COMED KARES
Community Centric Education

REPORT ON

Robotics Internship Program

At ComedKares Innovation Hub, Bangalore.
(17/05/2023 – 05/06/2023)

by

ANKITHA D

1SB21EE003

SRI SAIRAM COLLEGE OF ENGINEERING

COMED KARES
Community Centric Education

Table of contents

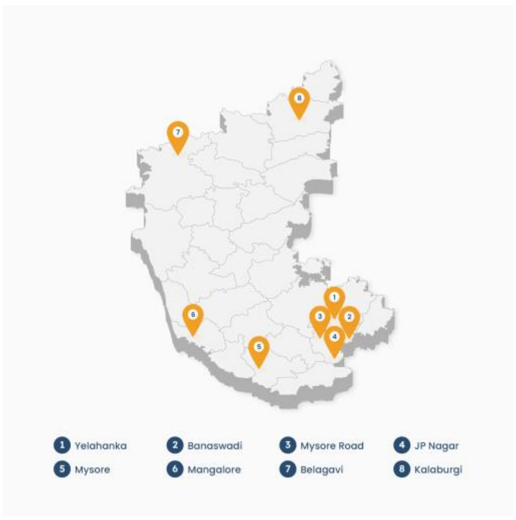
Karnataka Professional Colleges Foundation, in their endeavour to offer an effective, fair and objective testing procedure to determine merit of students seeking admission to the member institutions, have formed “Consortium of Medical, Engineering and Dental Colleges of Karnataka” (COMEDK). Professional education in engineering has about 125000 students who pass out into the employability arena every year. It is the idea of COMEDK to help these engineers get upgraded in their 21st Century skills as well as the latest technologies like AI, ML, IOT so that they are more employable and ready for industry.

ComedKares is an initiative by ComedK to help students of engineering, dental, and medical colleges choose the right career by building their competitiveness and exposing them to various opportunities available in the industry. ComedKares offers courses designed along with industry experts and endorsed by VTU that will enhance the learner’s 21st-century skills while also giving access to machines, tools and mentors to build products and solutions that positively impact the community. ComedKares catalyses the interaction between the Community (industry and regional professional groups), Industry experts and academic stakeholders for knowledge sharing and co-creating solutions through various programs spread across the year.

COMEDKARES CENTRES

World-class Innovation Hubs

ComedKares is establishing eight state-of-the-art Innovation Hubs across Karnataka, that empower students to become problem solvers, acquire competency in emerging technologies and develop solutions that have a positive impact on the community. ComedKares will conduct structured programs at the Innovation Hubs to equip the students with practical experience and give them hands-on exposure to advanced technologies, tools and machinery.



1 . Introduction : This course covers the nuances of building robotic applications. This is a completely hands-on course where students are introduced to a variety of physical and digital tools that can aid them in the process of building projects involving both hardware and software. The course is comprehensive with the use of development boards like Raspberry Pi & Arduino UNO for computer vision and motor control capabilities, benefiting students from a variety of domains. With projects involving concepts like OpenCV, GPIO interface and Serial communication along with Arduino and Python programming language, the course touches upon all the essentials that a product developer needs to create, all in a span of 12 weeks.

2. Objective: To built an automated bot using Raspberry pi and OpenCV libraries, understanding the python programming concepts.

3. Program Overview : This course covers the nuances of building robotic applications. This is a completely hands-on course where students are introduced to a variety of physical and digital tools that can aid them in the process of building projects involving both hardware and software. The course is comprehensive with the use of development boards like Raspberry Pi & Arduino UNO for computer vision and motor control capabilities, benefiting students from a variety of domains. With projects involving concepts like OpenCV, GPIO interface and Serial communication along with Arduino and Python programming language, the course touches upon all the essentials that student would require to build basic robots all in a span of 12 weeks.

I aim to provide a comprehensive overview of the hybrid course on robotics, highlighting its value in preparing students for a career in this exciting field. My experience will not only serve as a reflection of my own journey but also as a resource for future students to gain insights into the world of robotics and the efficacy of hybrid learning in this domain.

3.1. Details

Dates:
No. of days:

3.2 Schedule

Day	Time	Session
Day 1	9:30 - 1:00	◆ Arduino pinout and Arduino IDE Software ◆ Coding for blinking of light, traffic light ◆ Gaming Console
	2:00 - 4:30	◆ Gaming Console using Arduino
Day 2	9:30 -1:00	◆ Use of Servo Motor ◆ Angle rotation of servo motor ◆ Purpose of motor drivers ◆ Pin connections
	2:00 -4:30	◆ Servo motor Angle Rotation ◆ Motor rotation
Day 3	9:30 - 1:00	◆ Working of ultrasonic sensor and parts of it ◆ Types of sensors and its applications ◆ Pinout connection ◆ Parking assistance system ◆ RC Control Bot
	2:00 - 4:30	◆ Parking Assistance System ◆ RC Control Bot
Day 4	9:30 - 1:00	◆ Built an obstacle-detecting BOT using a servo motor, Arduino, and L298N motor driver.
	2:00 - 4:30	◆ Built an obstacle-detecting BOT using a servo motor, Arduino, and L298N motor driver.
Day 5	9:30 - 1:00	◆ Continuation of building bot
	2:00 - 4:30	◆ Evaluation

Day	Time	Session
Day 6	9:30 - 1:00	◆ Introduction to python ◆ History of python, applications of python ◆ Types of comments, data types, variables, input from user ◆ Conditional statements ◆ Types of loops and how loops are used in python program ◆ Purpose of using append
	2:00 - 4:30	◆ Executed python programs in google colabs
Day 7	9:30 - 1:00	◆ Raspberry Pi (RPi) OS Installation with system configuration ◆ Format of SD card ◆ Inserting operating system to SD Card
	2:00 - 4:30	◆ Installed OS, formatted the SD card, inserted OS to it
Day 8	9:30 - 1:00	◆ Interfacing sensors with Raspberry Pi ◆ Understanding Classes and Functions n Python
	2:00 - 4:30	◆ Humidity and Temperature has been detected using DTH11 sensor
Day 9	9:30 - 1:00	◆ PWM pins ◆ Duty cycle ◆ PWM frequency ◆ Raspberry Pi interfacing with duty cycle ◆ GPIO zero ◆ Motor control of bot
	2:00 - 4:30	◆ Motor control of bot using Raspberry Pi
Day 10	9:30 - 1:00	◆ Built the line following bot using the IR sensors ,motor drivers, raspberry pi
	2:00 - 4:30	◆ Built the line following bot using the IR sensors ,motor drivers, raspberry pi
Day 11	9:30 - 1:00	◆ Continuation of building bot

Day	Time	Session
	2:00 - 4:30	◆ Evaluation
Day 12	9:30 - 1:00	◆ How to install OpenCV and OpenCV libraries ◆ How to connect camera to raspberry Pi, installing camera libraries ◆ Reading an image ◆ Resizing image ◆ Rotating an image ◆ Accessing Pi camera for Real time video
	2:00 - 4:30	◆ Resizing the video frame ◆ Capturing image from live feed video ◆ Capturing video from live feed video
Day 13	9:30 - 1:00	◆ HSV colour space ◆ Code for detecting colour ◆ Creating trackbars for masking ◆ Adjusting the colour code
	2:00 - 4:30	◆ Square detection based on colour ◆ Hexagon detection based on colour
Day 14	9:30 - 1:00	◆ Lane detection steps ◆ Grayscale conversion ◆ Edge detection ◆ Feature extraction ◆ Convolution ◆ Segmentation ◆ Hough transform ◆ Image filtering
	2:00 - 4:30	◆ Different types of edge detection techniques and code for edge detection using canny function ◆ Sobel operator edge detection techniques ◆ Laplacian of Gaussian operator ◆ Hough line transform

Day 1
The concepts learnt

- Arduino pinout and Arduino IDE Software
- Coding for blinking of light, traffic light
- Gaming Console

I acquired comprehensive knowledge about the Arduino pinout and proficiently utilized the Arduino IDE software. This allowed me to effectively program and control various electronic components. I successfully implemented coding techniques for blinking lights, traffic light simulations, and console operations, demonstrating my understanding of the concepts and practical application of the Arduino platform.

By studying the pinout diagram of popular Arduino boards, such as Arduino Uno, Mega, and Nano, I familiarized myself with the arrangement and functionalities of the digital pins, analog pins, and power pins.

Using the Arduino IDE software, I honed my programming skills by writing code for blinking lights. By manipulating the digital pins, I programmed LEDs to blink at specific intervals, employing loops and delays to achieve the desired effect. This exercise solidified my comprehension of coding logic and syntax within the Arduino environment.

Building upon the blinking lights project, I further expanded my coding expertise to simulate a traffic light system. By coding for different traffic light states, including green, yellow, and red, I created a realistic traffic light sequence using multiple LEDs. This project allowed me to showcase my ability to design and implement more complex coding structures.

