involves reading sensor values and adjusting the robot's behavior accordingly. Different sensors communicate in different ways, this can include: digital readings, analog values, wireless communication etc. Using this information, a main microcontroller can be chosen that facilitates this form of communication to allow control sequences based on the data from the sensors. An example of the type of data returned by an IR sensor is in figure 2.2.

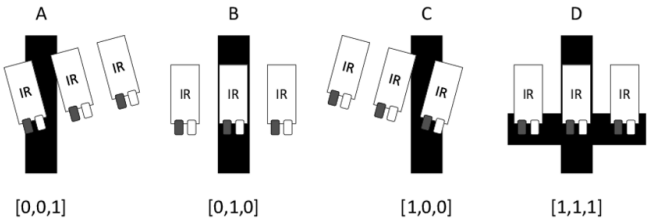

Figure 2.2: Sensor values returned by IR sensors.

3 Assumptions

Below are the assumptions made during our project:

Functional assumptions are related to what the system is intended to do and how it should perform certain functions.

Design assumptions involve decisions made during the design phase and expectations about how certain elements will be structured.

Table 3.1: Functional and design assumptions of the line follower robot:

Functional Assumptions	**Design Assumptions**
Line Detection	Line follower uses infrared (IR) sensors for line detection. It is assumed to accurately detect and follow a predefined line on a surface.
Communication and signal processing	Arduino UNO will process analog signals and facilitate communication between sensors and the control system.
Speed Adjustment	The closed-loop control system implemented will adjust the motor speeds to maintain the robot's alignment with the line, ensuring smooth and accurate navigation.
Power Management	The line follower will be powered by two Li-ion batteries, connected in series to provide sufficient voltage and current for the motors and electronics.
Chassis Design	Custom 3D printed chassis will be designed to house the components securely, ensuring optimal sensor positioning, weight distribution, and structural integrity.
Integration Testing	Validation of the functionality of sensor readings, motor control, and the overall responsiveness of the line follower robot.

4 Description of the hardware parts

This section provides a concise overview of our robot's hardware, covering key elements such as the Arduino Uno microcontroller, DC Motors with L293D Motor Driver Shields, Infrared Sensors for line detection, a custom-built chassis, and Samsung Rechargeable Lithium-Ion Batteries. It includes block diagrams, photos, and details on mechanical components, electrical schematics, and printed circuit boards.

4.1 Block diagram

Below a diagram to better understand the general ideas and process of building the robot. (Figure 4.1)

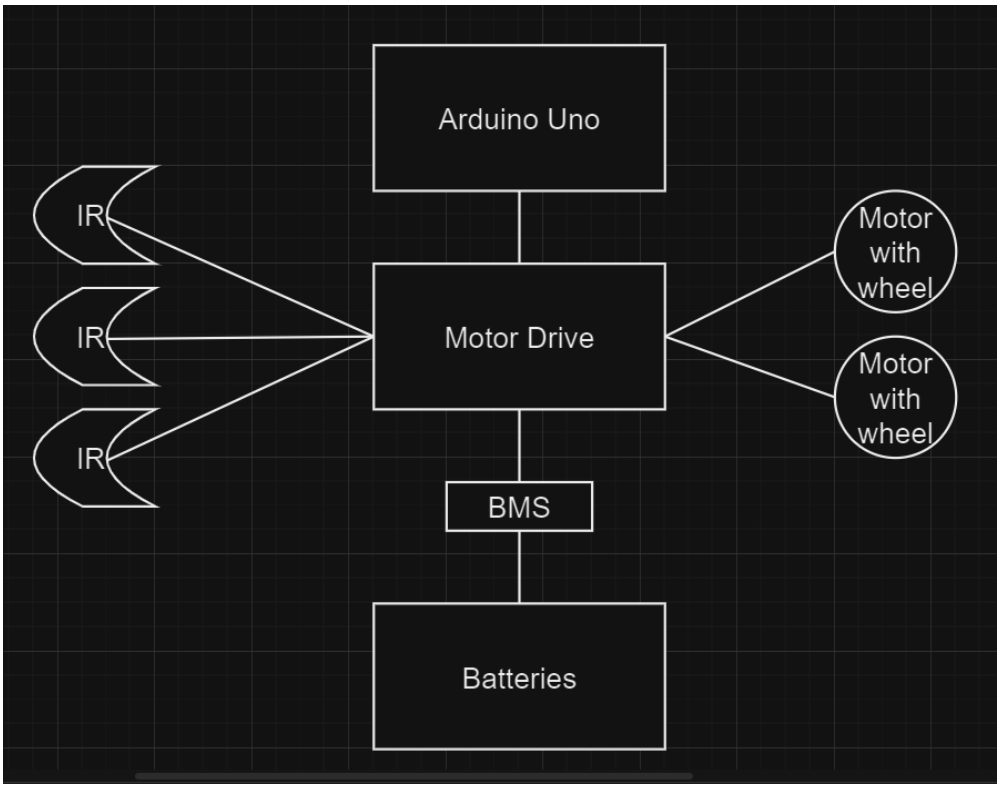

Figure 4.1: Block diagram for hardware

Microcontroller (Arduino Uno) [9]:

Facilitates program execution and interfaces with various hardware components.
User-friendly interface and versatile nature for controlling and coordinating robot operations.

Motors (DC Motors with L293D Motor Driver Shcclds):

DC Motors [2] integrated with L293D motor driver shields [4].
Converts electrical energy into mechanical motion, allowing precise control over the robot's movement.
L293D motor driver shields provide bidirectional control, regulating speed and direction.

Sensors (Infrared Sensors for Line Detection):

Infrared sensors designed for detecting surface contrasts.
Enables the robot to follow or detect predefined paths.
Interpret IR light reflections to identify lines or track patterns, guiding robot navigation.

Chassis (Custom-Built):

Custom-built chassis provides a purpose-designed framework.
Houses and securely mounts all project components.
Ensures stability, structural integrity, and strategic placement of components for efficient operation and maneuverability.

Power Source (Samsung Rechargeable Lithium-Ion Batteries):

Samsung rechargeable lithium-ion batteries [3] (2 x 3.7V) selected for compatibility, and capacity.
Meets the project's power requirements, providing sustained power for uninterrupted robot functionality.

4.2 Photo of the Actual System

From the pictures Figure 4.2 to Figure 4.4 is presented the pictures of the line follower itself.

