Summer Projects
Science and Technology Council
IIT Kanpur

Fastest Line Follower

Documentation

Date of Submission:

July 9, 2019

Contents

1 Acknowledgements **4**

2 Introduction **5**
2.1 Aim 5
2.1.1 Problem Statement 5

3 Brief Approach **5**

4 Components **5**
4.1 Mechanical Parts 5
4.1.1 Chassis 6
4.1.2 Wheels 6
4.1.3 Caster 7
4.1.4 Mounting Bracket 8
4.2 Electronic Components 8
4.2.1 Batteries 9
4.2.2 Controller 9
4.2.3 Sensor 10
4.2.4 Motors 11
4.2.5 Motor Driver 12
4.2.6 Data Relay (HC05 Bluetooth Module) 13

5 Design **14**
5.1 Bot 14
5.1.1 Ideation 14
5.1.2 Implementation 15
5.2 Arena 16
5.2.1 Ideation 16
5.2.2 Implementation 16
5.3 Algorithms 17
5.3.1 Sensor Input 17
5.3.2 Line Position 17
5.3.3 PID Controller 18
5.3.4 Left-Hand Algorithm 19

6 Assembly **20**
6.1 Bot 20
6.2 Electronics 21
6.2.1 Motor Driver 21
6.2.2 LFS Sensors 22
6.2.3 Encoder Motor 22
6.2.4 Battery 22
6.2.5 HC05 23

7 Implementation **25**

7.1 Testing Pipeline 25

7.2 Speed Control 25

7.3 Line Following 25

7.4 Left Hand Algorithm 26

7.5 Data Relay 26

Bibliography **27**

1 Acknowledgements

We are immensely grateful to Robotics Club, IIT Kanpur and Science & Technology Council, IIT Kanpur for providing us with the resources and motivation for this Summer Project. We thank our mentors, the Coordinators of Robotics Club: Neil Shirude, Abhay Varshney and Madhur Deep Jain for their guidance, constant support and invaluable inputs, without which we would not have been able to do this project in a timely manner.

2 Introduction

2.1 Aim

The goal of this project is to build a highly optimized line following robot, aimed to win the Technoxian World Robotics Championship 2019 - Fastest Line Follower Challenge.

2.1.1 Problem Statement

The official website cites the following Problem Statement:
"Build your own autonomous robot within the specified dimensions to achieve the maximum speed to beat other robots on the given track to reach the destination in minimum time. The robot must start behind the starting point and is considered to have crossed the finishing line if any part of the robot crosses it in a full lap of the course. The robot must follow the black line."
In addition, there will be two rounds:

Knockout Round: In this round two teams will compete one-on-one starting at the same time, the team finishing the race in the minimum time will qualify for the Final round.

Final Round: Qualified teams will compete in this round to achieve the minimum runtime. Single run will be conducted for each qualified team. Teams with the minimum runtime will be nominated as the winner of the competition.

Further details about the competition are as follows:

- Pre-Game setup: Before starting the competition, each team will get 10 mins to calibrate the robot and make a trial/testing runs. All the preparations must be done during this time (adjusting the sensors, reprogramming the robot etc).
- Scoring: The only criteria to score is the final runtime achieved during each round of the competition. Teams with the minimum Runtime will get maximum points.
- Robot Design: The robot participating in the competition must not be ready made. It must be autonomous, non-destructive, non-harmful and electrically powered. The robot must also fit inside a 20cm X 20cm X 20cm box.

3 Brief Approach

The bot was first designed with basic components - motors, wheels, batteries, casters etc. Then the LFS sensors were attached to it, along with a motor driver and the Arduino MEGA. Then an arena was designed and printed on a flex sheet. After this, we started working on the implementation of the line following algorithms in a module-based manner. The different code modules included - Motor control, Line Following using a PID controller, Memory and Graph-based algorithm for arena mapping and Data Relay for easy debugging.

4 Components

This section gives a detailed description of all the components used in this project.

4.1 Mechanical Parts

This section describes all the mechanical components used.

4.1.1 Chassis

An acrylic sheet of thickness 5mm was used to make the chassis of the robot. The width and length of the sheet were decided according to the constraints specified in the problem statement.

Figure 1: Acrylic Sheet

Features:

- Can be folded easily by applying heat
- Lightweight and rigid
- Inexpensive

4.1.2 Wheels

For the wheels of the robot we used the 3PI miniQ Car wheel Tyre N20 DC Gear Motor Wheel(1). These are customized high-quality rubber wheels of diameter 42mm.

Figure 2: Wheels

Features:

- Coder Accuracy: 12 pulses per revolution
- Heavy Duty Material
- Durable and specially designed

Loading Capacity (Kg)	3
Weight (gm)	43
Wheel Diameter(mm)	42
Color	Black (Tyre) White (Rim)
Wheel Width	18
Body Material	Plastic
Grip Material	Rubber
Center Shaft Hole Diameter	3 mm D-type

Table 1: Wheel Specifications

4.1.3 Caster

Heavy duty metal steel ball encased in plastic casing. The caster(2) allows smooth omni-directional movements.

Features:

- Three equally spaced mounting holes to fix it tightly with chassis
- Easy to apply lubricant
- Lightweight
- Inexpensive

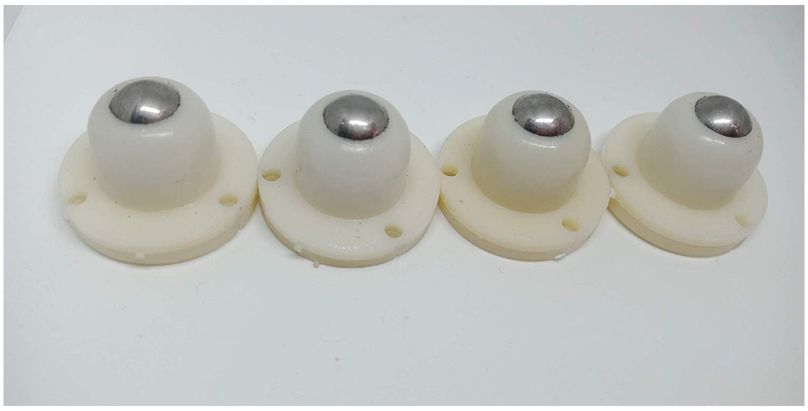


Figure 3: Caster Wheel

Wheel Total Height	23 mm
Mounting Hub Diameter	33.5 mm
Mounting Hole Diameter	4 mm
Mounting Hole PCD	27.5 mm

Table 2: Caster Specifications

4.1.4 Mounting Bracket

The bracket(3) is used to clamp the motors onto the chassis.

Features:

- Durable
- Lightweight
- Perfectly fits to micro motors
- Inexpensive

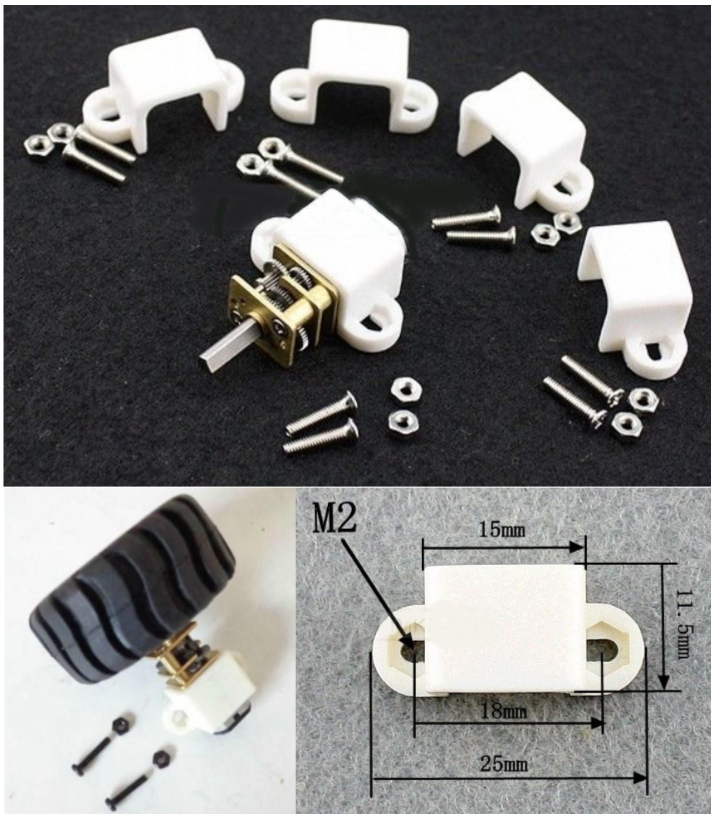


Figure 4: Mounting Bracket

Height	11 mm
Width	25 mm
Material	High Quality Plastic

Table 3: Bracket Specifications

4.2 Electronic Components

This section describes all the electronic components used.