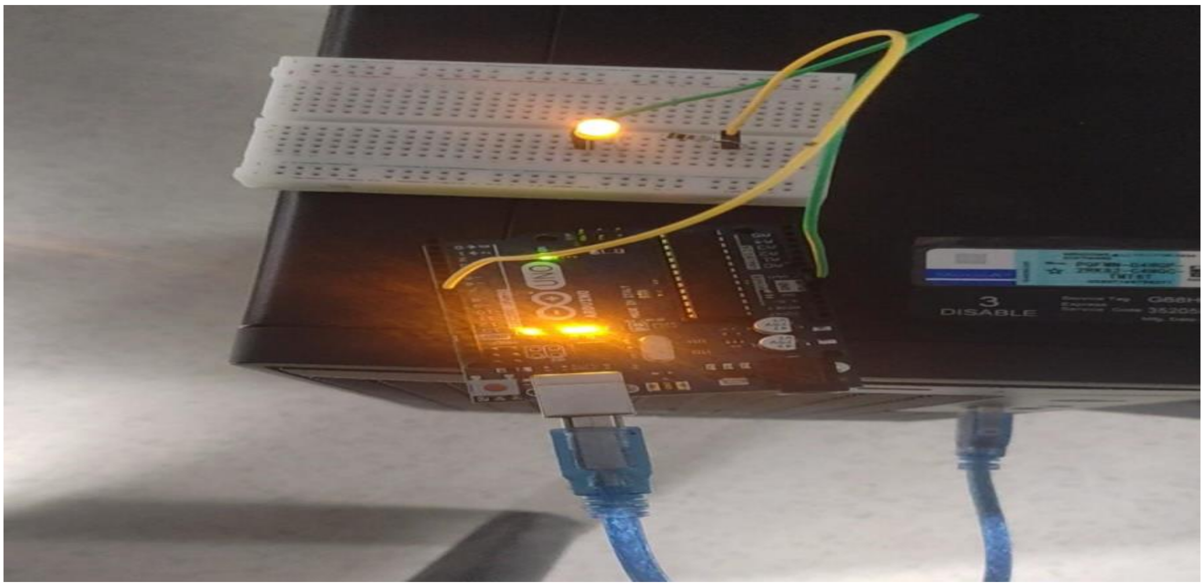


Fig 1.1: Blinking Of Light

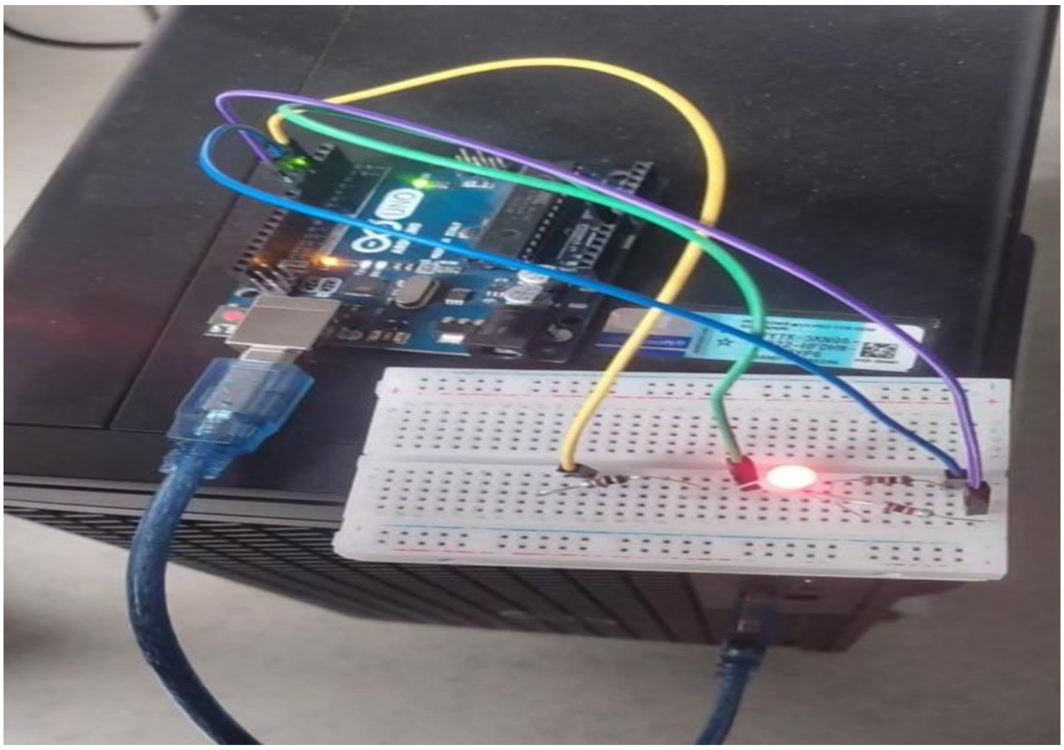


Fig 1.2: Traffic Light

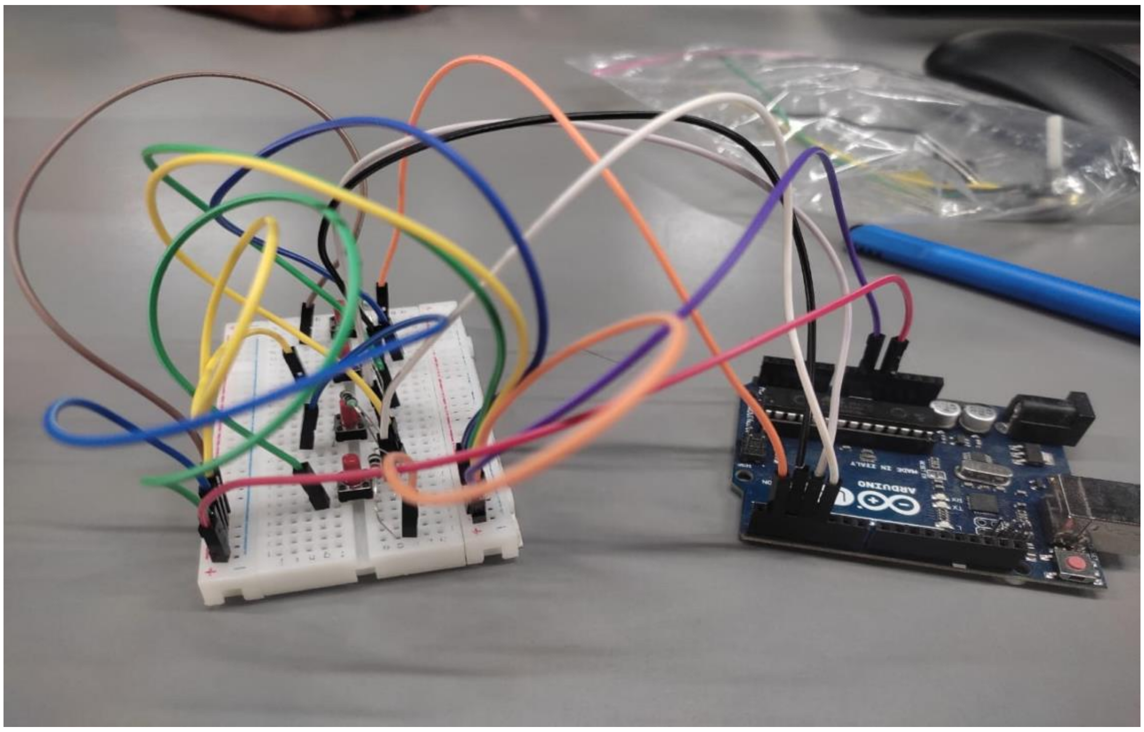


Fig 1.3: Gaming Console

Day 2
The concepts learnt

- Use of Servo Motor
- Angular rotation of servo motor
- Purpose of motor drivers
- Pin connections

I acquired a comprehensive understanding of servo motors and their applications. I learned how to control the angle rotation of servo motors through coding, allowing for precise movement and positioning in robotic systems.
I gained knowledge about motor drives and their significance in robotics. I explored the purpose of motor drives, which are electronic devices used to control the speed, direction, and torque of motors.

Furthermore, I grasped the importance of proper pin connections when working with servo motors. By referring to the pinout diagrams and specifications of the servo motor, I accurately connected it to the appropriate pins of the Arduino board and motor driver. This ensured seamless communication and control between the microcontroller and the servo motor.

Applying the knowledge gained, I successfully developed a robot using a servo motor, L298N motor driver, and Arduino. The integration of these components allowed for precise control and movement capabilities in the robot. Through systematic wiring and pin connections, I ensured the proper functioning and synchronization of the servo motor with the motor driver and Arduino.

To enable the desired movements and actions of the robot, I implemented coding techniques specific to the servo motor. By utilizing the servo library in the Arduino IDE, I wrote code that controlled the angle of rotation for the servo motor. This coding allowed me to program various motions and behaviours of the robot, creating a functional and interactive system.

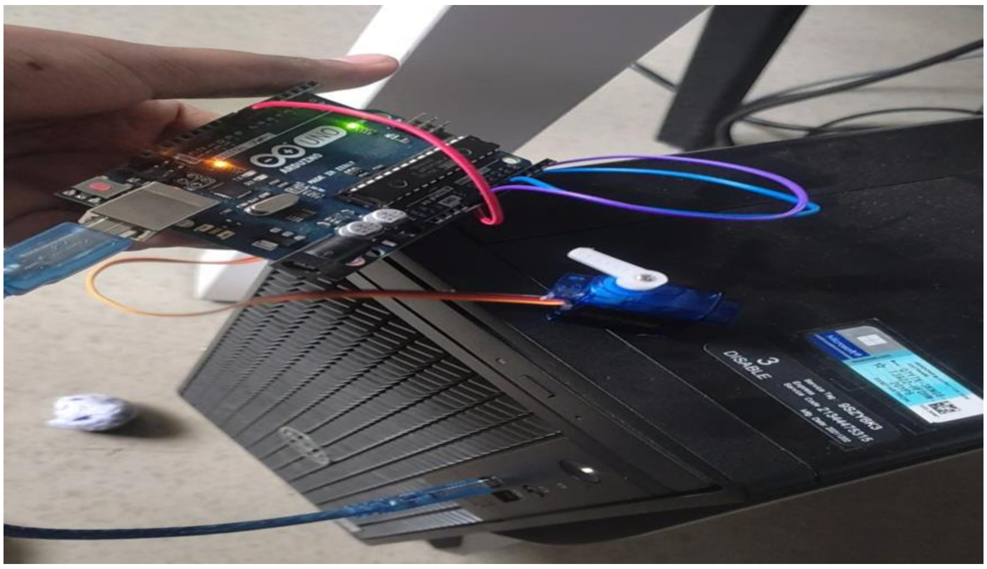


Fig 2.1: Angular rotation of Servo Motor

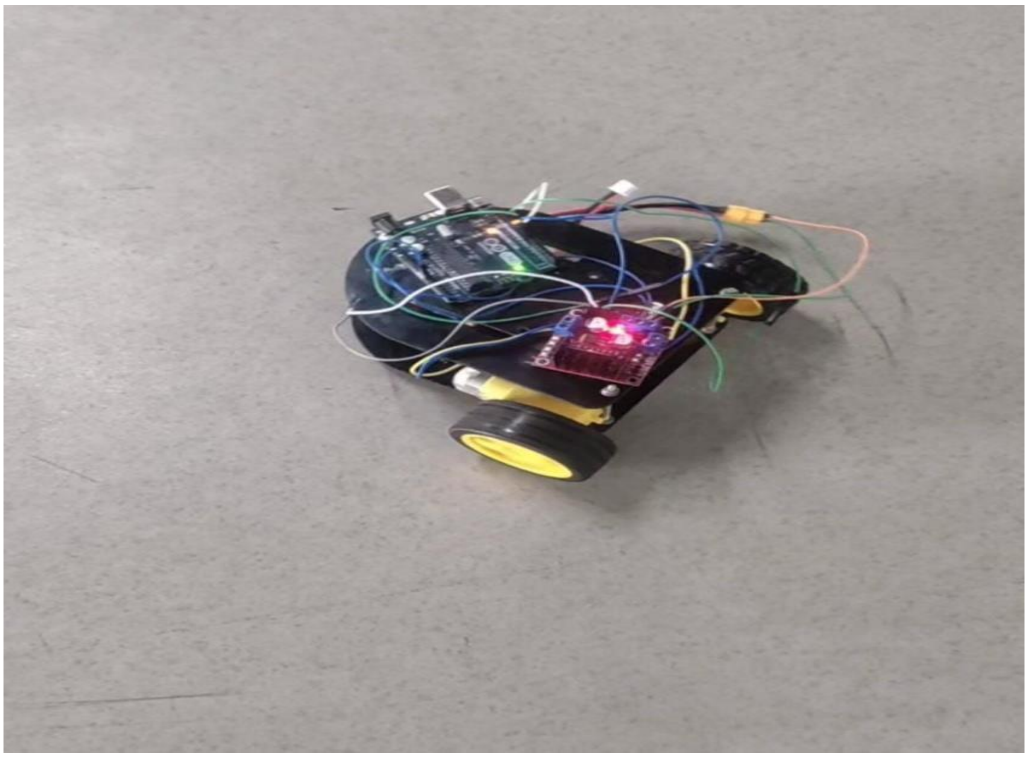


Fig 2.2 : Motor Rotation

Day 3

The concepts learnt

- Working of ultrasonic sensor and parts of it
- Types of sensors and its applications
- Pinout connection
- Parking assistance system
- RC Control Bot

I acquired a comprehensive understanding of ultrasonic sensors and their functionalities. I explored the inner workings and components of ultrasonic sensors, gaining insights into their ability to measure distance using sound waves. Additionally, I gained knowledge about different types of sensors and their various applications in the field of robotics.

I learned to program the ultrasonic sensor using the Arduino IDE. By leveraging the capabilities of the Arduino and the ultrasonic sensor library, I developed code that enabled distance measurement and object detection based on the sensor's readings.

To apply my knowledge, I successfully built a parking assistance system using an Arduino, ultrasonic sensor, light, and buzzer.
Additionally, I expanded my skills by constructing a remote-controlled (RC) bot. By integrating an Arduino board, motors, and other necessary components, I developed a bot capable of precise movement and control. Through coding, I programmed the Arduino to interpret signals from the RC controller and convert them into appropriate motor commands, enabling the desired movements and operations of the bot.