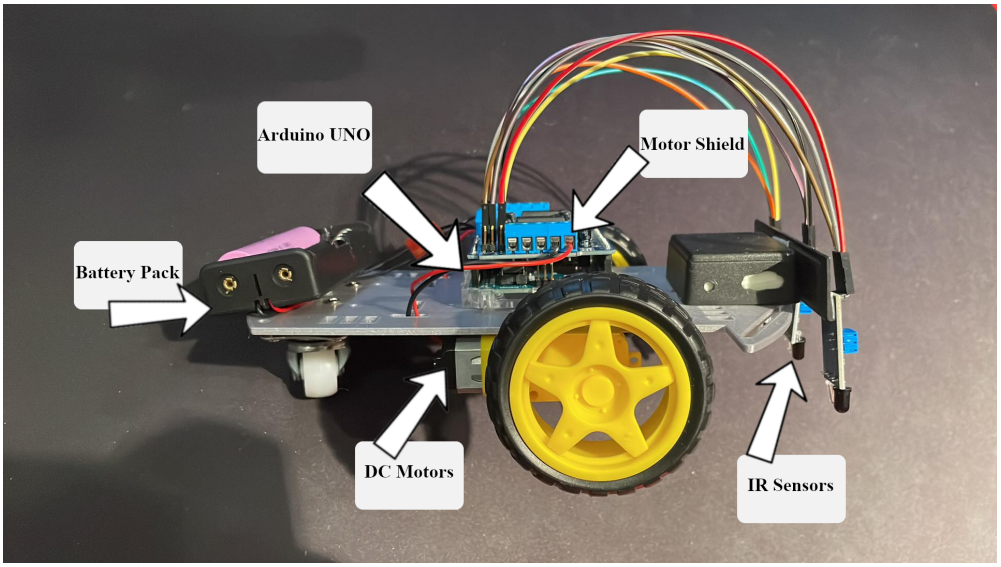

Figure 4.2: System with reference to the block diagram

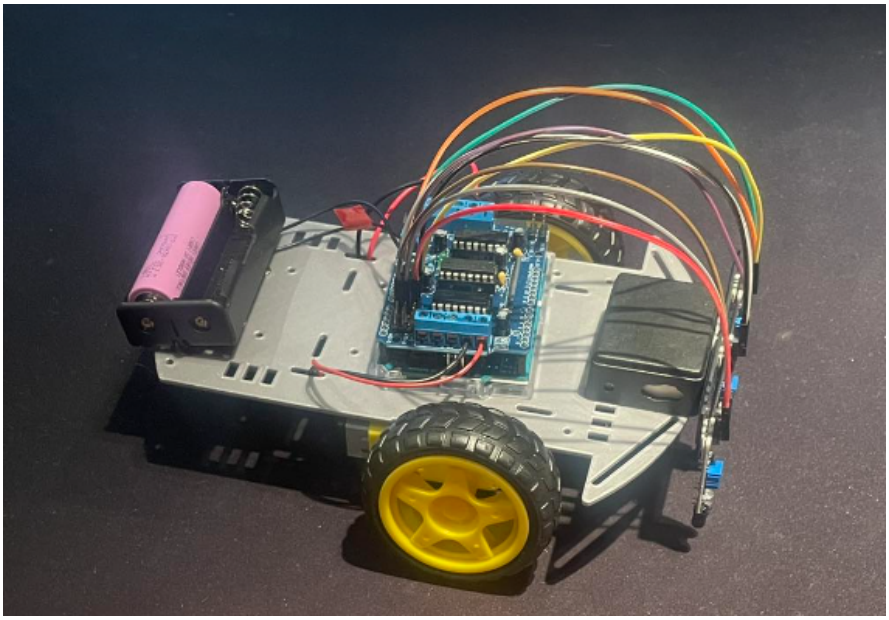

Figure 4.3: Diagonal View

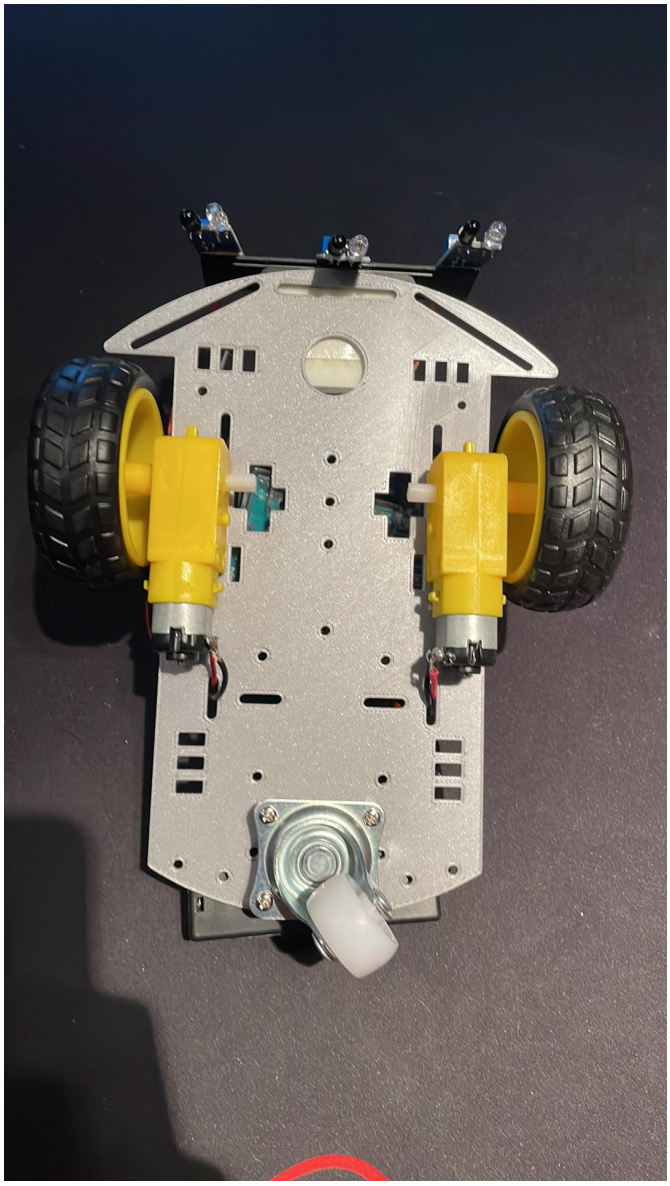

Figure 4.4: Under view

4.3 Mechanical Part

Chassis:
3D printed chassis designed for stability and strategic component placement.
Provides a solid base for optimal performance.
Utilizes 3D printing technology for precision and customization.