4.2.1 Batteries

For powering the robot, we have used Lithium Polymer batteries. These batteries, also known as Li-Po batteries(4) are widely used in GPS, DVD, i-pod, Tablet PC, MP4 Player, Power Bank, Mobile Backup Power Supply, Bluetooth Speaker, and Industrial applications.

`Features:`

- Lightweight
- Rechargeable
- Small size
- Easy and safe to use

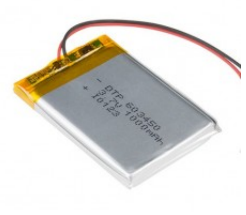


Figure 5: Batteries

Voltage	3.7V
Capacity	1000 mAh
Dimensions	50mm X 30mm X 4mm

Table 4: Battery Specifications

4.2.2 Controller

Arduino Mega 2560 follows Open-source principles, which means you can freely download the development environment and many associated resources. Ideal for project work and often used in schools, teaching, motor control, robotics and similar applications; the Arduino platform allows for rapid application development using either Java (cross-platform) or Processing (a C derivative used to write sketches using Arduino IDE downloadable from Arduino.cc website.)

Input voltage should be limited to 6 12V, but if the voltage supplied is less than 6V, I / O port may not be supplied to a voltage of 5V, and therefore readings may become unstable. If the voltage is greater than 12V, the regulator device may overheat causing damage to the control board. It is therefore recommended that you supply between 6.5 and 12V, utilizing typical power supplies of 7.5V or 9V.

Mega 2560 is fully compatible with the Arduino software and has complete hardware compatibility with the Mega 2560 Arduino Shield high-performance processor and enough memory to handle even complex solutions.

Features:

- ATmega2560 Microcontroller. Boot-loader to allow downloading of ‘sketches’ via USB without having to go through other external writers.
- Powered from USB or via external PSU (not supplied). The device will automatically switch between power inputs.
- Heavy gold plate construction.

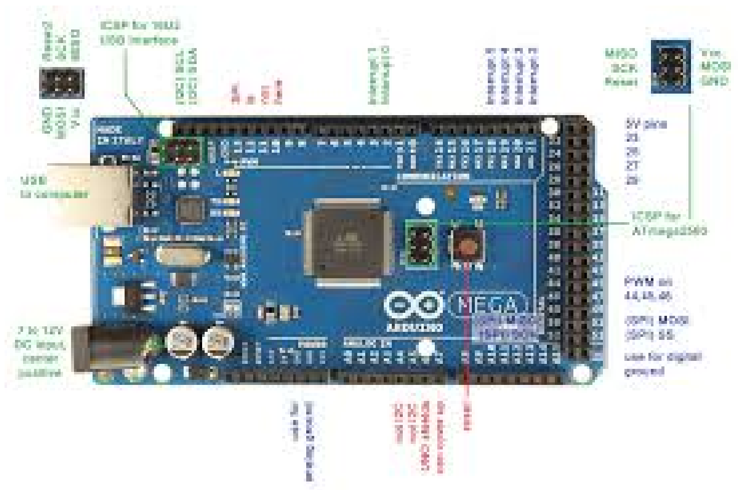


Figure 6: Arduino mega labelled diagram

Microcontroller	ATmega2560
EEPROM	4KB
Flash Memory	256 KB of which 8 KB used by bootloader
SRAM	8KB
PWM Output Pins	15
Analog I/O Pins	16
Digital I/O Pins	54 digital input / output terminals (14 of which have programmable PWM outputs)
DC Current per I/O Pin	DC Current per I/O Pin: 40 mA ; DC Current for 3.3V Pin: 50 mA
Architecture	Advanced RISC Architecture
Clock Speed	16 MHz
Input Voltage(Recommended)	7-12V
Operating Voltage	5
Weight (gm)	35g
Color	Blue
Dimensions (mm) LxWxH	110 x 53 x 15

Table 5: Arduino MEGA specifications

4.2.3 Sensor

The Line Follower Sensor (LFS)(5) gives the robot the ability to sense the lines. It consists of reflectance sensors to differentiate between white and black or dark and light lines by detecting the reflected light coming from its own infrared LED. A single board has an array of six sensors precisely designed for smooth line following. It provides analog outputs and can be interfaced with any microcontroller.

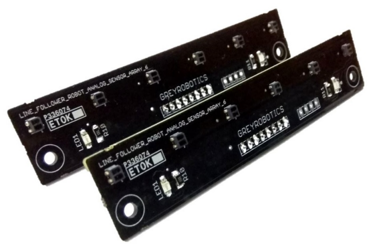


Figure 7: LFS Sensor

Features:

- Uses Reflectance Sensors for high precision
- Easy to use analog output pins
- TTL (Transistor Transistor Logic) compatible outputs
- Two power LED indicators
- Six sensors in an array
- Black and white line following
- 3mm diameter hole for easy mounting
- Compatible with Arduino, AVR, PIC, ARM, etc

Operating Voltage	5V
Current Requirement	210 mA
Output Voltage Range	0-5V
Recommended Sensing Distance	5 mm
Maximum Sensing Distance	12 mm
Operating Wavelength	920 mm
Distance between two reflectance sensors	18 mm
Dimensions	100 mm X 20 mm
Weight	less than 10 grams

Table 6: Sensor Specifications

4.2.4 Motors

The Micro Metal Geared encoder motor(6) comes with a built-in encoder which measures the motor's speed in real time. The average output number of pulses can reach up to 420 pulses per revolution. The motor has a long (0.354"/9.0 mm) D profile metal output shaft. The brass faceplate has two mounting holes threaded for M1.6 screws (1.6mm diameter, 0.35mm thread pitch). The hall sensor has 7 pole pairs, so the encoder resolution will be increased by 7-fold.

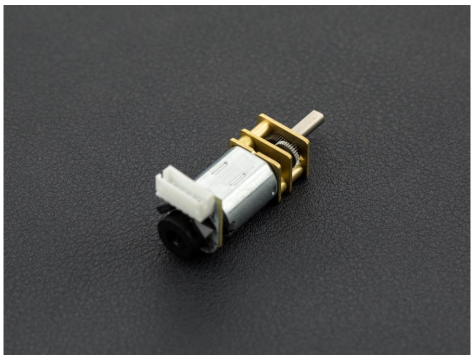


Figure 8: Motor with Encoder

Rated Voltage	6V
Motor Speed	15000 rpm
Gear Reduction Ratio	30:1
Reducer Length	9.0 mm
No-Load Speed	530 rpm at 6V
No-Load Current	60 mA
Rated Torque	0.2 kgcm
Rated Speed	300 rpm at 6V
Current Rating	170 mA
Instant Torque	0.45 kgcm
Hall Feedback Resolution	420
Weight	18g

Table 7: Motor Specifications

4.2.5 Motor Driver

The L298N Motor Driver is a high power motor driver perfect for driving DC Motors and stepper motors. It uses the popular L298 motor driver IC and has an onboard 5V regulator which it can supply to an external circuit. It can control up to 2 DC motors with directional and speed control. This motor driver is perfect for controlling motors using microcontrollers, switches, relays, etc.