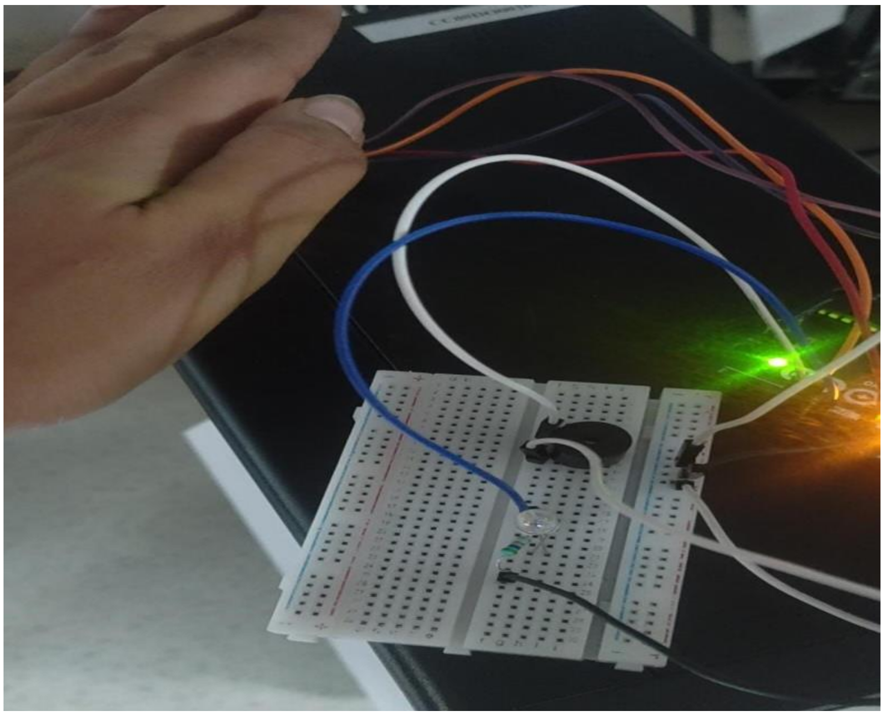


Fig 3.1: Parking Assistance System

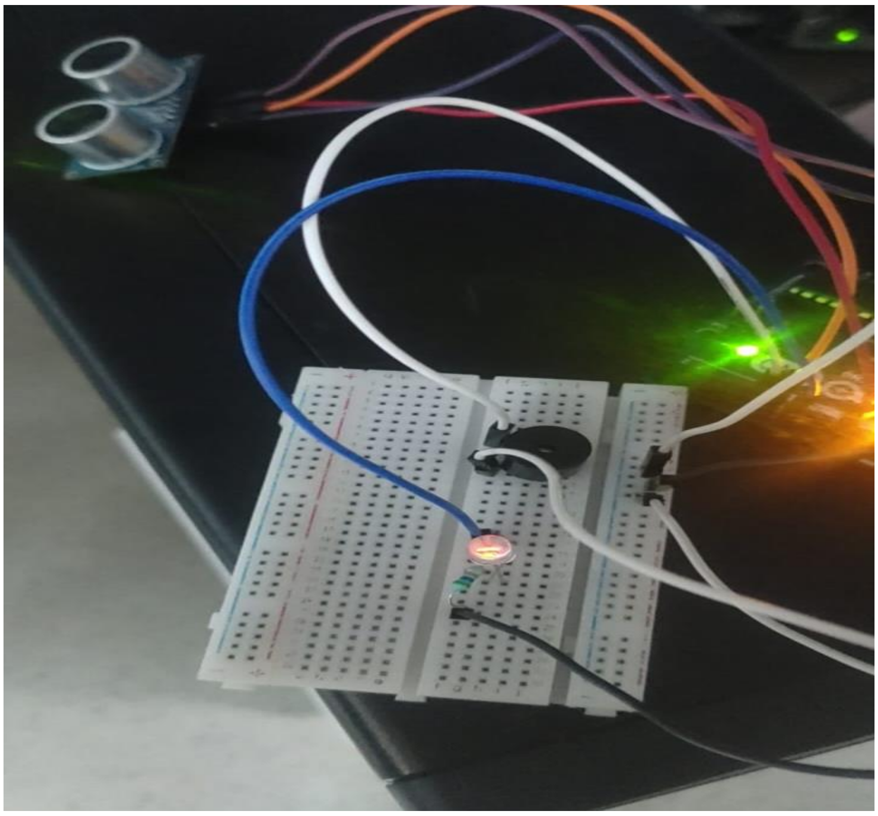


Fig 3.2: Parking Assistance System

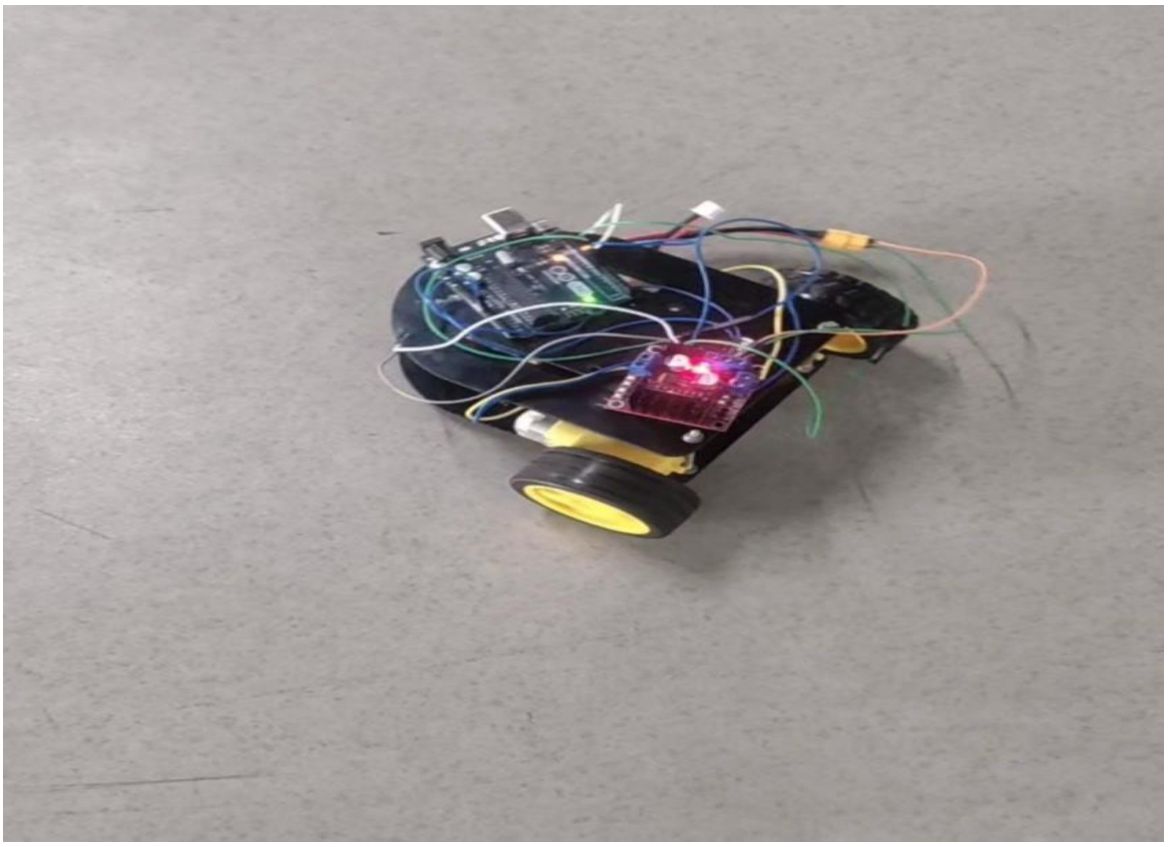


Fig 3.3: RC Control Bot

Day 4

The concepts learnt

- Built an obstacle-detecting BOT using a servo motor, Arduino, and L298N motor driver.

Introduction: The process of designing an obstacle-detecting BOT and setting up the necessary connections to operate it in an arena. The primary objective was to develop a BOT that could autonomously detect obstacles and navigate through the arena.

Connections: To establish the necessary connections, I carefully wired the components according to their specifications. The servo motor was connected to a suitable pin on the Arduino board, allowing it to rotate and scan the surroundings for obstacles. The L298N motor driver was connected to the Arduino to control the BOT's movement, including forward, backward, left, and right turns.

Software Implementation: The software implementation involved writing code using the Arduino IDE. The code included algorithms for servo motor control to enable obstacle detection through rotational movement. Additionally, it incorporated logic to interpret the obstacle detection signals and trigger appropriate actions for obstacle avoidance using the L298N motor driver.

Testing and Calibration: To ensure the functionality and effectiveness of the obstacle-detecting BOT, extensive testing and calibration were performed. Tests were conducted in the arena to evaluate the accuracy of obstacle detection, the responsiveness of the BOT's movements, and its overall performance in the designed arena. Calibration adjustments were made based on the test results to enhance the BOT's performance.

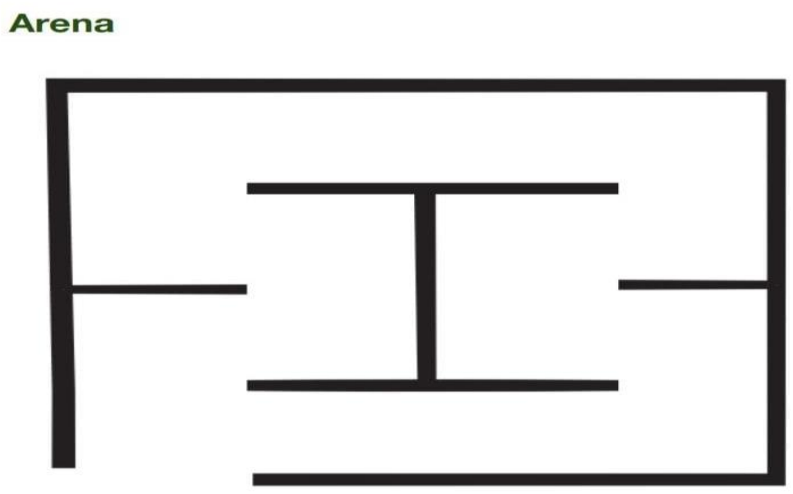


Fig 4.1 : Arena of Obstacle Detecting bot

Day 5
The concepts learnt

- Continuation of designing an obstacle-detecting BOT using a servo motor, Arduino, and L298N motor driver.

Recorrected the algorithm to enhance the obstacle-detecting bot capabilities, considering factors such as speed adjustment and obstacle detection.
And finally designed successfully working obstacle detecting BOT and passed the evaluation.

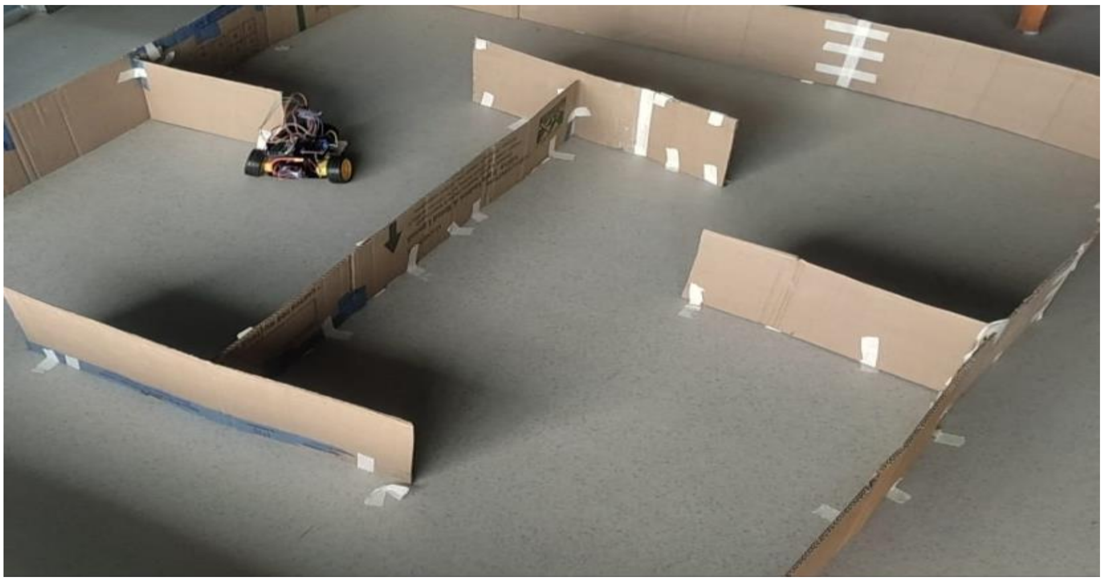


Fig 5.1 Obstacle-detecting BOT

Day 6

The concepts learnt

- Introduction to python
- History of python, applications of python
- Types of comments, data types, variables, input from user
- Conditional statements
- Types of loops and how loops are used in python program
- Purpose of using append

I received a comprehensive introduction to the language, starting with its history and evolution. Python, known for its simplicity and readability, has gained immense popularity due to its versatility and wide range of applications across various domains, including web development, data analysis, artificial intelligence, and more.

I learned about the use of comments in Python programming. Comments serve the purpose of providing explanations, clarifications, or notes within the code. Python supports two types of comments: single-line comments, denoted by the hash (#) symbol, and multi-line comments enclosed within triple quotes ('''comment''').

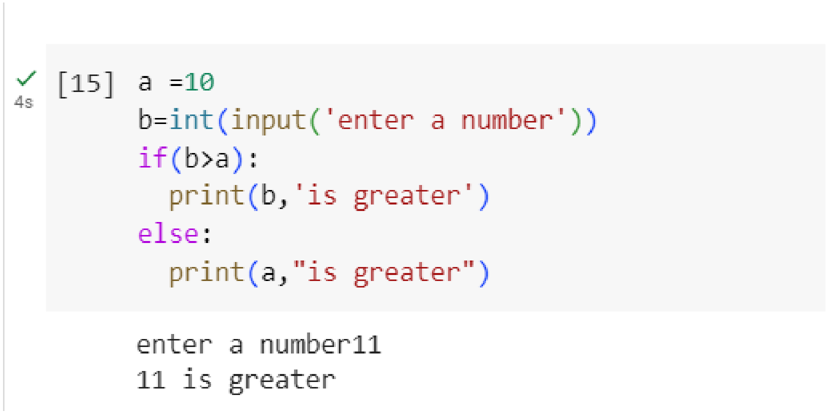
Next, I delved into the fundamental concepts of Python programming, starting with data types and variables. Python offers several built-in data types, including integers, floats, strings, lists, tuples, and dictionaries. These data types allow for efficient storage and manipulation of different kinds of information within a program. Variables, on the other hand, are used to store values and provide a means of accessing and modifying data during program execution.

I also explored the concept of user input, which enables programs to receive information from users during runtime. By utilizing the input() function. Conditional statements, such as if, elif, and else, were crucial in incorporating decision-making logic into my Python programs. These statements allowed me to create branching paths within the code, executing specific code blocks based on given conditions. This enabled program flexibility and adaptability based on different scenarios.

Loops played a vital role in repetitive tasks and iterative operations within Python programs. I learned about two types of loops: the for loop, which iterates over a sequence or range of values, and the while loop, which repeats until a specific condition is no longer true. By utilizing loops, I successfully implemented iterative actions and performed operations on lists, such as appending elements or segregating even and odd numbers.