Rotating Chassis Wheel:
Enables smooth movement of the back of the robot.[6]

Cables:
Female-to-Female establish connections between components with the same connector type.

Motor Drive Connection Cables:
Cables for connecting components with motor drives. Facilitates efficient power and signal transmission. Selected for durability and compatibility with motor drive systems.

4.4 Electrical Schematic Diagram

Here (Figure 4.5) is the diagram of the components used and how their are connected.

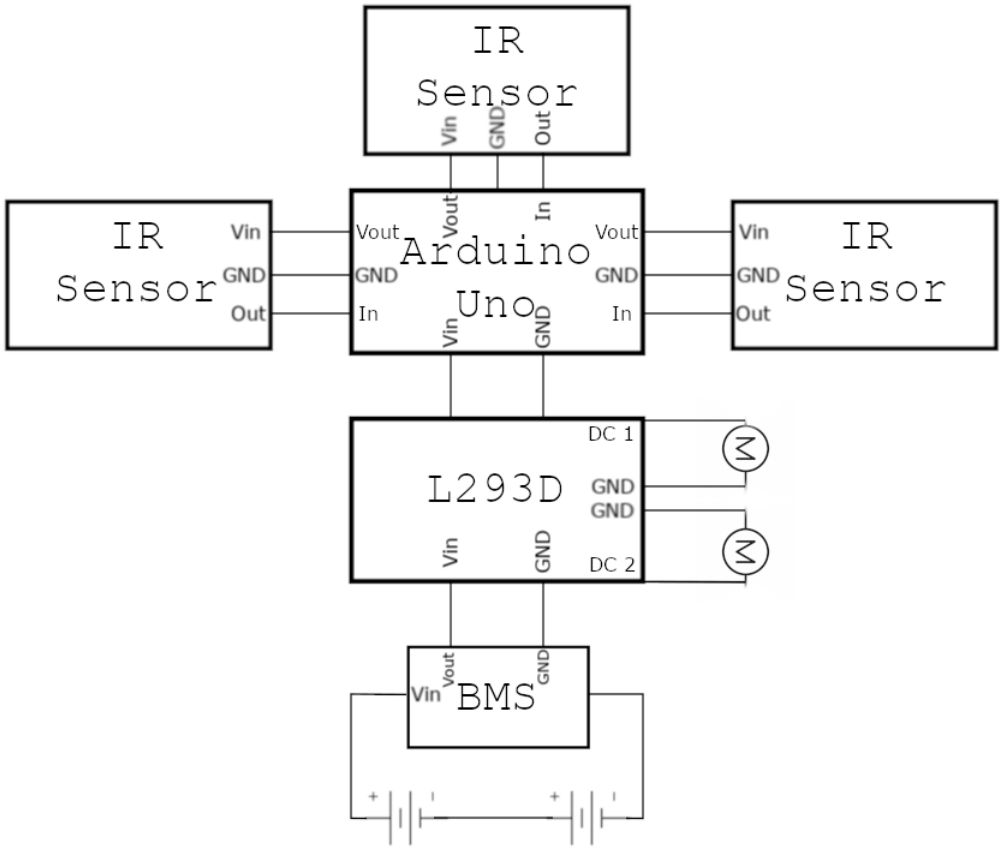

Figure 4.5: Electrical view

Batteries and Battery Management:

- Two Rechargeable Lithium-Ion Batteries (3.7V each) are securely placed and connected to the Battery Management System (BMS).
- The BMS [7], responsible for precise voltage management, is directly connected to the power input of the Motor Driver Shield.

Power Distribution:

- The positive terminal of the batteries is linked to the power input of the BMS, ensuring regulated power distribution.
- The BMS connects to the Uno Power Input on the Motor Driver Shield, maintaining a centralized power control point.

Motor Driver Shield and Wheels:

- The Motor Driver Shield serves as the intermediary between the Arduino Uno and the motorized components.
- The wheels, crucial for robot movement, are directly connected to the Motor Driver Shield ports (M1 and M4).
- The Motor Driver Shield receives signals and power from the Arduino Uno, translating commands into actions for the wheels' operation.

IR Sensors and Arduino Uno via Motor Driver Shield:

- Infrared (IR) Sensors [8], designed for line detection, are connected to specific ports on the Motor Driver Shield.
- The Motor Driver Shield acts as the interface between the IR sensors and the Arduino Uno.
- Sensor data is processed by the Arduino Uno, which, in turn, communicates with the Motor Driver Shield to influence robot movement.

4.5 PCBs:

Main PCB:

Centralizes major components, including the Arduino Uno, Motor Driver Shield, and connectors for other peripherals. Integrates power distribution and signal routing for seamless communication between components. Figure 4.6 shows the PCB of the motor driver and Figure 4.7 present the PCB for Arduino UNO.