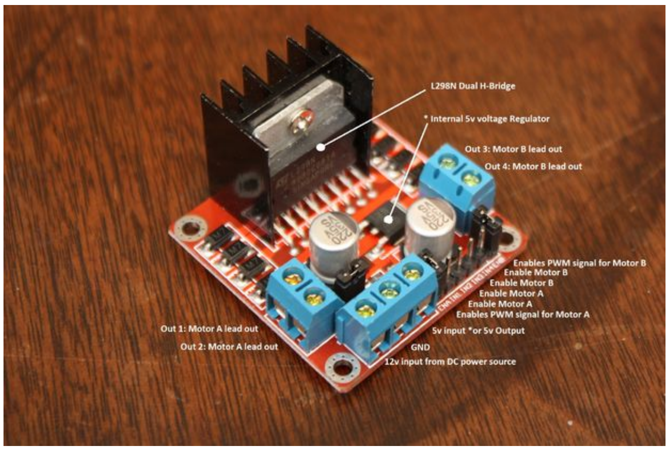


Figure 9: L298N Motor Driver

Features:

- Maximum motor supply current: 2A per motor.
- Current Sense for each motor.
- Heat sink for better performance.
- Power-On LED indicator.
- Double H bridge Drive Chip: L298N.

4.2.6 Data Relay (HC05 Bluetooth Module)

The Bluetooth Transceiver HC-05 TTL Module(7) (With EN Pin) is the latest Bluetooth wireless serial cable. This version of the popular Bluetooth uses the HC-05/HC-06 module. These modems work as a serial (RX/TX) pipe. Any serial stream from 9600 to 115200bps can be passed seamlessly from your computer to target.
The remote unit can be powered from 3.3V up to 6V for easy battery attachment. All signal pins on the remote unit are 3V-6V tolerant. No level shifting is required. Do not attach this device directly to a serial port. It needs an RS232 to TTL converter circuit or Arduino XBee USB Adapter to attach with a computer.

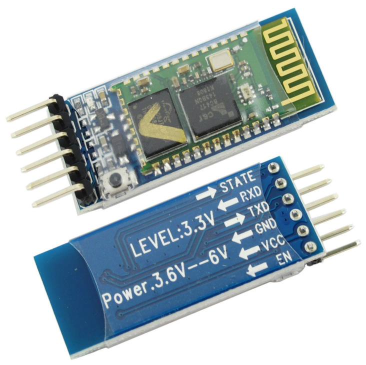


Figure 10: HC05

`Features:`

- Bluetooth Spec v2.0 EDR Compliant
- Enhanced Data Rate (EDR) compliant with V2.0.E.2 of the specification for both 2Mbps and 3Mbps modulation modes
- Incredible small size with 3.3V input, and RoHS Compliant
- UART interface and with baud rate setup function
- Support for 8Mbit External Flash Onboard
- A range of operation: 8 – 10 meter.

5 Design

This section covers all the design aspects of the project.

5.1 Bot

Here we describe the process followed in designing the bot.

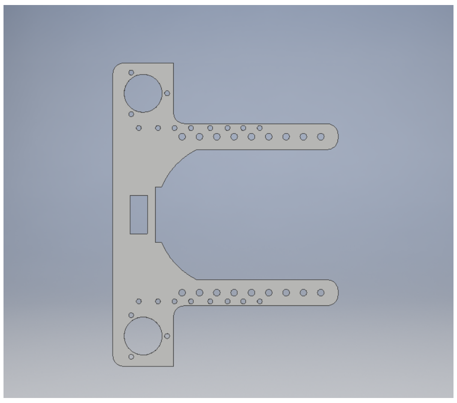
5.1.1 Ideation

The starting phase of the project involved reading various research papers(8)(9) and looking at already built line follower robots to get design ideas. After watching numerous Youtube videos(10) on line follower bots and comparing their performances, we concluded that the design of the bot should be as simple as possible. We also concluded that the bot should have at least the following features:

- **Ground Clearance:** We observed that the robots which had very low ground clearance always seemed to perform better. This was due to the fact that a low ground clearance implied a low of centre of gravity, which increased the stability of the bot and allowed it to accurately take sharp turns at high speeds. However the clearance would also have to be greater than a minimum threshold so that the sensor could sense the line properly.
- **Cylindrical Wheels:** We observed that robots that had small wheels that were not wide enough tended to skid a lot, which would take it off track at a turn and then the controller would be unable to find the line again. Hence we decided to use cylindrical roller-like wheels so that there would be less slipping with greater area of contact.
- **Two Sensors:** Most robots were implemented using only one simple array of IR sensors. We observed that the robots in such a case were unable to handle discontinuities and dead-ends properly, as the bots would overshoot at such points in the path. Hence we decided to use 2 LFS sensors, one at the front of the bot which would be used to control the motion of the robot, and a reference sensor at the back which would be used in case the first sensor fails to detect the line at any point in the path.
- **Adjustable Length:** Since our goal is to optimize the line following robot as much as possible, the distance between the two sensors must be optimum to achieve best results. So we decided to make the length of the robot adjustable by splitting its chassis into two parts, one with the front sensor and one with the back sensor.

5.1.2 Implementation

The two parts of the chassis were first designed using AutoCAD Inventor 2018. Slots were made in the frame according to the measured sizes of each component. Additionally, some holes were made in order to have adjustable placement of both the sensors. The front part was made curved in order to decrease the total weight of the robot. The back part of the chassis is rectangular since it would carry most of the components. The robot was then first assembled in AutoCAD after importing CAD diagrams of the major components from the internet. Once all the discrepancies had been removed from the assembly, the top faces of the chassis parts were exported into .dxf files. These files were then used to laser cut the acrylic sheet to specifications.