In Python, the append() function is commonly used to add elements to lists dynamically. I gained hands-on experience in using this function to create arrays or lists, defining their structure, and dynamically populating them with elements as needed.

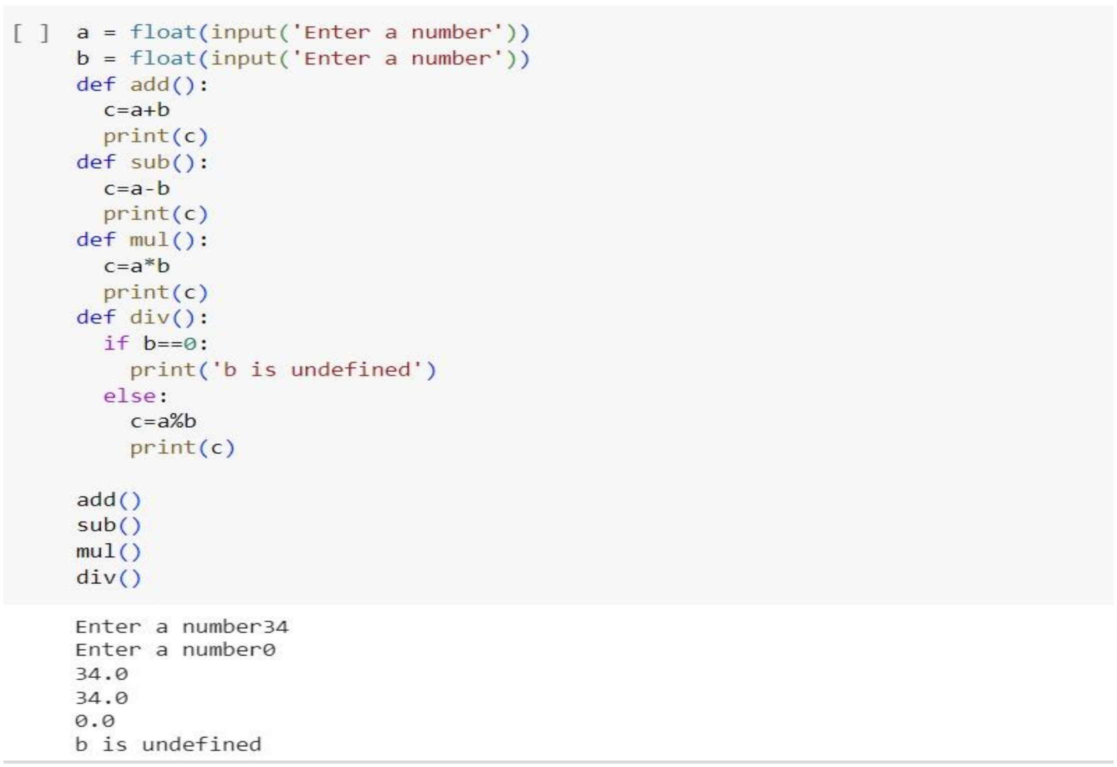
I further expanded my programming skills by working with functions in Python. Functions allowed me to encapsulate blocks of code, making them reusable and modular. I successfully created functions to determine if a given number is prime or not, find factors of a given number, and perform arithmetic operations on two numbers based on user input.



```
✓  [15]  a =10
4s       b=int(input('enter a number'))
         if(b>a):
           print(b,'is greater')
         else:
           print(a,"is greater")

         enter a number11
         11 is greater
```

Fig 6.1: Python Program to find greater number



```
[ ]  a = float(input('Enter a number'))
     b = float(input('Enter a number'))
     def add():
       c=a+b
       print(c)
     def sub():
       c=a-b
       print(c)
     def mul():
       c=a*b
       print(c)
     def div():
       if b==0:
         print('b is undefined')
       else:
         c=a%b
         print(c)

     add()
     sub()
     mul()
     div()

     Enter a number34
     Enter a number0
     34.0
     34.0
     0.0
     b is undefined
```

Day 7

The concepts learnt

- Raspberry pi (RPi) OS Installation with system configuration
- Format of SD card
- Inserting operating system to SD Card

Introduction: The Raspberry Pi (RPi) is a versatile and widely used single-board computer that requires an operating system (OS) to function.

1. Gathering Required Materials:
   - Raspberry Pi board
   - SD card (recommended: Class 10, minimum 8GB capacity)
   - Computer with an SD card reader
   - Internet connection
2. Choosing an Operating System:
   - Select an appropriate operating system for the Raspberry Pi, such as Raspbian (official OS), Ubuntu, or other specialized distributions.
   - Download the preferred OS image from the official Raspberry Pi website or the respective distribution's website.

3. Formatting the SD Card:
   - Insert the SD card into the computer's SD card reader.
   - Open the disk management utility (e.g., Disk Management in Windows or Disk Utility on macOS).
   - Locate the SD card and format it to the desired file system (typically FAT32).
   - Ensure that all existing data on the SD card is backed up, as formatting will erase all contents.
4. Writing the OS Image to the SD Card:
   - Download and install the Etcher software, a popular tool for writing OS images to SD cards.
   - Open Etcher and select the downloaded OS image file.
   - Choose the SD card as the target drive.
   - Verify that the correct drive is selected, as selecting the wrong drive may result in data loss.
   - Click on the "Flash" or "Write" button to initiate the process.
   - Wait for Etcher to write the OS image to the SD card and verify its integrity.
5. Inserting the SD Card into the Raspberry Pi:
   - Safely eject the SD card from the computer once the flashing process is complete.
   - Locate the SD card slot on the Raspberry Pi board.
   - Gently insert the SD card into the slot, ensuring it is properly seated.
   - Avoid applying excessive force, as it may damage the SD card or the Raspberry Pi.
6. Powering up the Raspberry Pi:
   - Connect the necessary peripherals (keyboard, mouse, display) to the Raspberry Pi.
   - Connect the power supply to the Raspberry Pi to provide it with power.
   - The Raspberry Pi will boot up and initiate the installed operating system.

By following these procedures, we successfully done setup of the Raspberry Pi and utilised it.

Day 8

The concepts learnt

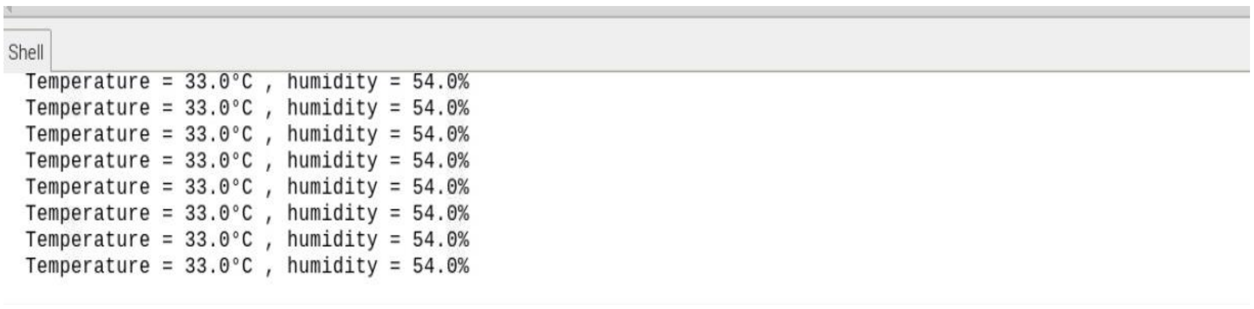
- Interfacing sensors with Raspberry Pi
- Understanding Classes And Functions In Python

1. Connection of IR Sensor for Alert System:
   - IR sensor is connected to the appropriate pins.
   - Developed a program that reads the sensor data and triggers an alert system (e.g., activating a buzzer) upon detecting any movement or interruption of the infrared beam.
   - Implemented logic and conditions to customize the alert system based on specific requirements.

2. Detection of Humidity and Temperature using DTH11 Sensor:
   - Connected the DTH11 sensor to the microcontroller.
   - Developed a program that reads the sensor data, including humidity and temperature values.
   - Utilized relevant libraries and functions to process and displayed the sensor readings.

3. Student Mark Analysis Program:
   - Created a program that interacts with the user and performs mark analysis using classes and functions.
   - Student details such as first name, last name, roll number, and the number of subjects has been given as input from the user.
   - Implemented a class to store the student information and mark details.

4. Develop functions to:
   - Accepted and stored the marks for each subject.
   - Sorted the marks in ascending order.
   - Calculated the average marks for all subjects.
   - Displayed the student's details along with the average and sum of marks.



```
Temperature = 33.0°C , humidity = 54.0%
Temperature = 33.0°C , humidity = 54.0%
Temperature = 33.0°C , humidity = 54.0%
Temperature = 33.0°C , humidity = 54.0%
Temperature = 33.0°C , humidity = 54.0%
Temperature = 33.0°C , humidity = 54.0%
Temperature = 33.0°C , humidity = 54.0%
Temperature = 33.0°C , humidity = 54.0%
```

Fig 8: Detection of Humidity and Temperature using DTH11 Sensor

Day 9
The concepts learnt

- PWM pins
- Duty cycle
- PWM frequency
- Raspberry Pi interfacing with duty cycle
- GPIO zero
- Motor control of bot

I acquired a comprehensive understanding of Pulse Width Modulation (PWM) pins and their functionalities. PWM allows for precise control of digital signals by varying the duty cycle and frequency of the signal. I learned how to interface Raspberry Pi with PWM pins

Utilizing the GPIO zero library, I gained proficiency in programming Raspberry Pi to control the duty cycle of PWM signals. This allowed me to regulate the speed and direction of motors. By adjusting the duty cycle, I could fine-tune the power output and achieve smooth and precise motor control.

By combining the knowledge of PWM pins, duty cycle, Raspberry Pi interfacing, and GPIO zero library, I successfully programmed the bot to perform desired tasks. The program encompassed efficient motor control, allowing the bot to navigate and respond to its environment effectively.

Fig 9: Motor Control Of BOT

Day 10

The concepts learnt

- Built the line following bot using the IR sensors ,motor drivers, raspberry pi.

1. Selection and Integration of Hardware Components:
    - IR Sensors: Suitable IR sensors which are capable of detecting the contrast between the line and the surface has been chosen. These sensors provide vital feedback for the robot's navigation.
    - Motor Drivers: Motor drivers that meet the requirements of the robot's motors and voltage specifications has been selected. Motor drivers play a crucial role in controlling the movement of the robot.
    - Raspberry Pi: Utilized a Raspberry Pi as the central control unit for the robot. The Raspberry Pi enables real-time data processing and facilitates motor control through programming.

2. Wiring and Connections:
    - Established the necessary connections between the motor drivers and the motors. proper power and signal connections has been enabled for motor control.
    - Connected the IR sensors to the appropriate GPIO pins of the Raspberry Pi.
    - Verified the wiring and connections to ensure proper communication and functionality between the components.

3. Programming and Algorithm Development:
    - Developed a program using Python and the Raspberry Pi's GPIO library to control the motors and process data from the IR sensors.
    - Designed an algorithm that enables the robot to detect the line and adjusted its movement accordingly. This algorithm typically involves reading the sensor inputs, interpreting the data, and implementing control logic to maintain alignment with the line.

4. Testing and Optimization:
    - Conducted testing of the line following robot to ensure its accurate navigation along the line.
    - Identified and addressed issues or challenges encountered during testing, such as sensor calibration, motor control precision, or algorithm responsiveness.
    - Optimized the programming and algorithm as necessary to improve the robot's line following performance.

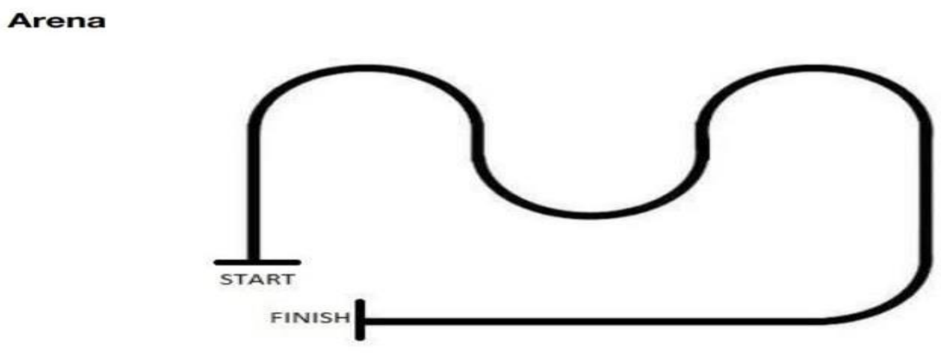


Fig 10: Arena of line following bot

Day 11

The concepts learnt

- Continuation of designing the line following bot using the IR Sensors, Motor Drivers, Raspberry Pi.

Again recorrected the algorithm to enhance the robot's line following capabilities, considering factors such as speed adjustment and responsiveness.
Refined the connections to make sure the bot is working as per required instructions. And finally designed successfully running line following bot and passed the evaluation.