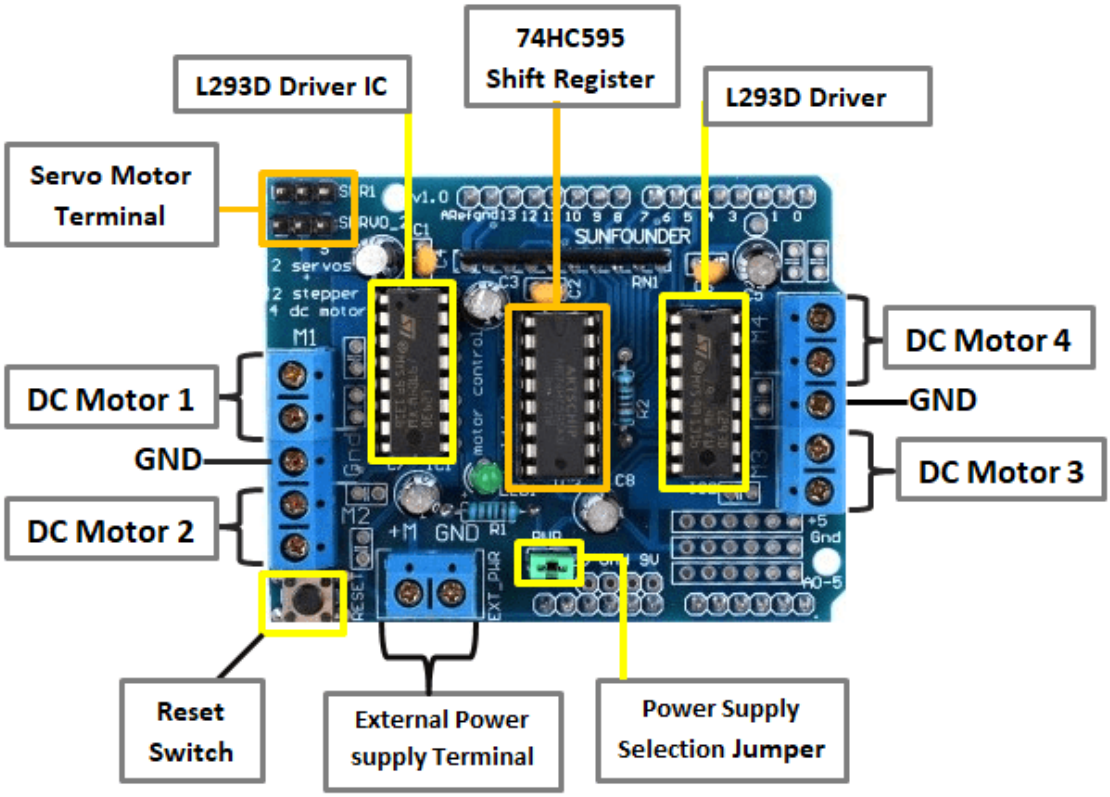

Figure 4.6: Layout of Motor Driver Shield

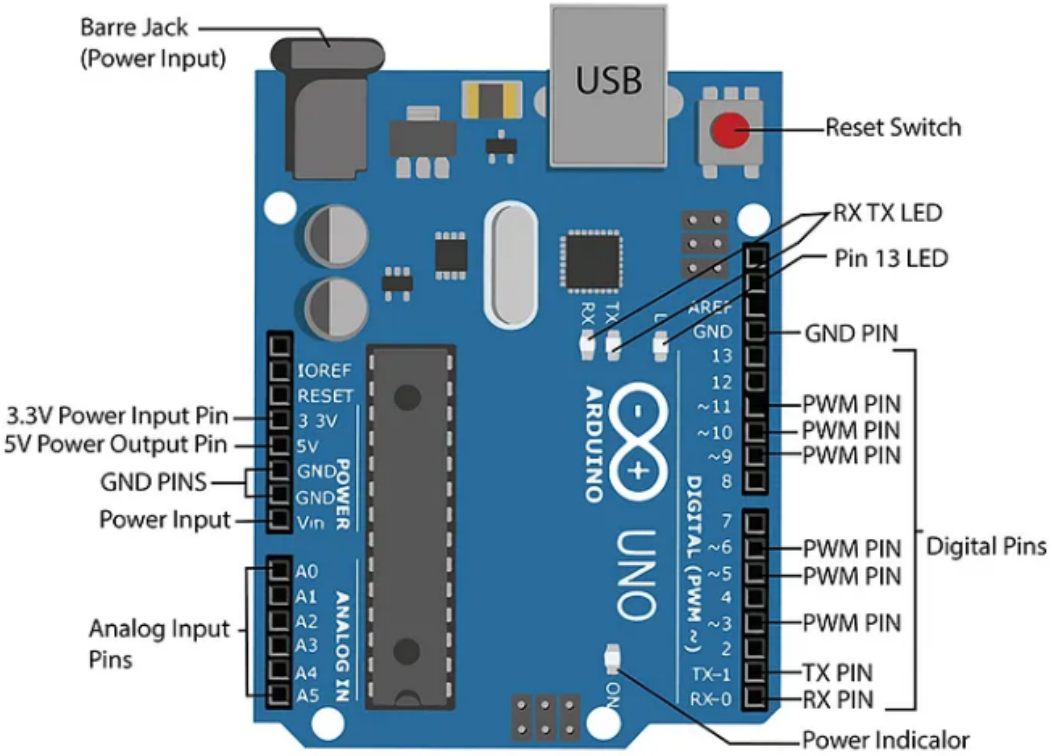

Figure 4.7: Layout of Arduino Uno

Sensor Interface PCB:
PCB for IR Sensors, consolidating sensor connections and ensuring precise data transmission. Connected to the Motor Driver Shield for streamlined communication with the Arduino Uno. Figure 4.8 presents the IR sensor.

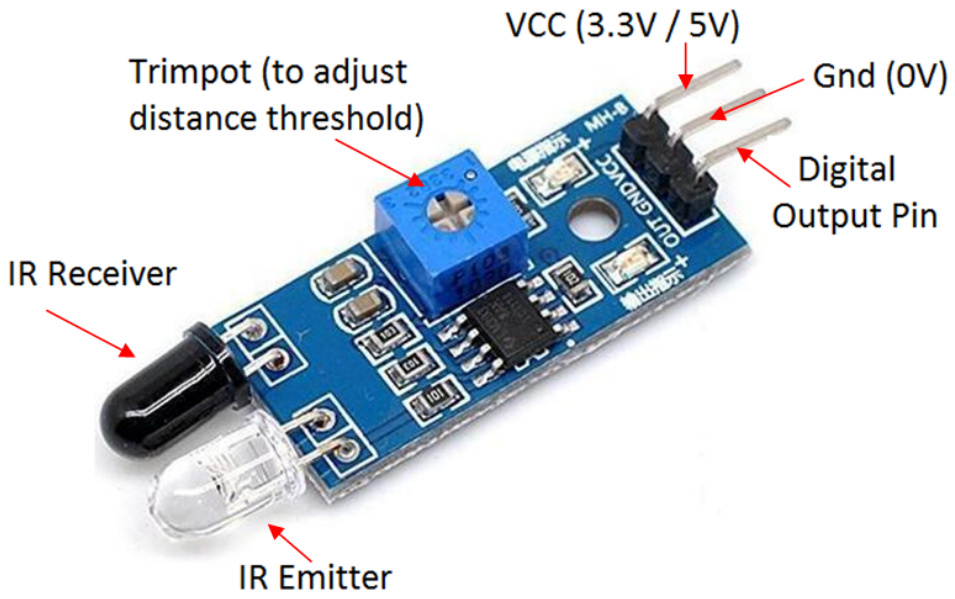

Figure 4.8: Layout of IR sensor

Power Distribution PCB: Manages the power flow from the batteries to the BMS (Figure 4.9) and subsequently to the Motor Driver Shield and Arduino Uno. Enhances system reliability and stability.