(a) Front Part

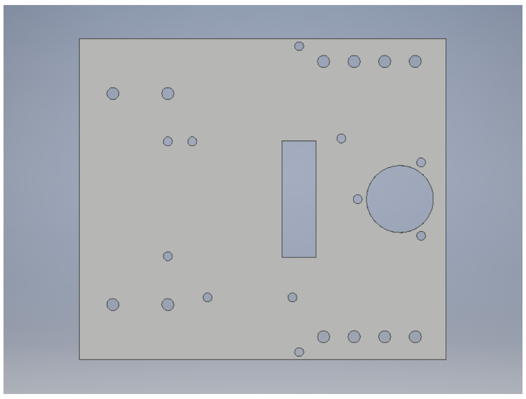

(b) Back Part

Figure 11: Chassis Frame

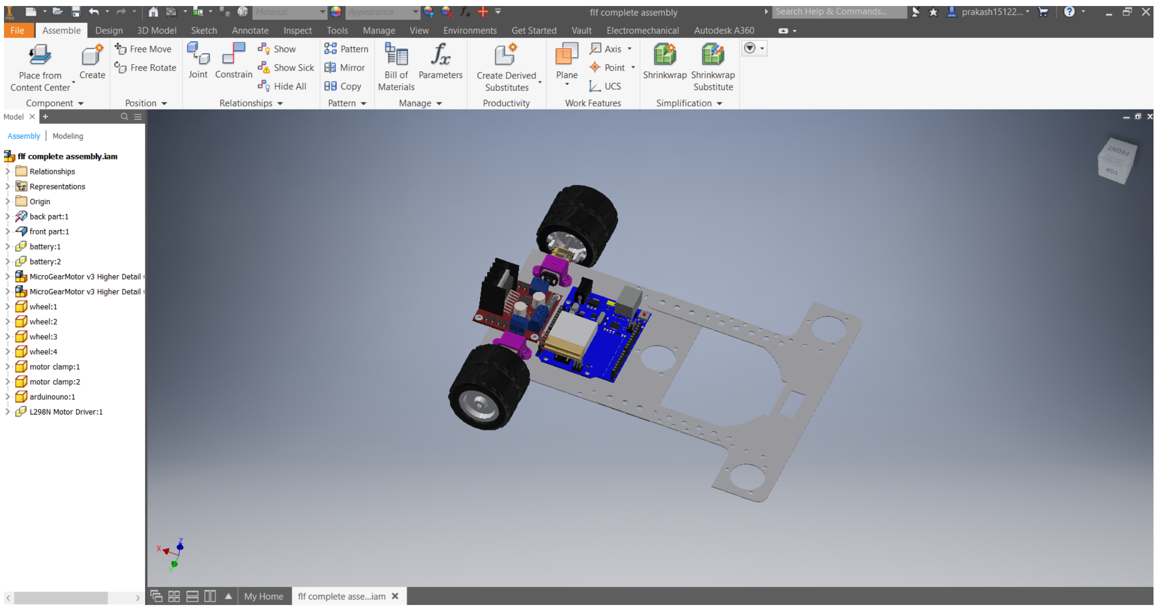


Figure 12: Assembled CAD with Electronics

5.2 Arena

5.2.1 Ideation

In order to properly test the working of the robot, a sufficiently large line following arena was required. Our inital idea was to stick together white chart papers and make the lines on them using black electrical tape. However this idea had many flaws - The chart paper would be difficult to maintain, get deformed and covered with dust easily and in general be very less durable.
Hence we decided to print the arena on a flex sheet. Most competition arenas are also built using flex sheets, as they have many advantages - very easy to maintain, extremely durable and also allows uniform printing, being almost as expensive as the chart paper. For the design of the arena, we first thought of simply printing some line follower arenas obtained via Google Image Search onto the flex sheet. However, this turned out to be inadequate as most of them were low quality and did not contain all the features that the competition arena would have. We hence chose to refer to the sample arena provided on the website of the competition, as it would be a good reference to the actual competition arena. It had all the features that had to be tested - lines, curves, loops and dead ends except discontinuities in the line.

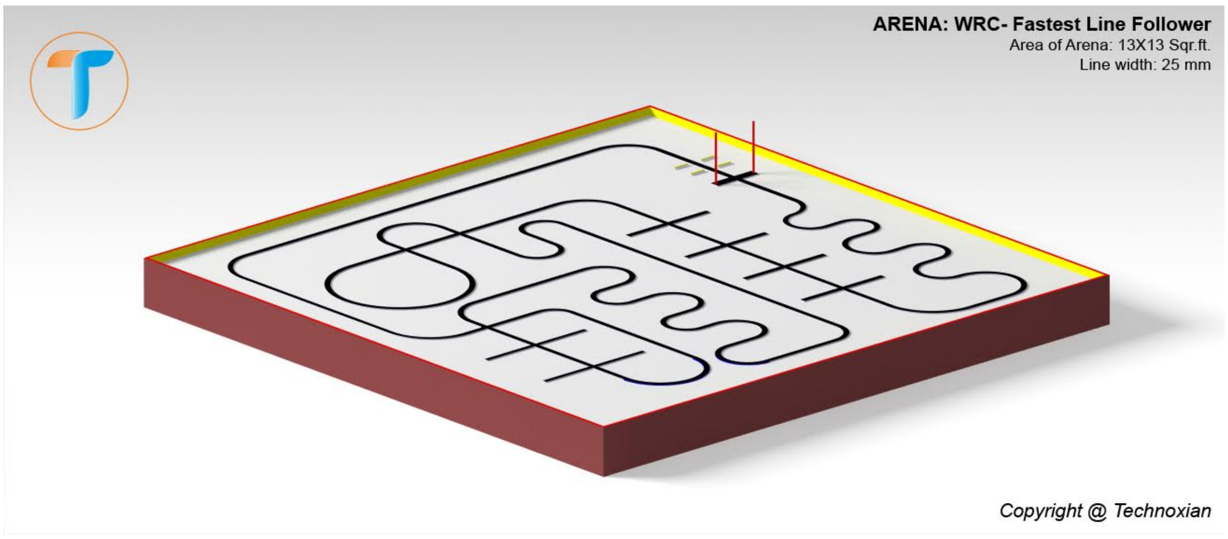

ARENA: WRC- Fastest Line Follower
Area of Arena: 13X13 Sqr.ft.
Line width: 25 mm
Copyright @ Technoxian

Figure 13: Sample Arena provided for the competition

5.2.2 Implementation

The arena was printed on a flex sheet of dimensions 3m X 3m. It was first digitally designed using GIMP - a lightweight, open-source, free photo editor that provides almost all the basic features one finds in Photoshop.
An exact to scale 3m x 3m image was created in GIMP so that there would be no chance of any pixelation that would occur on scaling or resizing a not-to-scale image before printing. The lines and curves were then drawn manually and carefully onto the image using the mouse and the Path Selection Tool provided in the editor GUI. Once the image had been completed, it was exported into a .tif file and then printed onto the flex sheet.