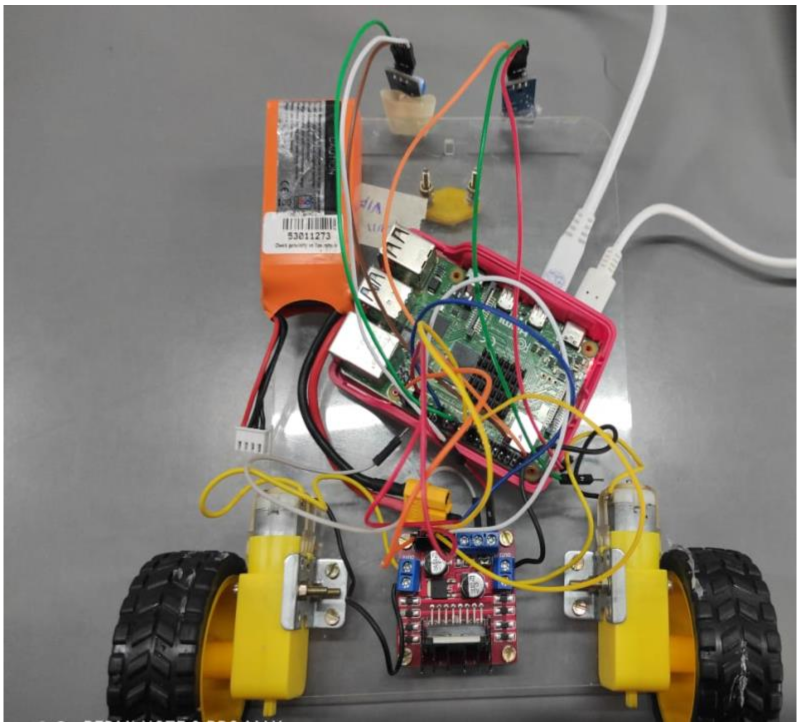


Fig 11.1: Line Following bot

Day 12

The concepts learnt

- How to install OpenCV and OpenCV libraries
- How to connect camera to raspberry Pi, installing camera libraries
- Reading an image
- Resizing image
- Rotating an image
- Accessing Pi camera for Real time video
- Resizing the video frame
- Capturing image from live feed video
- Capturing video from live feed video

I acquired comprehensive knowledge on installing OpenCV and its libraries, as well as connecting a camera to the Raspberry Pi. By following the necessary steps, I successfully installed OpenCV on the Raspberry Pi and established the camera's connection, enabling me to access its functionalities.

I gained proficiency in installing camera libraries specific to the Raspberry Pi, allowing for seamless integration and interaction between the camera and the Python programming environment.

Furthermore, I learned how to implement code for reading images using OpenCV. By using the library's functions and methods, I successfully loaded and processed images within Python, enabling me to perform various image manipulation tasks.

I acquired the skills to resize images, rotate images to different angles, and access the Raspberry Pi camera for real-time video processing. By utilizing OpenCV's functionalities, I achieved efficient resizing of video frames and real-time manipulation of video streams. This enabled me to adapt and modify the visual output according to specific requirements and applications.

I learned how to capture images from the live camera feed, utilizing the Raspberry Pi camera's capabilities. By implementing the appropriate code, I successfully extracted and saved images from the real-time video stream, enabling me to capture specific moments or objects of interest.

Expanding on image capture, I also gained the expertise to capture videos from the live camera feed. By employing OpenCV's video capture functionalities, I successfully recorded and saved video sequences from the Raspberry Pi camera, allowing for subsequent analysis or playback.

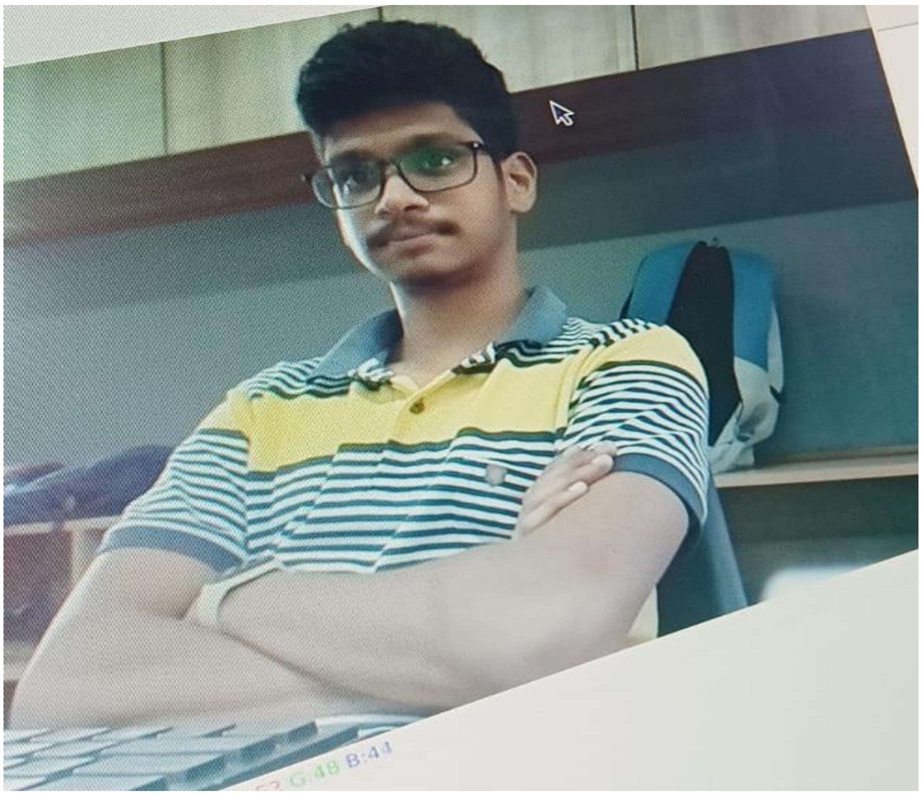


Fig 12.1: Capturing image from live feed video

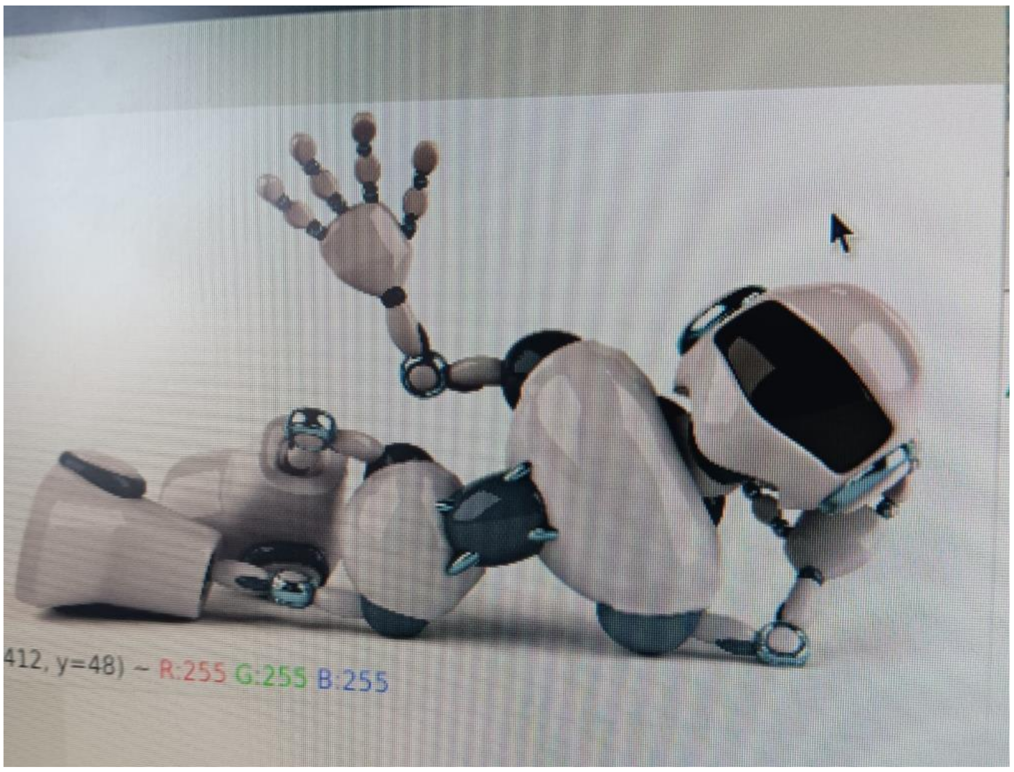


Fig 12.2: Resizing Image

Day 13
The concepts learnt

- HSV colour space
- Code for detecting colour
- Creating trackbars for masking
- Adjusting the colour code
- Square detection based on colour
- Hexagon detection based on colour

I gained a comprehensive understanding of the HSV (Hue, Saturation, Value) colour space and its significance in computer vision applications. I learned how to leverage the HSV colour space to effectively detect and identify specific colours within images or video streams.

By implementing code utilizing OpenCV, I successfully developed algorithms for colour detection. These algorithms allowed me to isolate and extract specific colours from an image or video feed, facilitating subsequent analysis or further processing.

To enhance the colour detection process, I learned how to create trackbars using OpenCV. These trackbars provided interactive interfaces for adjusting colour codes and parameters, enabling real-time manipulation and fine-tuning of the colour detection algorithm.

Building upon the colour detection capabilities, I successfully implemented algorithms for square detection based on colour. By leveraging the colour information, I created code that identified and outlined squares within an image or video stream, providing precise localization and recognition of square-shaped objects.

Expanding further, I applied my knowledge to detect hexagons based on colour. By utilizing the colour information and leveraging appropriate algorithms, I successfully developed code that identified and highlighted hexagonal shapes within images or video streams.

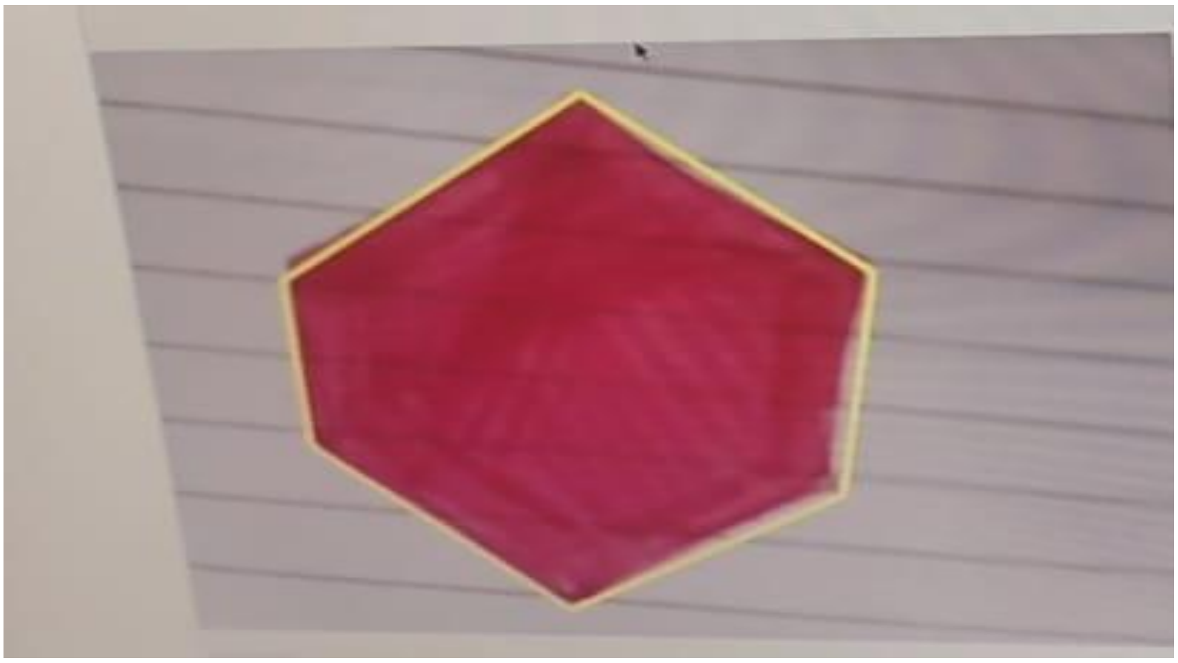


Fig 13.1: Hexagon Edge Detection

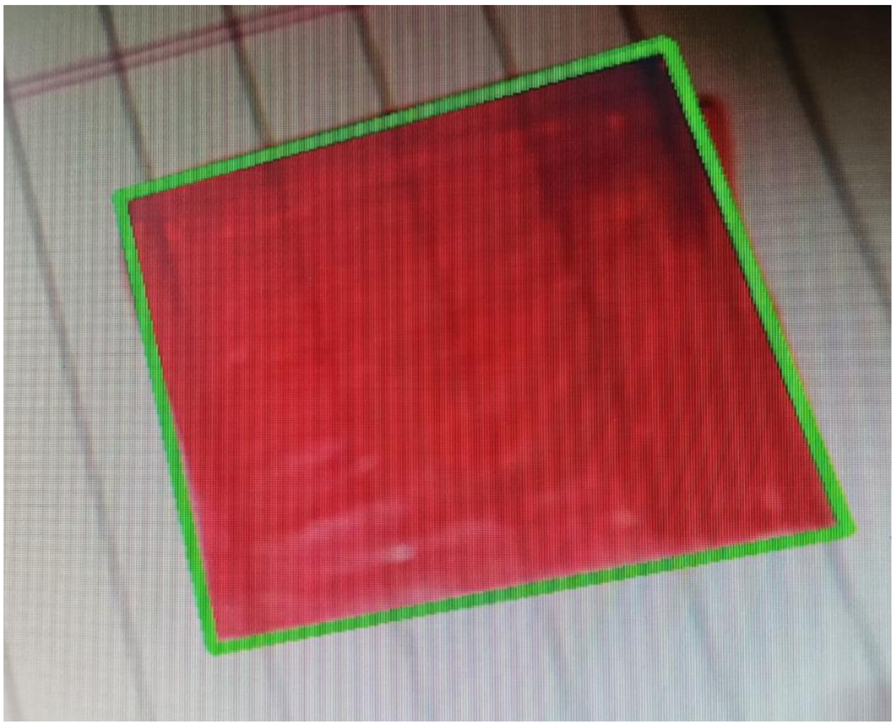


Fig 13.2: Square Edge Detection

Day 14
The concepts learnt

- Lane detection steps
- Grayscale conversion
- Edge detection
- Feature extraction
- Convolution
- Segmentation
- Hough transform
- Image filtering
- Different types of edge detection techniques and code for edge detection using canny function
- Sobel operator edge detection techniques
- Laplacian of Gaussian operator
- Hough line transform