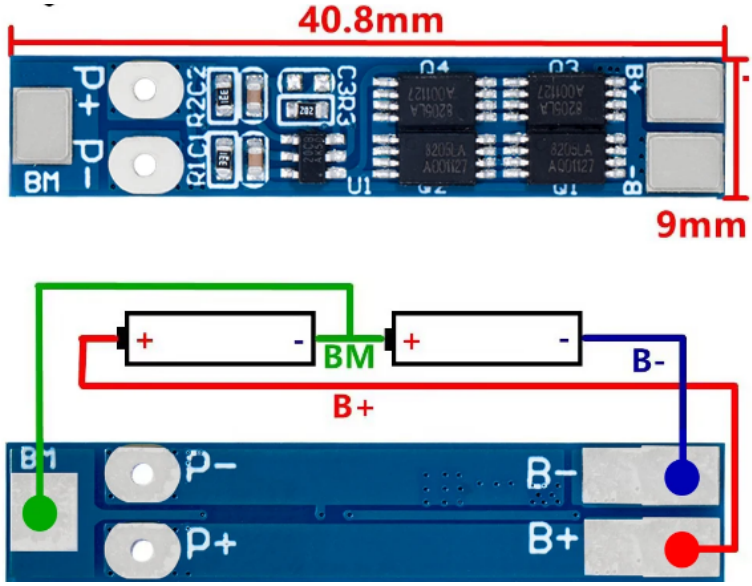

Figure 4.9: Layout BMS

5 Description of the software

The line-following software for the Arduino Uno, equipped with the L293D motor shield and three binary IR sensors (left, middle, and right), is designed to enable precise navigation along a predefined path. The software is written in the Arduino programming language, a variant of C/C++, using the Arduino IDE. Figure 5.1 shows the key points describing the functionality and components of the implemented software.
The full code can be found in the appendix.

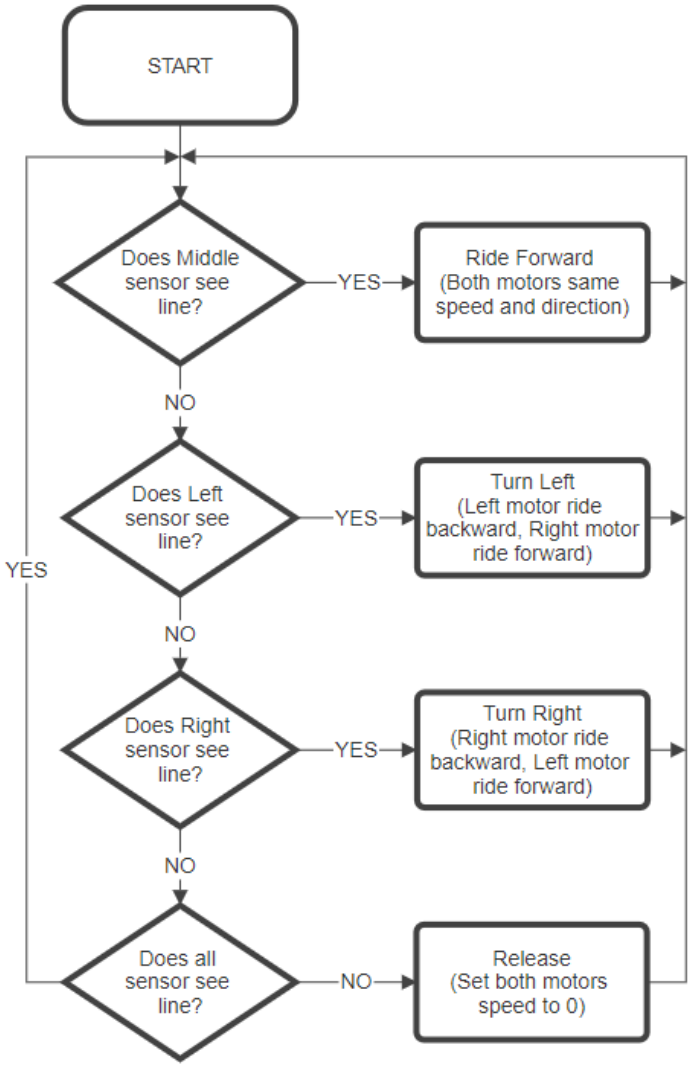

Figure 5.1: Flowchart of code

Sensor Integration:
The software integrates data from three sensors placed strategically on the robot: left sensor, middle sensor, and right sensor. These sensors provide binary readings (0 or 1) based on whether they detect the line. When sensors detect the white line it provides 0 reading.

Motor Control with L293D Motor Shield:
We have tested two strategies — opposite speed turning and zero speed turning. In opposite speed turning, the software adjusts the speed of one wheel opposite to the other for turning maneuvers. In zero speed turning, one or both wheels are brought to a complete stop to facilitate turns.

Forward Movement:
The software ensures straightforward movement when middle sensor read 0 or all the sensors read 0, indicating that the robot is on the correct path. Both wheels move forward at a specified speed in this scenario. See the full code lines 33 - 40 (Appendix).

Adjustment for Straight Paths:
To maintain a straight path, the software incorporates logic to balance the wheel speeds when the center sensor detects the line while the left and right sensors remain off the line.

Calibration and Error Handling:
The software allows calibration to adjust for changes in sensor characteristics, wheelbase and working surfaces. On the other hand, due to the specification of our components, among other things, binary sensors (0 or 1) which provide less accurate values. See the full code lines 57 - 65 (Appendix).

Arduino IDE Implementation:
The entire codebase is written using the Arduino IDE, providing an easy-to-use platform for programming the Arduino Uno. This facilitates code upload, debugging, and iterative development.

Opposite speed turning:
We decided to work with opposite speed turning because of its benefits in achieving a better turning radius and a smoother ride. Opposite speed turning is a technique employed in our line-following robot, where, upon detecting a deviation from the desired path, the robot adjusts its direction by reversing the wheel on the side where the deviation is detected, while the wheel on the opposite side continues moving forward. This differential speed between the two wheels allows the robot to smoothly navigate turns with a reduced turning radius, enhancing its agility and overall performance in following the designated path. See the full code lines 41 - 56 (Appendix).

Code: Code for robot with comments is in appendix.