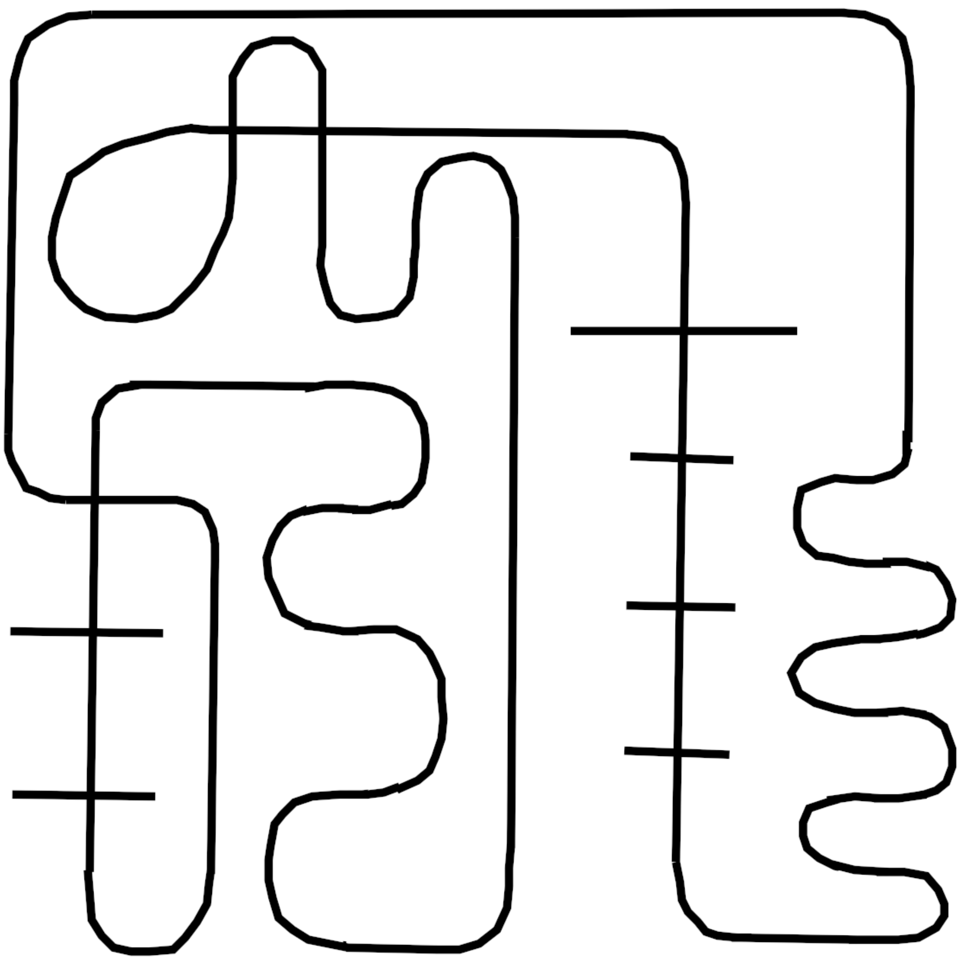


Figure 14: Arena created using GIMP

5.3 Algorithms

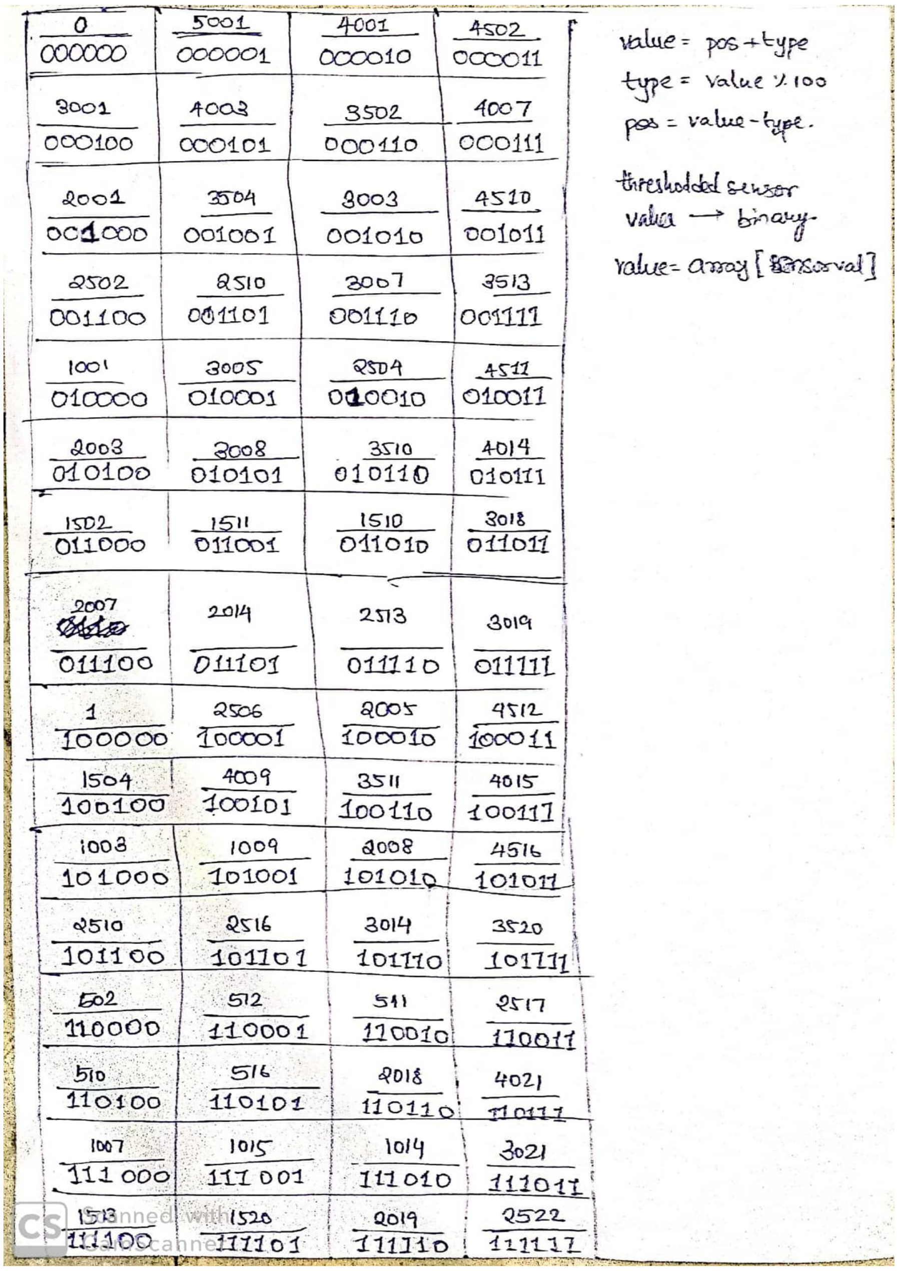
This section describes all the algorithms designed for the functioning of the robot.

5.3.1 Sensor Input

The LFS sensor has an array of 6 IR sensors, each of which output an integer in the range 0-1000 which is proportional to the ratio of reflected and transmitted light. Hence values towards the higher end correspond to black, while values on the lower end belong to white. A threshold value was hence set to determine whether to classify the sensor reading as black or white. This threshold value is affected by the height of the sensor, external lighting conditions and the surface's reflectivity.

5.3.2 Line Position

A 1D coordinate system is set up to pinpoint the position of the line with respect to the central axis of the end. The positive axis of this system points towards the right when the bot is viewed from above, the sensors are read in this direction. Since there are 6 sensors, after thresholding there remains a set of 64 possible configurations that the surface just below the bot can be in. In order to process all the possible configurations, a lookup array of length 64 was created. Each configuration is first converted into a binary number with 1 as black and 0 as white (for example, 'WWBBWW' gives 001100) which is then converted to base 10 to give the array index of its lookup value. The lookup value consists of a type value, which is less than 100, summed with the position value. (an example: the reading 'WBBWWW' converts to '011000', i.e array index 24. Its position value is 1500, and type is 2. The array value at index 24 is 1502.)



0	5001	4001	4502
000000	000001	000010	000011
3001	4003	3502	4007
000100	000101	000110	000111
2001	3504	3003	4510
001000	001001	001010	001011
2502	2510	3007	3513
001100	001101	001110	001111
1001	3005	2504	4511
010000	010001	010010	010011
2003	3008	3510	4014
010100	010101	010110	010111
1502	1511	1510	3018
011000	011001	011010	011011
2007	2014	2513	3019
011100	011101	011110	011111
1	2506	2005	4512
100000	100001	100010	100011
1504	4009	3511	4015
100100	100101	100110	100111
1003	1009	2008	4516
101000	101001	101010	101011
2510	2516	3014	3520
101100	101101	101110	101111
502	512	511	2517
110000	110001	110010	110011
510	516	2018	4021
110100	110101	110110	110111
1007	1015	1014	3021
111000	111001	111010	111011
1513	1520	2019	2522
111100	111101	111110	111111

value = pos + type

type = value % 100

pos = value - type.

thresholded sensor value $\longrightarrow$ binary.

value = array [sensorval]

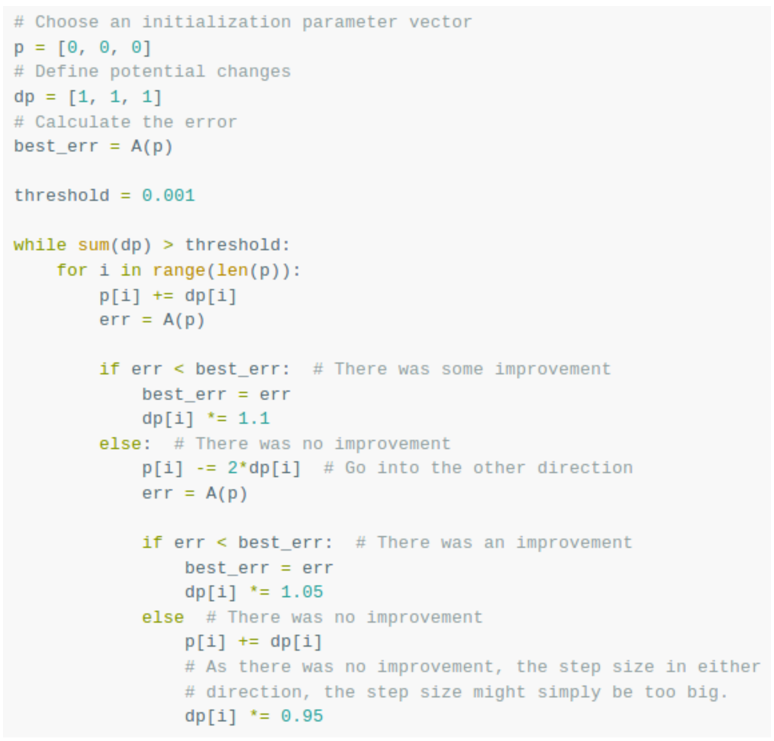
Figure 15: Lookup Table

5.3.3 PID Controller

At the heart of the line following algorithm lies a basic PID controller. The PID error is computed using the position of the line with respect to the defined coordinate system. At each step, firstly the sensor values are read. The lookup array is then used to fetch the position corresponding to the sensor reading just obtained. Since the position values are precalculated and stored in the lookup table, we save computation and time in each step since finding the position involves just one variable fetch. After obtaining the line position, the controller gains are applied to obtain the error as