I gained a comprehensive understanding of the lane detection process, encompassing several key steps. I learned how to convert images to grayscale, perform edge detection using various techniques, extract relevant features, apply convolution and segmentation methods, and employ the Hough transform for lane detection.

One of the fundamental steps I learned was converting colour images to grayscale. By converting images to grayscale, I effectively reduced the dimensionality and simplified subsequent processing steps, focusing solely on the intensity information of the image.

To identify potential lane edges, I explored different edge detection techniques. Through code implementation using OpenCV, I successfully applied the Canny edge detection function, which employs a multi-stage algorithm to detect prominent edges in the grayscale image. And also, I gained proficiency in using the Sobel operator for edge detection, which computes the gradient magnitude and direction to identify edges.

Furthermore, I learned about the Laplacian of Gaussian (LoG) operator, which combines the Laplacian edge detection and Gaussian smoothing techniques. By applying the LoG operator, I successfully detected edges with enhanced accuracy, especially in cases where noise was present in the image.

To identify lane markings from the detected edges, I implemented image filtering techniques such as convolution and segmentation. These techniques allowed me to enhance and isolate the lane markings, improving the accuracy of subsequent lane detection.

For the final stage of lane detection, I applied the Hough transform. This transformative technique enabled me to detect and extract lines from the edge image, representing the lane markings. By setting appropriate parameters and thresholds, I successfully identified and drew lines corresponding to the detected lanes.

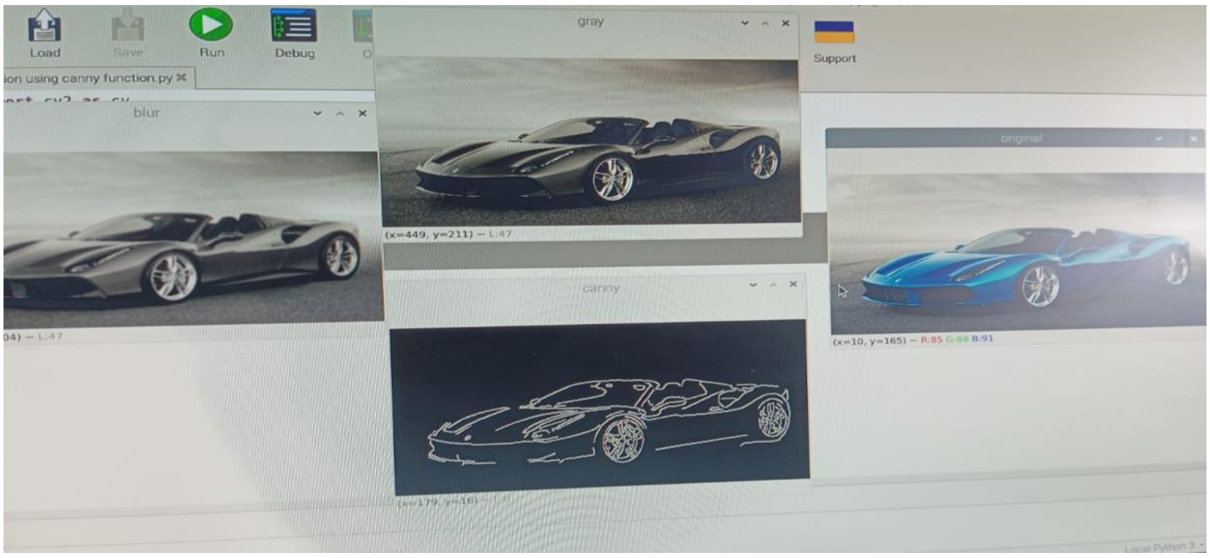
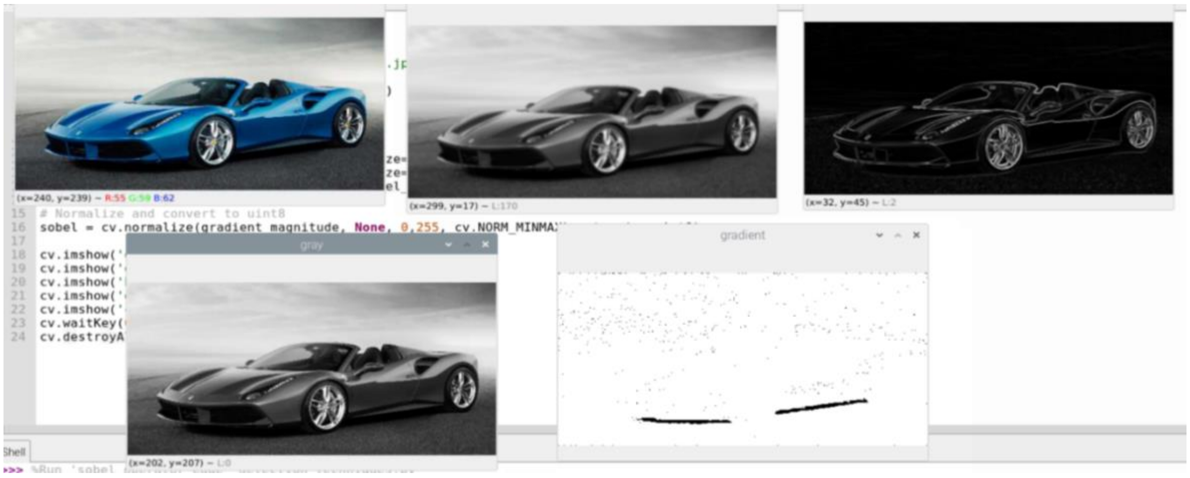


Fig 14.1: Edge Detection Using Canny



```
# Normalize and convert to uint8
sobel = cv.normalize(gradient_magnitude, None, 0,255, cv.NORM_MINMA

cv.imshow('
cv.imshow('
cv.imshow('
cv.imshow('
cv.imshow('
cv.waitKey(
cv.destroyA
```

```
>>> %Run 'sobel
```

Fig 14.2: Sobel Operator

7. Projects

1. Milestone 1

Objective: To be able to integrate sensors, actuators & Arduino UNO microcontroller for developing an Obstacle avoidance robot.

Challenge Brief: To build an Obstacle avoidance robot capable of avoiding obstacles using Ultrasonic sensor and Arduino UNO with integration to basic robot motion. There will be a boundary built with 4 walls, the robot has to avoid itself from colliding against the walls and traverse inside the defined perimeter.

Solution:

Code:

```
#define TRIGGER_PIN A1
#define ECHO_PIN A0
#define maximum_distance 200
 #include<NewPing.h>
 #include <Servo.h>
 NewPing sonar(TRIGGER_PIN, ECHO_PIN,maximum_distance );
Servo motor;
boolean goesForward=false;
int distance = 100;
 int speedSet = 0;
 void setup() {
 motor.attach(7);
motor.write(90);
 delay(2000);
distance = read1();
delay(100);
distance = read1();
delay(100);
distance = read1();
delay(100);
distance = read1();
delay(100);
 }
void loop() {
 int distanceR = 0;
int distanceL = 0;
delay(40);
//Finding
if(distance<=15)
 {
moveStop();
delay(100);
moveBackward();
delay(300);
moveStop();
```

```
 delay(200);
distanceR = lookRight();
 delay(200);
distanceL = lookLeft();
delay(200);
if(distanceR>=distanceL)
 {
turnRight();
 moveStop();
 }else
 {
turnLeft();
moveStop();
 }
}else
 {
moveForward();
}
distance = read1();
}
//Looking Right
 int lookRight()
 {
 motor.write(20);
 delay(500);
int distance = read1();
 delay(100);
motor.write(90);
return distance;
delay(100);
}
//Looking left
 int lookLeft()
 {
motor.write(170);
delay(500);
int distance = read1();
delay(100);
motor.write(90);
 return distance;
delay(100);
 }
//Reading Obstacle
 int read1()
 {
delay(70);
int cm = sonar.ping_cm();
 if(cm==0)
 {
 cm = 250;
 }
return cm;
 }
 //Stop
 void moveStop() {
digitalWrite(8,LOW);
digitalWrite(9,LOW);
digitalWrite(10,LOW);
 digitalWrite(11,LOW);
}
 //Forward
void moveForward() {
 if(!goesForward)
{
goesForward=true;
 digitalWrite(8,HIGH);
 digitalWrite(9,LOW);
digitalWrite(10,HIGH);
digitalWrite(11,LOW);
delay(10);
 }
      }
      //Backward
       void moveBackward() {
       goesForward=false;
       digitalWrite(8,LOW);
      digitalWrite(9,HIGH);
      digitalWrite(10,LOW);
      digitalWrite(11,HIGH);
      delay(10);
      }
       //Right
       void turnRight() {
      digitalWrite(8,LOW);
       digitalWrite(9,LOW);
      digitalWrite(10,HIGH);
      digitalWrite(11,LOW);
      delay(500);
       }
      //Left
      void turnLeft() {
      digitalWrite(8,HIGH);
       digitalWrite(9,LOW);
      digitalWrite(10,LOW);
      digitalWrite(11,LOW);
      delay(500); }
```