Summary: The line-following software for the Arduino Uno, L293D motor shield, and three binary sensors is a comprehensive implementation that leverages sensor data and motor control strategies to achieve accurate and responsive navigation along a predefined path.The modular and adaptable nature of software allows for further refinement and customization to suit specific robot configurations and environmental conditions.

6 Start-up, calibration

This chapter includes the information we gathered to calibrate our components as well as how we got out first start-up.

6.1 Wheel Movement:

Initially, following the advice from the professor, the decision was made to test the robot's functionality with a single wheel movement. This initial test proved successful, indicating a functional setup.

Subsequently (figure 6.1), activating both wheels enabled the robot to move forward, confirming the basic functionality of the system.

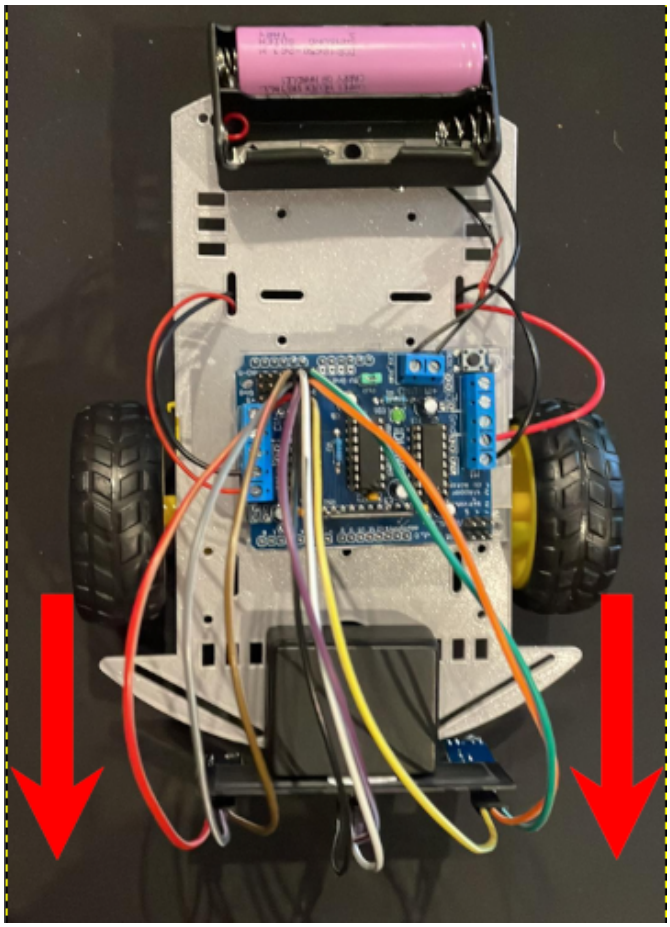

Figure 6.1: Demonstration of activating both wheels

6.2 Sensor Calibration Challenges:

Calibration Attempt 1: The initial attempt to calibrate the sensors involved setting the left and right sensors for white and the middle sensor for black. However, this configuration did not yield the expected results.

Code Modification: Recognizing the difference, modifications were made to the code to ensure proper interpretation of the sensor data. Adjustments were implemented to achieve the desired functionality.

Optimizing Sensor Calibration: Achieving accurate and effective sensor calibration proved to be the most challenging aspect of the process. Fine-tuning the sensitivity and thresholds of the sensors required minor adjustments to accurately detect line deviations.

6.2.1 Iterative Calibration Process:

Iterative Approach: To achieve correct sensor calibration, an iterative approach was adopted. This involved multiple adjustments in the code and sensor settings to find the optimal configuration for precise line detection.

Trial and Error: Several iterations of trial and error were necessary to achieve the desired accuracy in line detection. Small incremental changes were made and tested repeatedly to refine the sensor calibration.

Successful Calibration Outcome: Accomplished Calibration: After several iterations and adjustments, the correct calibration settings were determined, resulting in the robot accurately following the predefined path.(Figure 6.2)

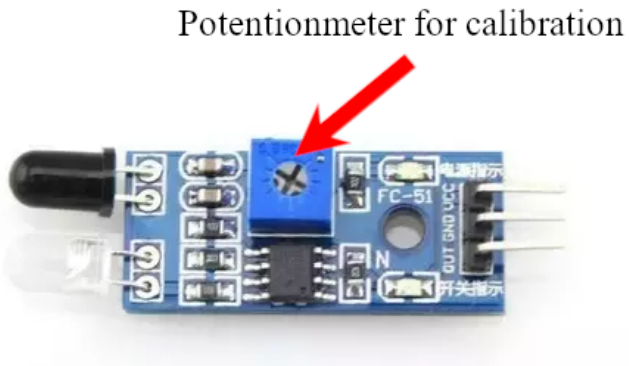

Figure 6.2: Demonstration of how to calibration the IR sensor

6.2.2 Lessons Learned:

The calibration process highlighted the importance of understanding sensor behavior and the significance of code adjustments to achieve desired functionality. Also it demonstrated the significance of persistence and an iterative approach in troubleshooting and fine-tuning complex systems like the line follower robot.

7 Test measurements

This section dives into precise measurements and specifications for the core components integral to our line follower robot project.

Battery Assessment:

Typical Capacity: 3000 mAh
Minimum Capacity: 2900 mAh
Nominal Voltage: 3.7 V
Max Charging Current:
Standard: 1.5 A
Maximum: 4 A
Max Discharge Current:
Dimensions:
Diameter: 18.3 mm
Height: 65 mm
Weight: 46 g

Motor Shield:

Supply Voltage: 4.5 V to 16 V
Max Current per Channel: 600 mA

IR Sensor

Measured Distance: to 5 cm
Operating Voltage: 3.3 to 5 V
Response Time: 0.5 s
Board Dimensions: 32 x 15 mm

Wheel with Motors:

Motor Parameters:
Supply Voltage: 5 V
Current Consumption: 180 mA
Wheel Parameters:
Tire Diameter: 65 mm
Tire Width: 26 mm

Chassis:

Color: Grey
Dimensions: 20 x 10 cm

Track:

Tape Color: white
Tape Width: 2,5 cm
Background: black paper
Background width: A1 paper
See figure 7.1.