$$\epsilon_i = K_p * (x_{t_i} - x_0) + K_d * (x_{t_{i-1}} - x_{t_i}) + K_i * \sum_{n=0}^{i} x_{t_n}$$

where x_{t_i} is the current line position, $x_{t_{i-1}}$ is the line position of the previous step, K_p, K_d, K_i are the controller gains and x_0 is the setpoint position, which corresponds to the central axis i.e a value of 2500 on the coordinate system. Initially, we began the controller tuning by manually setting the gains and running the bot on the arena again and again. This proved to be **very** tedious. In order to ease the process a bit, we decided to use the PID Tuner app, which sends gains to the bot via bluetooth through the HC05 module. Even then this process was time consuming as the gains were still set by trial and error. Hence we decided to use a coordinate descent based algorithm called Twiddle. The algorithm starts from zero gains and then increments or decrements each value at every step in order to move towards the minima. This optimization process can be to find parameters for any algorithm that outputs an error value.



```
# Choose an initialization parameter vector
p = [0, 0, 0]
# Define potential changes
dp = [1, 1, 1]
# Calculate the error
best_err = A(p)

threshold = 0.001

while sum(dp) > threshold:
    for i in range(len(p)):
        p[i] += dp[i]
        err = A(p)

        if err < best_err:  # There was some improvement
            best_err = err
            dp[i] *= 1.1
        else:  # There was no improvement
            p[i] -= 2*dp[i]  # Go into the other direction
            err = A(p)

            if err < best_err:  # There was an improvement
                best_err = err
                dp[i] *= 1.05
            else  # There was no improvement
                p[i] += dp[i]
                # As there was no improvement, the step size in either
                # direction, the step size might simply be too big.
                dp[i] *= 0.95
```

Figure 16: Pseudocode for the Twiddle Algorithm. A is an algorithm that outputs error. Image Source: https://martin-thoma.com/twiddle/

5.3.4 Left-Hand Algorithm

In order to map the intersections, loops and dead-ends of the arena, we decided to apply a simple left-hand algorithm. Whenever the bot detects an intersection, it turns left. If it detects no line in front of it, it turns anticlockwise until it has found the line again. These two actions constitute the left-hand algorithm and allow the bot to map the arena while staying on course to the finish.

6 Assembly

This section describes the assembly of the robot.

6.1 Bot

Once the chassis parts had been manufactured, they were joined together using nuts and screws. Then the casters, motors, mounting brackets and wheels were screwed onto the frame. The sensors, motor driver and Arduino MEGA were also fixed according to the designated slots on the frame.

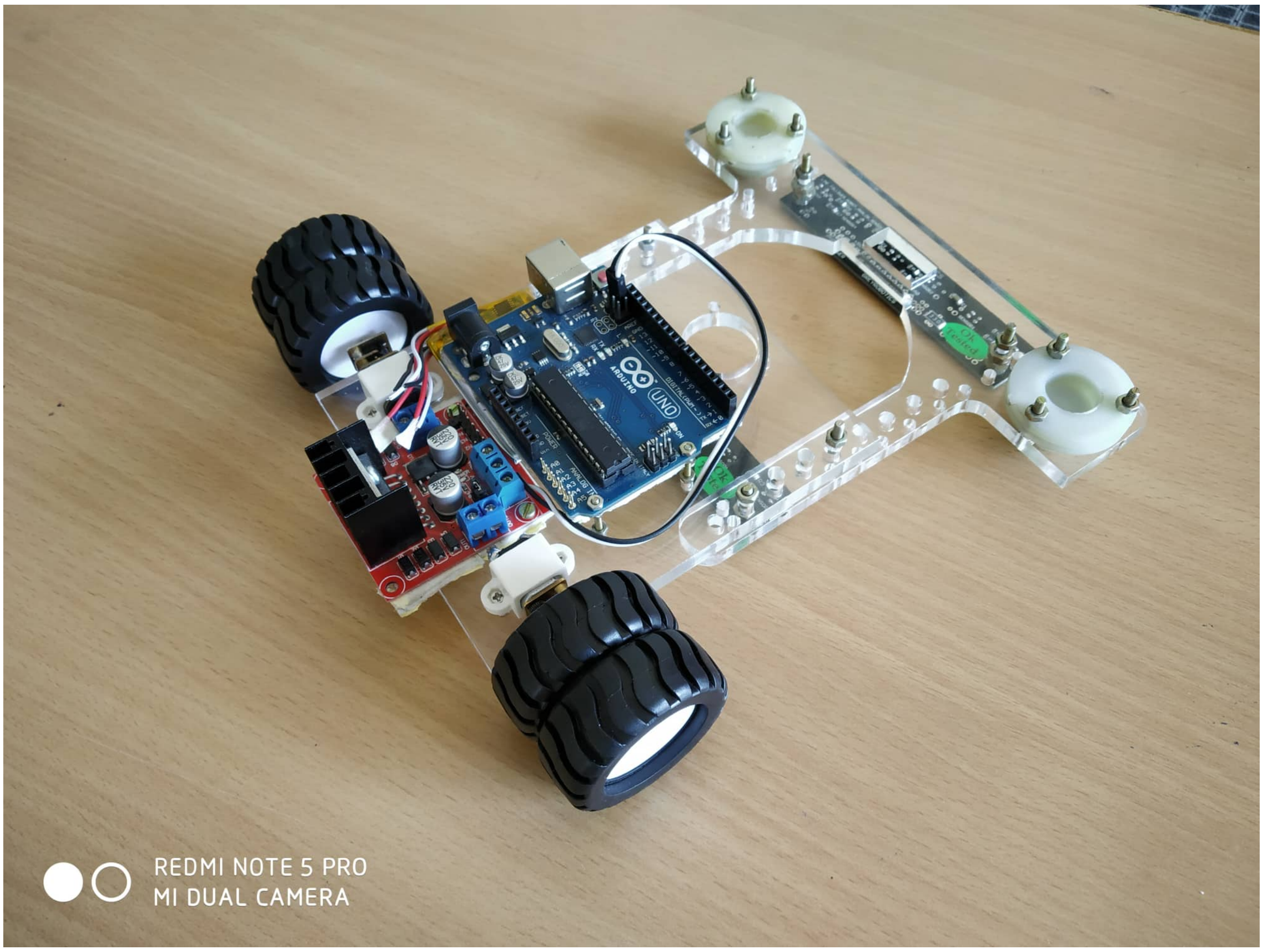


Figure 17: Assembled bot

6.2 Electronics

This section covers the assembly of all the electronic components, detailing the pin connections of each individual component.