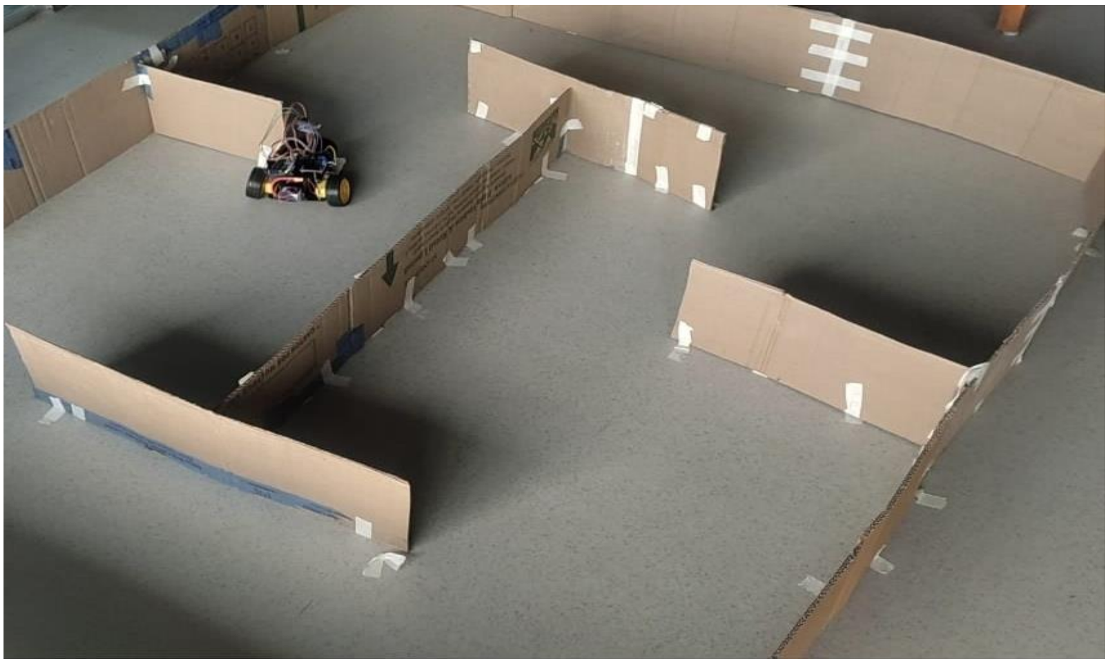


Fig 7.1 : Obstacle-Detecting BOT

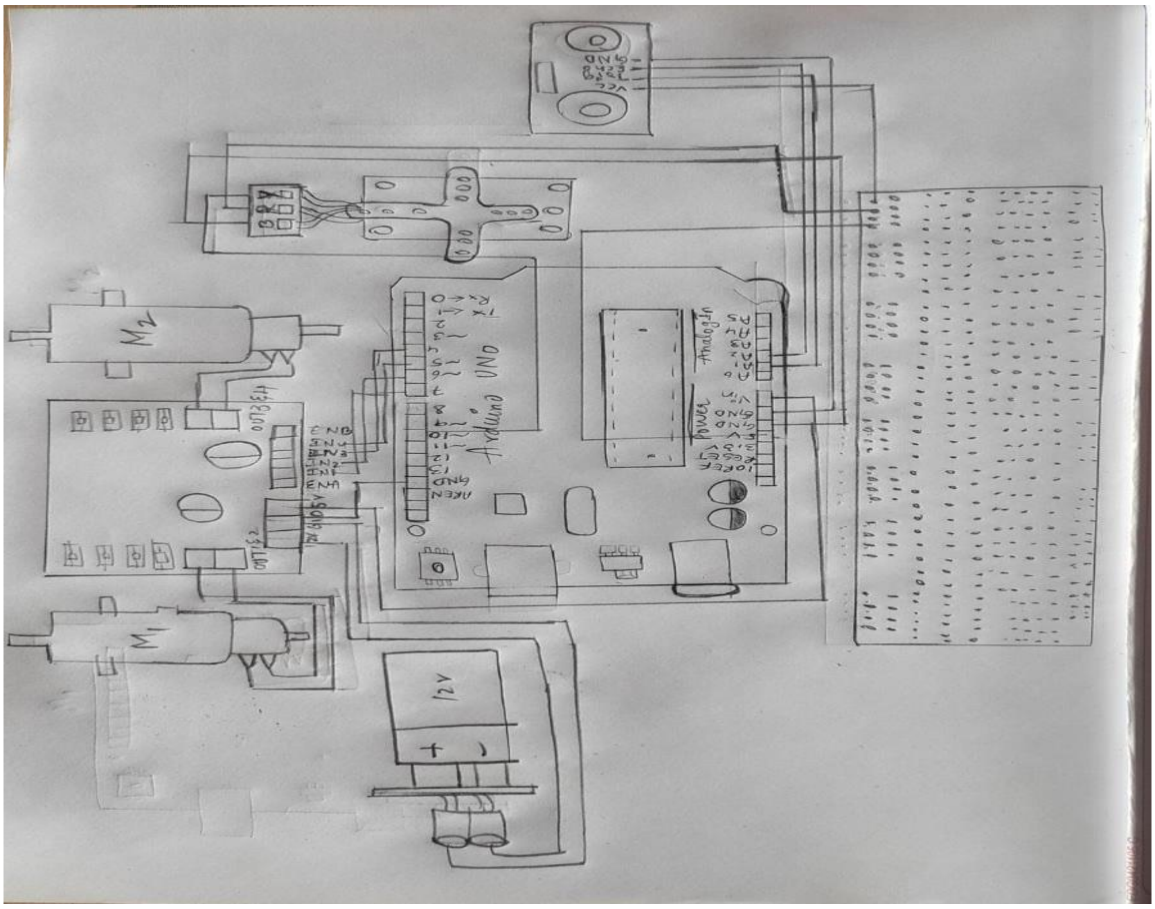


Fig 7.2: Circuit Diagram

2. Milestone 2

Objective: To be able to integrate sensors, actuators & Raspberry Pi for developing a line following robot.

Challenge Brief: The challenge is to build a line-following robot using IR Sensor and Raspberry Pi that can successfully navigate a lane with twists and turns. The lane is a black line on a white surface, and the bot should be able to follow the lane without deviating from it. The bot should be able to move forward, turn left or right, and stop when it reaches the end of the lane. It should also be able to adjust its speed based on the complexity of the lane, and it should be able to handle sharp turns and curves.

Solution:

Code:

```
import RPi.GPIO as GPIO
import time

# Define the GPIO pins for the sensors and the motors
sensor_left = 17
sensor_right = 22
motor_left_forward = 23
motor_left_backward = 24
motor_right_forward = 25
motor_right_backward = 26

# Set up the GPIO pins
GPIO.setmode(GPIO.BCM)
GPIO.setup(sensor_left, GPIO.IN)
GPIO.setup(sensor_right, GPIO.IN)
GPIO.setup(motor_left_forward, GPIO.OUT)
GPIO.setup(motor_left_backward, GPIO.OUT)
GPIO.setup(motor_right_forward, GPIO.OUT)
GPIO.setup(motor_right_backward, GPIO.OUT)

# Define the function to move forward

def move_forward():
    GPIO.output(motor_left_forward, GPIO.HIGH)
    GPIO.output(motor_left_backward, GPIO.LOW)
    GPIO.output(motor_right_forward, GPIO.HIGH)
    GPIO.output(motor_right_backward, GPIO.LOW)

# Define the function to move backward

def move_backward():
    GPIO.output(motor_left_forward, GPIO.LOW)
    GPIO.output(motor_left_backward, GPIO.HIGH)
    GPIO.output(motor_right_forward, GPIO.LOW)
    GPIO.output(motor_right_backward, GPIO.HIGH)

# Define the function to turn left

def turn_left():
    GPIO.output(motor_left_forward, GPIO.LOW)
    GPIO.output(motor_left_backward, GPIO.HIGH)
    GPIO.output(motor_right_forward, GPIO.HIGH)
    GPIO.output(motor_right_backward, GPIO.LOW)

# Define the function to turn right

def turn_right():
    GPIO.output(motor_left_forward, GPIO.HIGH)
    GPIO.output(motor_left_backward, GPIO.LOW)
    GPIO.output(motor_right_forward, GPIO.LOW)
    GPIO.output(motor_right_backward, GPIO.HIGH)

# Define the main loop

while True:
    sensor_1 = GPIO.input(sensor_left)
    sensor_2 = GPIO.input(sensor_middle)

    # If all sensors are on black line, move forward
    if (sensor_1 == 0 and sensor_2 == 0):
        move_forward()

    # If only left sensor is on black line, turn left
    elif (sensor_1 == 0):
        turn_left()

    # If only right sensor is on black line, turn right
    elif (sensor_2 == 0):
        turn_right()

    # If middle sensor is on black line, move forward
    else:
        move_forward()

    time.sleep(0.001)
```

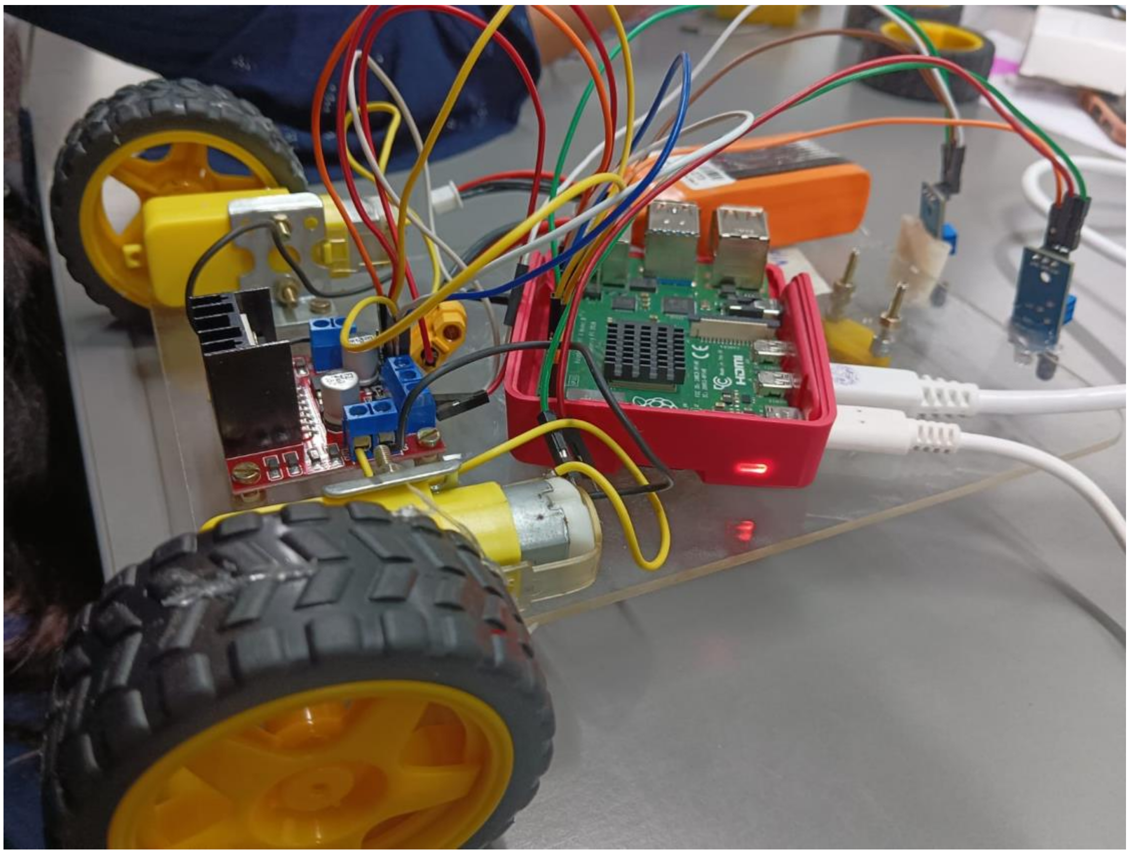


Fig 7.3: Built Solution

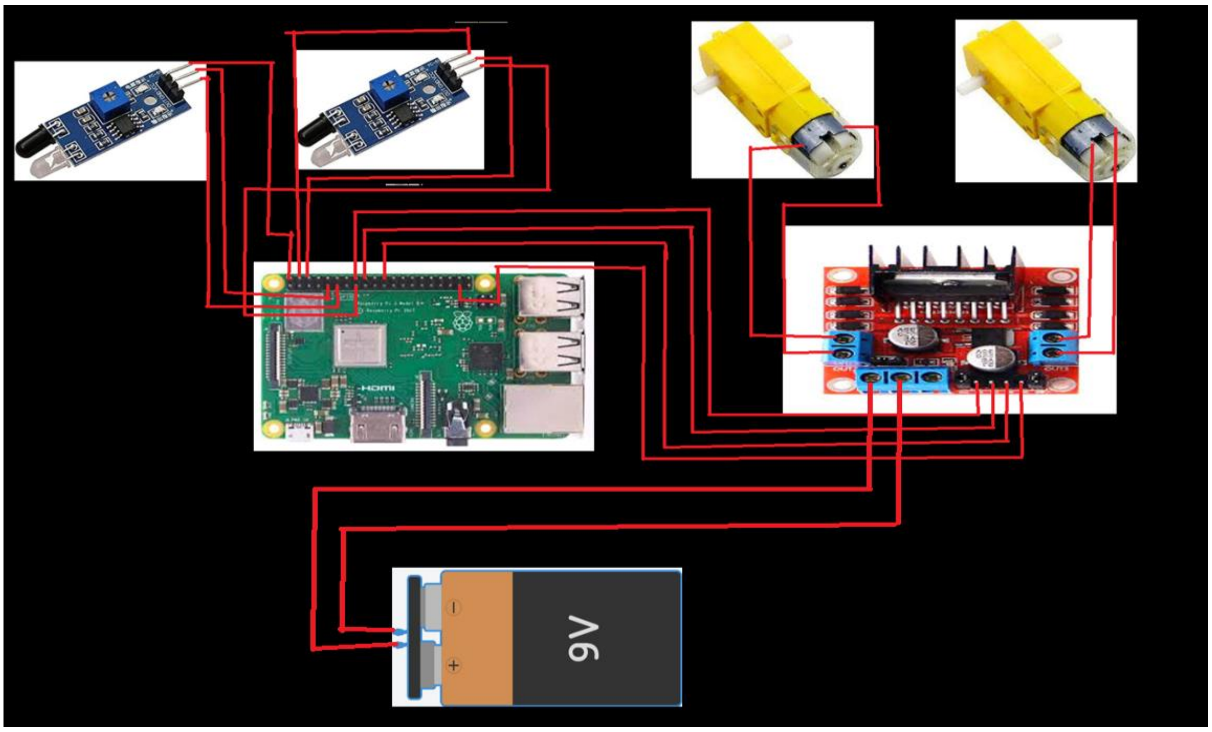


Fig 7.4: Circuit Diagram

3. **Milestone 3**

Objective: To be able to integrate Raspberry Pi cameras to detect lines or take turns based on colours.