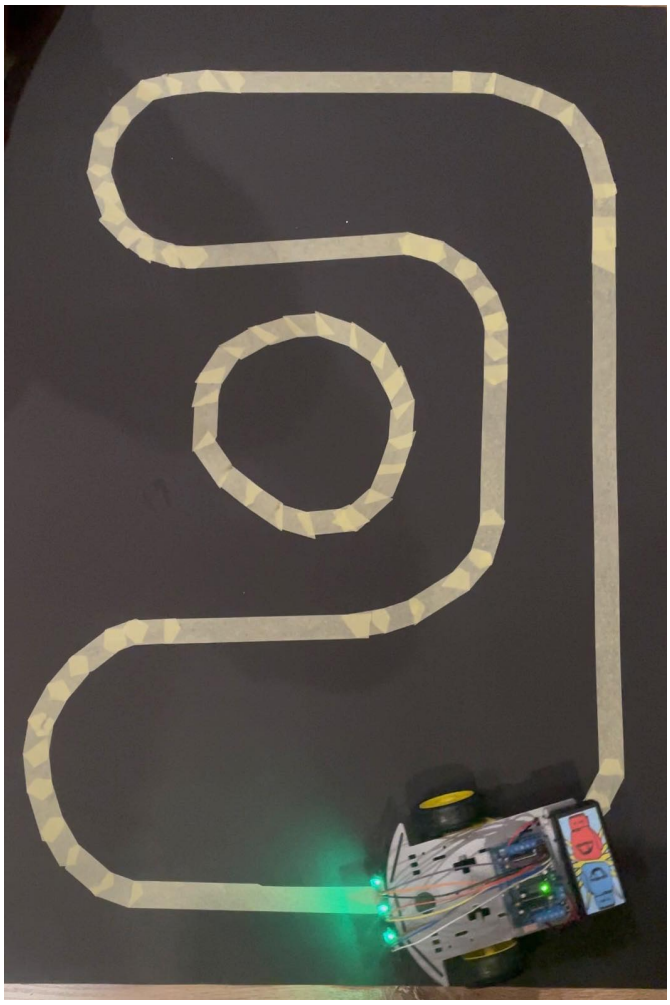

Figure 7.1: Image of track

These precise measurements and specifications serve as a comprehensive guide, providing in-depth insights into the characteristics of each component. Understanding these details is crucial for optimal integration and ensures the seamless functioning of our line follower robot.

8 User manual

How to use the line follower is very straightforward also this manual will not get into very specific details. Firstly we need to check that the line follower is charged (We are able to turn it on): After putting the two Lithium-Ion Batteries we may be able to see a led lighting up in the motor driver shield (Figure 8.1)

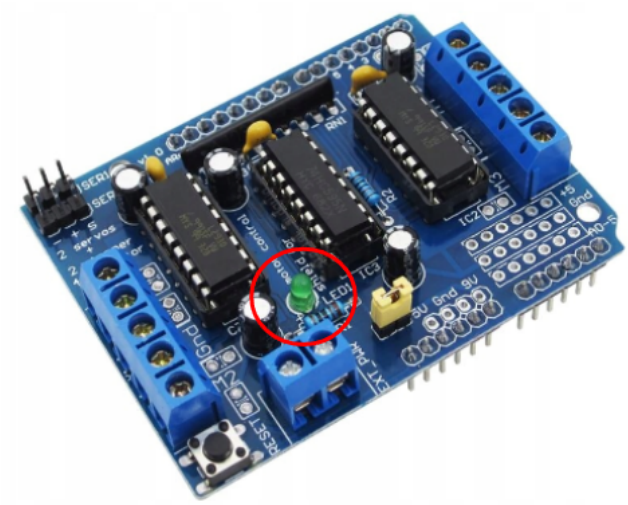

Figure 8.1: LED light

To begin using it we need to follow two main steps: Path set-up and Sensor calibration, optionally you can customize the source code. While Path set-up may be the only necessary step to begin using the line follower, the other steps can be used for customization.

Path set-up:

We need to have a good contrast between path and background as show in the figure 8.2. Depending on the quality of the material or room some calibration may be needed:

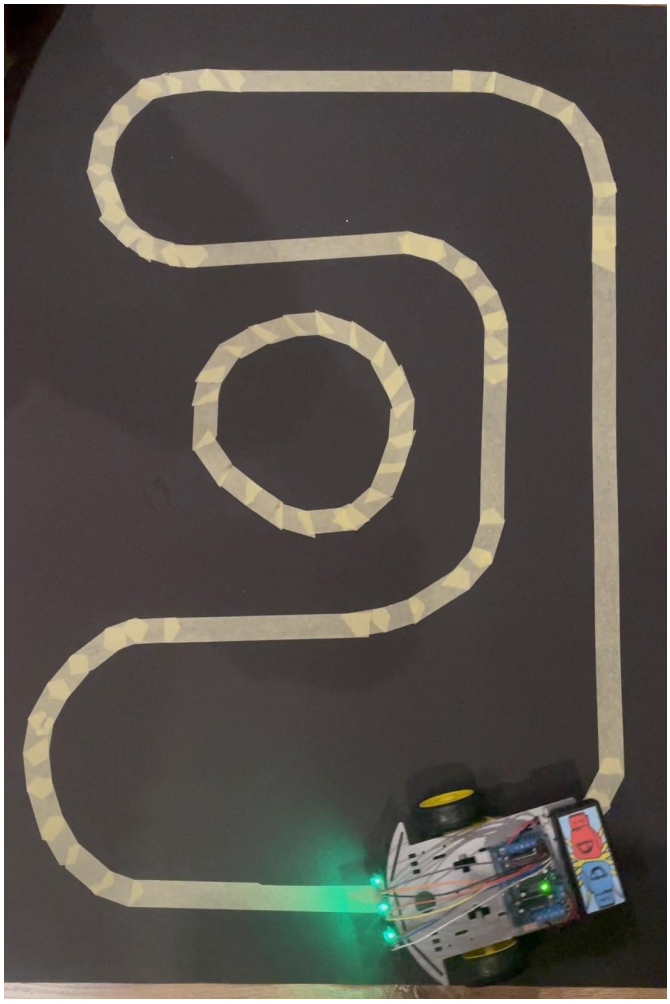

Figure 8.2: Image of track

Although the algorithm used in the line follower allow us to have intersecting path lines it will not necessarily follow a desired path, given the nature of the logic used in the algorithm, rather it will tend to continue straight without turnings: it does not have any kind algorithm to choose an optimal path.