6.2.1 Motor Driver

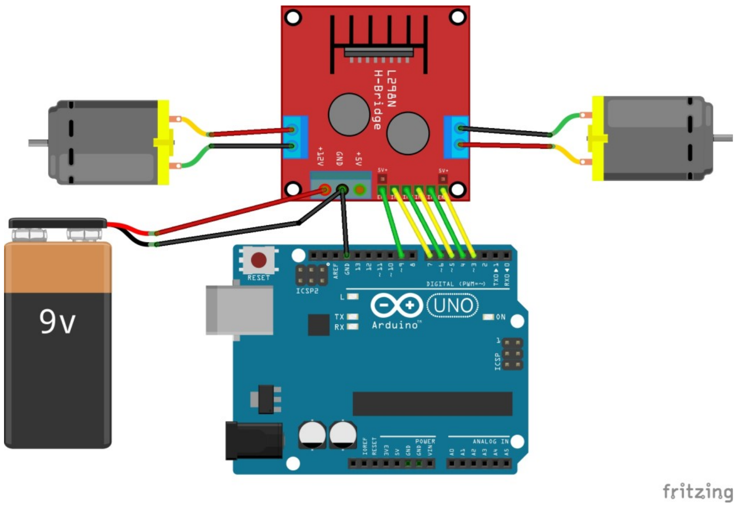


Figure 18: Connection Diagram

`Connections:`

- Out 1: Motor A lead out
- Out 2: Motor A lead out
- Out 3: Motor B lead out
- Out 4: Motor B lead out
- GND: Ground
- 5v: 5v Output
- 12v: 7-35V Input
- EnA: Enables PWM signal for Motor A
- In1: Input Motor A
- In2: Input Motor A
- In3: Input Motor B
- In4: Input Motor B
- EnB: Enables PWM signal for Motor B

`Important Points:`

- All the grounds (Arduino, Power source, and the Motor Controller) must be tied together.
- Since PWM features are being used, connect the motor driver pins to corresponding PWM-enabled pins on the controller.

`Arduino Sketch Considerations:`

Since there isn't a library for the L298N Dual H-Bridge Motor Controller, only declaring which pins the controller is hooked to is sufficient. The motor driver thus becomes very versatile in this project as a lot of the Arduino's pins are being used.
PWM pins that can be used on the MEGA are 2 to 13 and 44 to 46. These provide 8-bit PWM output with the analogWrite() function.

6.2.2 LFS Sensors

`Connections:`

- V_{cc} - Supply voltage
- GND - Common circuit ground
- A0-A5 - Analog output pins

6.2.3 Encoder Motor

`Connections:`

- V_{cc} - Supply voltage to encoder with Arduino's 3.3v pin
- c1 - encoder digital output1 pin with interrupt pin
- c2 - encoder digital output2 pin with interrupt pin
- m1 - motor input pin with motor driver's output pin1
- m2 - motor input pin with motor driver's output pin2

6.2.4 Battery

Two 3.7V batteries are connected in series in order to get desired voltage (7.4 V).

`Connections:`

- GND : with motor driver's ground
- Positive : with motor driver's 12v pin

6.2.5 HC05

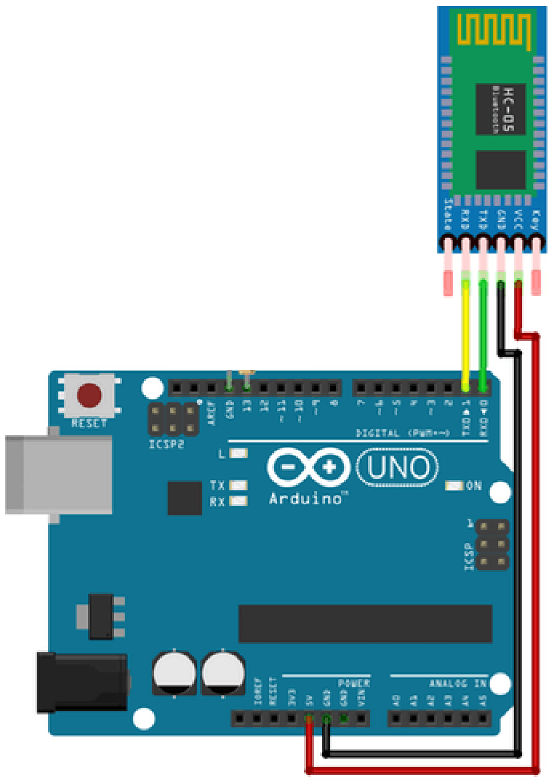


Figure 19: HC05 Pin Connections with Arduino

Pin	Description	Function
VCC	+5V	Connect to +5V
GND	Ground	Connect to Ground
TXD	UART_TXD, Bluetooth serial signal sending PIN	Connect with the MCU's (Microcontroller and etc) RXD PIN.
RXD	UART_RXD, Bluetooth serial signal sending PIN	Connect with the MCU's (Microcontroller and etc) TXD PIN.
ENABLE/KEY	Mode switch input	If it is input low level or connect to the air, the module is at paired or communication mode. If it's input high level, the module will enter to AT mode.
STATE	Status/Feedback pin	The state pin is connected to on board LED, it can be used as a feedback to check if Bluetooth is working properly.

Table 8: Pin Definitions

`Connections:`

- VCC: Connect to +5V of Arduino
- GND: Connect to Ground of Arduino
- TXD: With digital pin 10 (defined as RXD)
- RXD: With digital pin 11 (defined as TXD)

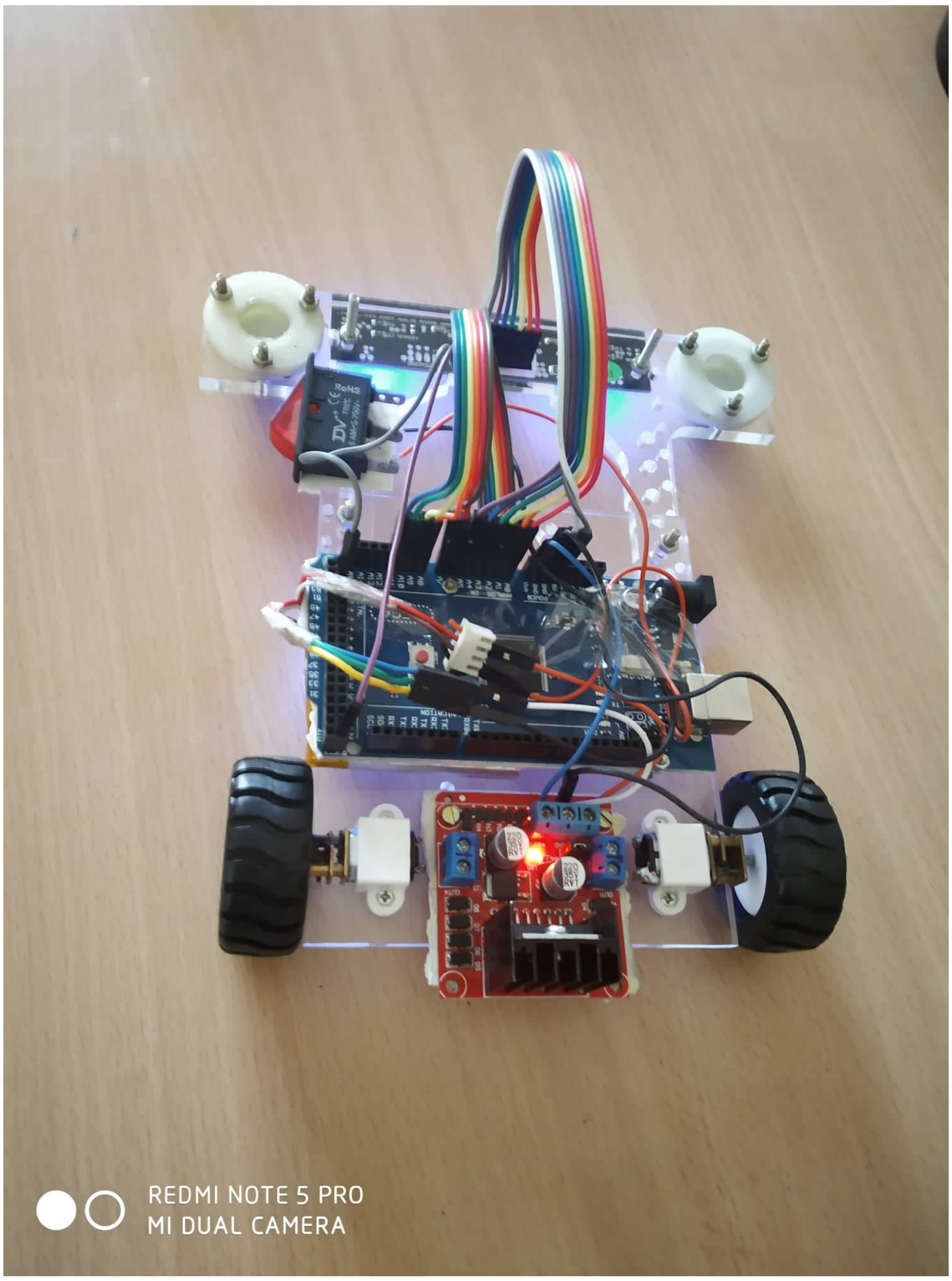


Figure 20: Assembled bot with electronics

7 Implementation

7.1 Testing Pipeline

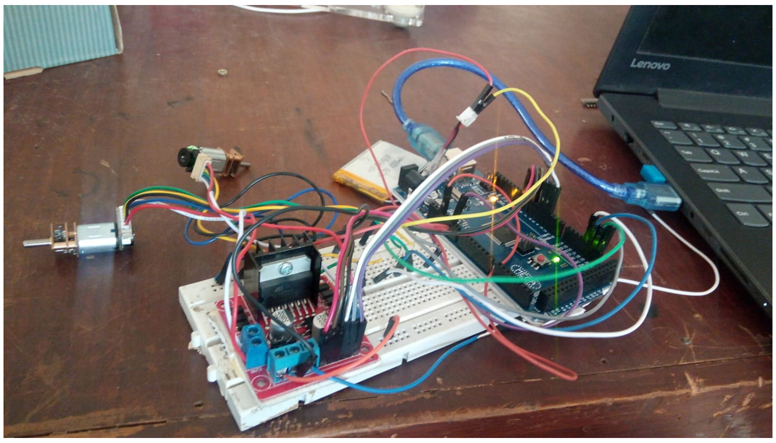


Figure 21: Motor Testing

Before implementing all the algorithms on the robot, we individually tested each module using a breadboard and only the required electronic components. An Arduino sketch implementing a specific module/algorithm was first written and compiled. The necessary connections were then made using the breadboard, the Arduino MEGA and other components. The compiled sketch was then uploaded onto the Arduino and its output was observed. If the output was as desired, the code was uploaded onto the GitHub repository. If the output was not as desired, changes were made to the sketch and it was tested again using the same procedure. Each algorithm module is individually described below:

7.2 Speed Control

The motor encoder contains a Hall sensor that returns 0 or 1 depending on whether the motor brushes are in front of it or not. This output is accumulated for a few cycles and then divided by the elapsed time to get a measure of the motor speed. The velocity from both the encoders is compared and the difference is used as a feedback for the motor speed input. Thus the speed of both motors is accurately synchronized.

7.3 Line Following

LFS creates an array of six reflectance integer value for the colour it sense. This value ranges from 0 to 1000 (nearly 0 for white colour and 1000 for black colour). Using these values, we calculate the position of the line with respect to the center of the sensor array. This output is then used as the process variable for a PID controller. The setpoint for the PID controller is set such that the line is exactly below the centre of the sensor array. The difference between the process variable and the setpoint value is used as feedback to control the speed of the motor and thereby the motion of the robot. The gains of the PID controller were tuned using the Twiddle algorithm. We repeatedly ran the algorithm, each time starting from the previous output and simultaneously decreasing the step size, until there was no change in the output. The gains were tuned to a precision of 6 decimal places.

7.4 Left Hand Algorithm

The left-hand algorithm was implemented on the readings of the front LFS. Using the lookup table, we identified the configurations that corresponded to a completely black line, which the bot sees when entering an intersection, and the ones that corresponded to no line, which the bot would see at a dead-end. Then these configurations are checked for before the controller actuates the bot. When the front LFS is in these configurations, the bot turns left until the readings of the front LFS again correspond to a centered line.

7.5 Data Relay

We used the HC05 bluetooth module to get the feedback of the LFS' 6 IR sensor readings. When the integer value of an single IR sensor corresponds to White or Black colour (i.e. around 0 or around 1000) it prints the character "W" or "B" respectively on the screen of the device connected wirelessly via the HC05 bluetooth module. Using this algorithm, we printed the array of six character corresponding to the white or black colour sensing of different IR sensor, which can be used to monitor the sensor output during runtime.

Bibliography

[1] Wheels. Website. [Online] https://robu.in/product/3pi-miniq-car-wheel-tyre-42mm-n20-dc-gear-motor-wh ?fbclid=IwAR2NCKTQOhppdMEaFgS3gBF2Gc6WcWmNtKCbgW6IM36N-en2OQRIRs5MXEQ.

[2] Caster. Amazon. [Online] https://techtonics.in/product/mini-3pi-car-n20-caster-robot-ball-wheel/.

[3] Mounting bracket. Website. [Online] https://techtonics.in/product/mini-3pi-car-n20-caster-robot-ball-wheel/.

[4] Lipo batteries. Website. [Online] https://www.electronicscomp.com/3.7v-1000mah-lipo-battery-india.

[5] Greyrobotics line following sensor. Website. [Online] https://www.amazon.in/GreyRobotics-Follower-Digital-Reflectance-Precision/dp/B07BCY5SJH/ref=pd_aw_sbs_328_2?_encoding=UTF8&pd_rd_i=B07BCY5SJH&pd_rd_r=5b3eace7-2de7-11e9-9866-7d0a2d898672&pd_rd_w=J6K2V&pd_rd_wg=liTVa&pf_rd_p=494e5557-afaf-46c4-b665-7fd8a9efbd16&pf_rd_r=2ZE62YEKJ4SHMRR2FD65&psc=1&refRID=2ZE62YEKJ4SHMRR2FD65&fbclid=IwAR1faL-ZI-GyMmOS7uZQVl6pHYF3jxPv-B92aKSEfhcSdU9V1Ay_BrTMrCc.

[6] Motors. Website. [Online] https://www.thingbits.net/products/micro-metal-geared-motor-w-encoder-6v-530rpm-30-1.

[7] Hc05 bluetooth module. Website. [Online] https://www.amazon.in/DIY-Retails-Bluetooth-Transceiver-Outputs/dp/B019OR9YVU?tag=googinhydr18418-21&tag=googinkenshoo-21&ascsubtag=_k_CjwKCAjwq-TmBRBdEiwAaO1en0d_ciqJC7gk7_DYheaMAxg3G3kNBEpt2DMO3rCI3CPJO4F13hPAPRoCZREQAvD_BwE_k_&gclid=CjwKCAjwq-TmBRBdEiwAaO1en0d_ciqJC7gk7_DYheaMAxg3G3kNBEpt2DMO3rCI3CPJO4F13hPAPRoCZREQAvD_BwE.

[8] Sample design guide. Instructables. [Online] https://www.instructables.com/id/High-performance-Line-follower-Robot/.

[9] Juing-Huei Su, Chyi-Shyong Lee, Hsin-Hsiung Huang, Sheng-Hsiung Chuang, and Chih-Yuan Lin. An intelligent line-following robot project for introductory robot courses. https://pdfs.semanticscholar.org/5b5a/cbe39571e0e41207fd7fefc96ec4bf72ef52.pdf, 2016–2018.

[10] Example line follower robots. YouTube. [Online] https://www.youtube.com/watch?v=7omDkur_fk8.