Sensor Calibration:

The LED's should be turned on it can be green or red but the fact that the LED is on indicates that they are working properly. You can use a screw driver to turn the potentiometer (Figure 8.3) on the sensors.

Specifications:
We will not go into the details of the source code, however you can try to change it if you know how to calibrate everything else (touching some parameters will may need some sensors calibrations).
Be carefully at the moment of handling the batteries the provided ones are rechargeable so you may need to charge them before using it. In case you need to replace the batteries their voltages are 3.7V. The motor driver shield used is L293D and we recommend

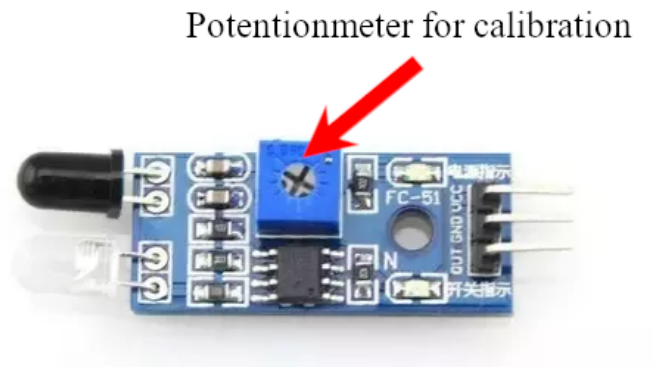

Figure 8.3: How to calibrate the sensor

to not trying to changing it unless you are able to connect everything back.

There is an on/off button so the way you turn off the line follower is by picking it up and removing the batteries.

9 Summary

We made a line follower robot project capable of navigating a predefined path with precision, leveraging infrared (IR) sensors. The heart of our creation comprised essential components like the Arduino Uno micro-controller, infrared sensors, DC motors with L293D motor driver shields, and a custom 3D-printed chassis.

The project's theoretical foundation was rooted in the workings of IR sensors, particularly the combination of IR LEDs and photodiodes forming the basis for our path-detection mechanism. The emphasis was on interpreting sensor values to adjust the robot's behavior, ensuring seamless navigation across diverse track conditions.

However, the project encountered its fair share of challenges during the sensor calibration process. Setting accurate threshold values to distinguish between dark and light surfaces proved to be a complex task, necessitating an iterative calibration approach. Despite initial setbacks, the team's persistence and iterative methodology triumphed, leading to successful sensor calibration, allowing the robot to adeptly follow the predefined path.

In addition to calibration hurdles, the project faced unexpected issues with the battery basket, resulting in three replacements. The Li-ion batteries and the integrated Battery Management System (BMS) played a critical role in powering the robot, accentuating the paramount importance of ensuring safety and reliability in the battery system.

The team collaborated on various aspects, with a focus on sensor calibration, software development, and integration testing. Deviations from the initial task distribution were minimal, with flexibility in addressing challenges as a team. Please refer to the appendix containing the project plan for a full view of milestones and tasks set.

Despite facing challenges during sensor calibration and battery cage durability, the project underscored the team's resilience and problem-solving prowess. The resulting line-following robot exhibited not only precise navigation but also opened the door for potential upgrades. Exploring the option of enhancing the robot with advanced motors and sensors is intriguing, albeit constrained by budget and time limitations. This project has highlighted the excitement and enjoyment inherent in scientific endeavors. Looking ahead, the experience gained will fuel the pursuit of more ambitious and captivating projects in the future.

10 Bibliography

1. Line Follower Robot using Arduino
2. DC Motor Wheel 65x26mm 5V with Gearbox
3. 18650 Li-ion Battery Cell - Samsung INR18650-30Q 3000mAh
4. L293D Motor Driver Board for Arduino
5. Battery Holder for 2x 18650 Cells - Series Connection
6. Rotating Wheel for Robot Chassis
7. Battery Management System (BMS) for Li-ion Cells - 2S 8.4V 5A
8. Infrared Obstacle Detection Sensor - LM393
9. Arduino Uno Rev3
10. Rafal Cyminski - Master's Thesis (PDF)
11. HBFS Line follower

11 Appendix containing code:

Code for our robot with comments.

```
1  // Purpose: Algorithm that detect the data from \\
2  // the sensor and follows a predefined line.\\
3  // Credits: Aleksander Pauch\\
4  // Date: 23/01/2024\\
5  // Model: Arduino UNO
6  // Including the required library for motor control
7  #include <AFMotor.h>
8
9  // Defining pins and variables for sensor inputs
10 #define left A0
11 #define middle A1
12 #define right A2
13
14 // Defining motors with specific pins and frequency
15 AF_DCMotor motor1(2, MOTOR12_1KHZ); // left motor
16 AF_DCMotor motor4(4, MOTOR34_1KHZ); // right motor
17
18 // Setup function runs once at the beginning
19 void setup() {
20   // Configuring sensor pins as INPUT
21   pinMode(left, INPUT);
22   pinMode(right, INPUT);
23   pinMode(middle, INPUT);
24
25   // Initializing serial communication for debugging
26   Serial.begin(9600);
27 }
28
29 // Loop function runs repeatedly after setup
30 void loop() {
31   // Print sensor values occasionally for debugging
32   if (millis() % 100 == 0) {
33     Serial.println(digitalRead(left));
34     Serial.println(digitalRead(middle));
35     Serial.println(digitalRead(right));
36   }
37
38   // Line detected by middle sensor
39   if (digitalRead(middle) == 0) {
40     // Move forward
41     motor1.run(FORWARD);
42     motor1.setSpeed(100);
43     motor4.run(FORWARD);
```

```
44      motor4.setSpeed(100);
45    }
46    // Line detected by left sensor
47    else if (digitalRead(left) == 0) {
48      // Turn left
49      motor1.run(BACKWARD);
50      motor1.setSpeed(100);
51      motor4.run(FORWARD);
52      motor4.setSpeed(100);
53    }
54    // Line detected by right sensor
55    else if (digitalRead(right) == 0) {
56      // Turn right
57      motor1.run(FORWARD);
58      motor1.setSpeed(100);
59      motor4.run(BACKWARD);
60      motor4.setSpeed(100);
61    }
62    // No line detected by any sensor
63    else if (!digitalRead(middle) && !digitalRead(left)
64    && !digitalRead(right)) {
65      // Stop motors
66      motor1.run(RELEASE);
67      motor1.setSpeed(0);
68      motor4.run(RELEASE);
69      motor4.setSpeed(0);
70    }
71  }
```