Challenge Brief: The challenge is to build a lane-following robot using a Raspberry Pi camera which is integrated with Raspberry Pi that can successfully navigate a lane with twists and turns. The lane is a black line on a white surface, and the bot should be able to follow the lane without deviating from it. The bot should be able to move forward, turn left or right based on colours seen on the arena and stop when it reaches the end of the blue box. It should also be able to adjust its speed based on the complexity of the lane

Solution:

Code:

```
import numpy as np
import RPi.GPIO as GPIO
import cv2
import time

cap = cv2.VideoCapture(0)
cap.set(3, 160)
cap.set(4, 120)
cap.set(cv2.CAP_PROP_FPS, 20)
fps = int(cap.get(5))
print("fps:",fps)

in1 = 3 #17
in2 = 5 #27
in3 = 29 #5
in4 = 31 #6
enA = 32
enB = 33
c=""

GPIO.setmode(GPIO.BOARD)
GPIO.setup(enA, GPIO.OUT)
GPIO.setup(enB, GPIO.OUT)
GPIO.setup(in1, GPIO.OUT)
GPIO.setup(in2, GPIO.OUT)
GPIO.setup(in3, GPIO.OUT)
GPIO.setup(in4, GPIO.OUT)

p1 = GPIO.PWM(enA, 30)
p2 = GPIO.PWM(enB, 30)
p1.start(20)
p2.start(20)

def forward():
GPIO.output(in1, GPIO.HIGH)
GPIO.output(in2, GPIO.LOW)
GPIO.output(in3, GPIO.HIGH)
GPIO.output(in4, GPIO.LOW)
p1.start(18)
p2.start(18)

def backward():
GPIO.output(in1, GPIO.LOW)
GPIO.output(in2, GPIO.HIGH)
GPIO.output(in3, GPIO.LOW)
GPIO.output(in4, GPIO.HIGH)

def left():
GPIO.output(in1, GPIO.LOW)
GPIO.output(in2, GPIO.HIGH)
GPIO.output(in3, GPIO.HIGH)
GPIO.output(in4, GPIO.LOW)
p1.start(12)
p2.start(15 )

def right():
GPIO.output(in1, GPIO.HIGH)
GPIO.output(in2, GPIO.LOW)
GPIO.output(in3, GPIO.LOW)
GPIO.output(in4, GPIO.LOW)
p1.start(15)
p2.start(12)

def stop():
GPIO.output(in1, GPIO.LOW)
GPIO.output(in2, GPIO.LOW)
GPIO.output(in3, GPIO.LOW)
GPIO.output(in4, GPIO.LOW)

stop()

try:
while True:
ret, frame = cap.read()
hsvFrame = cv2.cvtColor(frame, cv2.COLOR_BGR2HSV)

# Set range for red color and
# define mask
red_lower = np.array([57,69,68], np.uint8)
red_upper = np.array([108, 255,255], np.uint8)
red_mask = cv2.inRange(hsvFrame, red_lower, red_upper)

# Set range for green color and
# define mask
green_lower = np.array([46, 127, 79], np.uint8)
green_upper = np.array([92, 255, 255], np.uint8)
green_mask = cv2.inRange(hsvFrame, green_lower, green_upper)

# Set range for blue color and
# define mask
blue_lower = np.array([79, 99, 121], np.uint8)
blue_upper = np.array([123, 255, 255], np.uint8)
blue_mask = cv2.inRange(hsvFrame, blue_lower, blue_upper)

kernel = np.ones((5, 5), "uint8")

# For red color
red_mask = cv2.dilate(red_mask, kernel)
res_red = cv2.bitwise_and(frame, frame,
mask = red_mask)

# For green color
green_mask = cv2.dilate(green_mask, kernel)
res_green = cv2.bitwise_and(frame, frame,
mask = green_mask)

# For blue color
blue_mask = cv2.dilate(blue_mask, kernel)
res_blue = cv2.bitwise_and(frame, frame,
mask = blue_mask)

low_b = np.uint8([80,80,80])
high_b = np.uint8([0,0,0])
mask = cv2.inRange(frame, high_b, low_b)
contours, hierarchy = cv2.findContours(mask, 1, cv2.CHAIN_APPROX_NONE)
if len(contours) > 0 :
c = max(contours, key=cv2.contourArea)
M = cv2.moments(c)
if M["m00"] !=0 :
cx = int(M['m10']/M['m00'])
cy = int(M['m01']/M['m00'])
print("CX : "+str(cx)+"  CY : "+str(cy))
if cx >= 120 :
print("Turn Right")
right()

if cx < 120 and cx > 40 :
print("On Track!")
forward()
if cx <=40 :
print("Turn Left")
```

```
left()


cv2.circle(frame, (cx,cy), 5, (255,255,255), -1)
else :
print("I don't see the line")
stop()
time.sleep(1)

# Creating contour to track green color
contours, hierarchy = cv2.findContours(green_mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

for pic, contour in enumerate(contours):
area = cv2.contourArea(contour)
if(area > 300):
print("Green Detected")
p1.ChangeDutyCycle(10)
p2.ChangeDutyCycle(10)
left()
time.sleep(0.5)

# Creating contour to track blue color
contours,hierarchycv2.findContours(blue_mask,cv2.RETR_TREE,cv2.CHAIN_APPROX_SIMPLE)

for pic, contour in enumerate(contours):
area = cv2.contourArea(contour)
if(area > 300):
print("Blue Detected")
forward()
time.sleep(14)
stop()
#time.sleep(5)
GPIO.cleanup()
cap.release()
cv2.destroyAllWindows()



# Creating contour to track red color
contours, hierarchy = cv2.findContours(red_mask,
cv2.RETR_TREE,
cv2.CHAIN_APPROX_SIMPLE)

for pic, contour in enumerate(contours):
area = cv2.contourArea(contour)
if(area > 300):
print("Red Detected")
p1.ChangeDutyCycle(10)
```

```
p2.ChangeDutyCycle(10)
right()
time.sleep(0.7)

cv2.drawContours(frame, c, -1, (0,255,0), 1)
cv2.imshow("Mask",mask)
cv2.imshow("Frame",frame)
if cv2.waitKey(1) & 0xff == ord('q'):   # 1 is the time in ms
stop()
break
finally:
GPIO.cleanup()
cap.release()
cv2.destroyAllWindows()
```

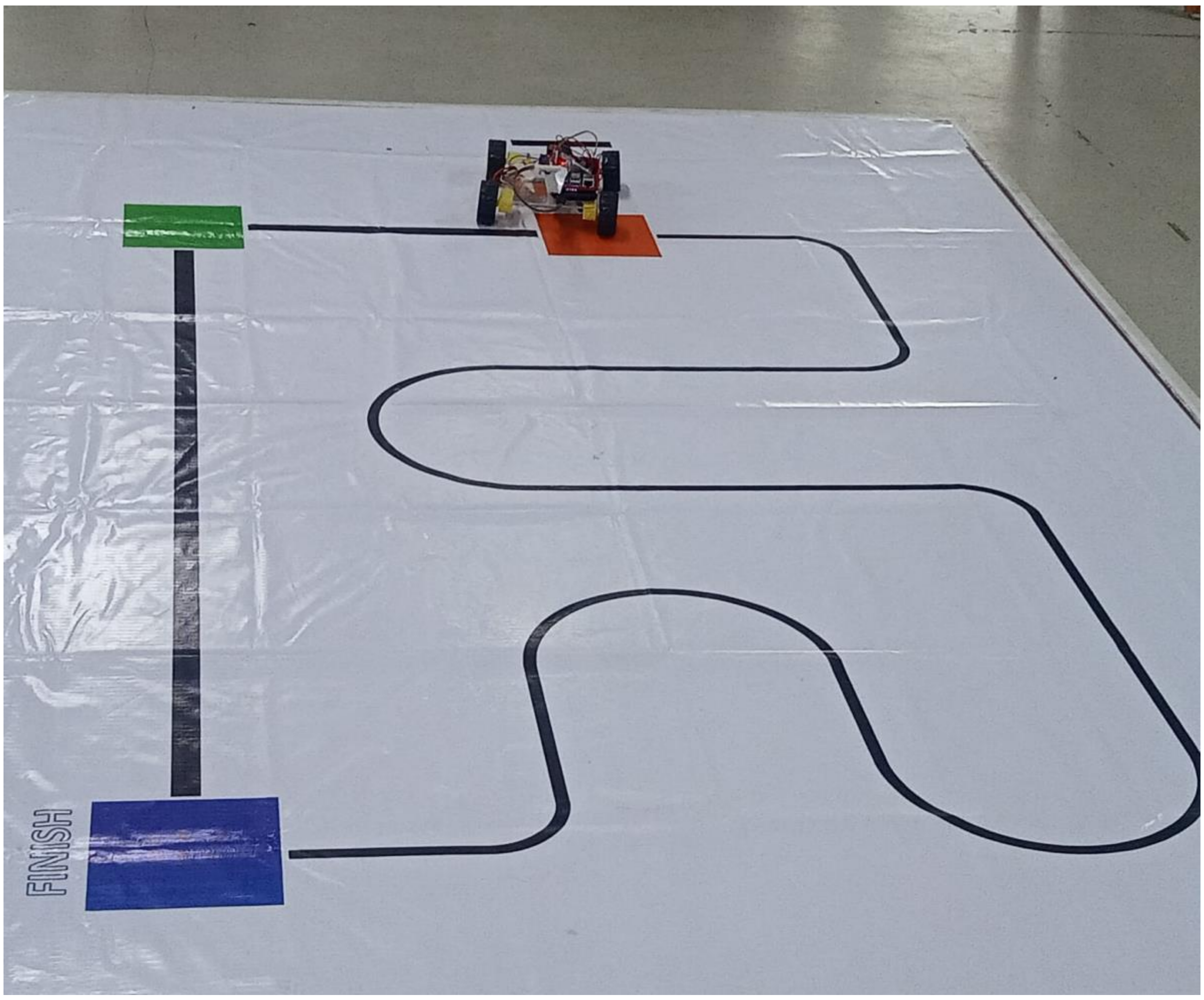


Fig 7.5: Arena of Raspberry Pi cameras to detect lines or take turns based on colours

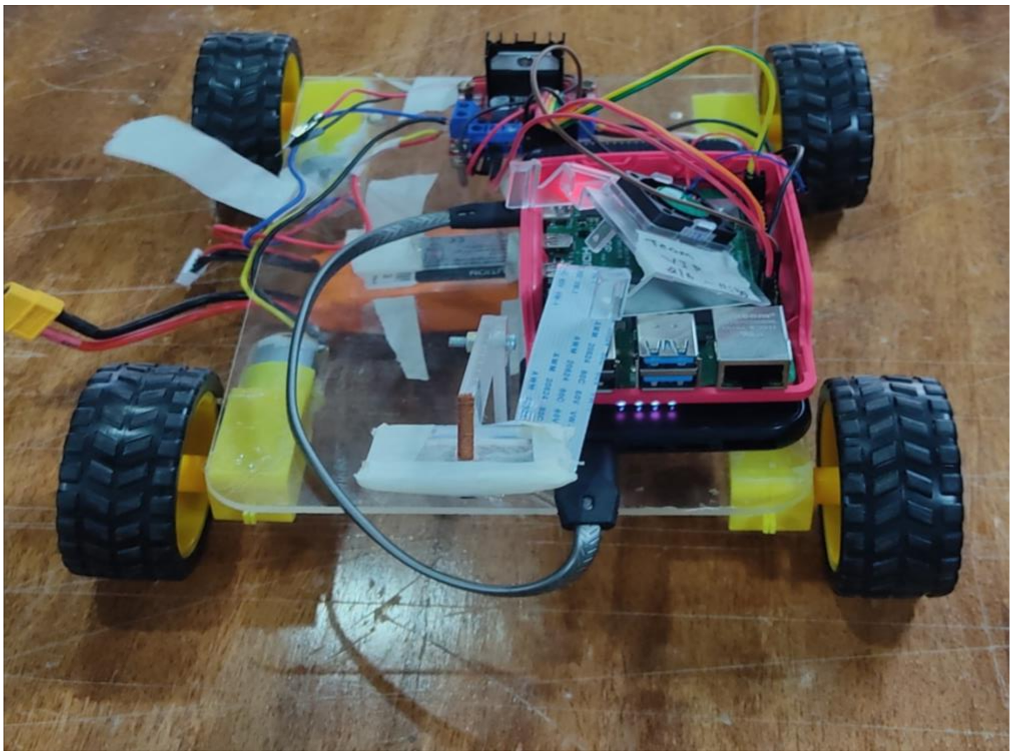


Fig7.6: Built Solution

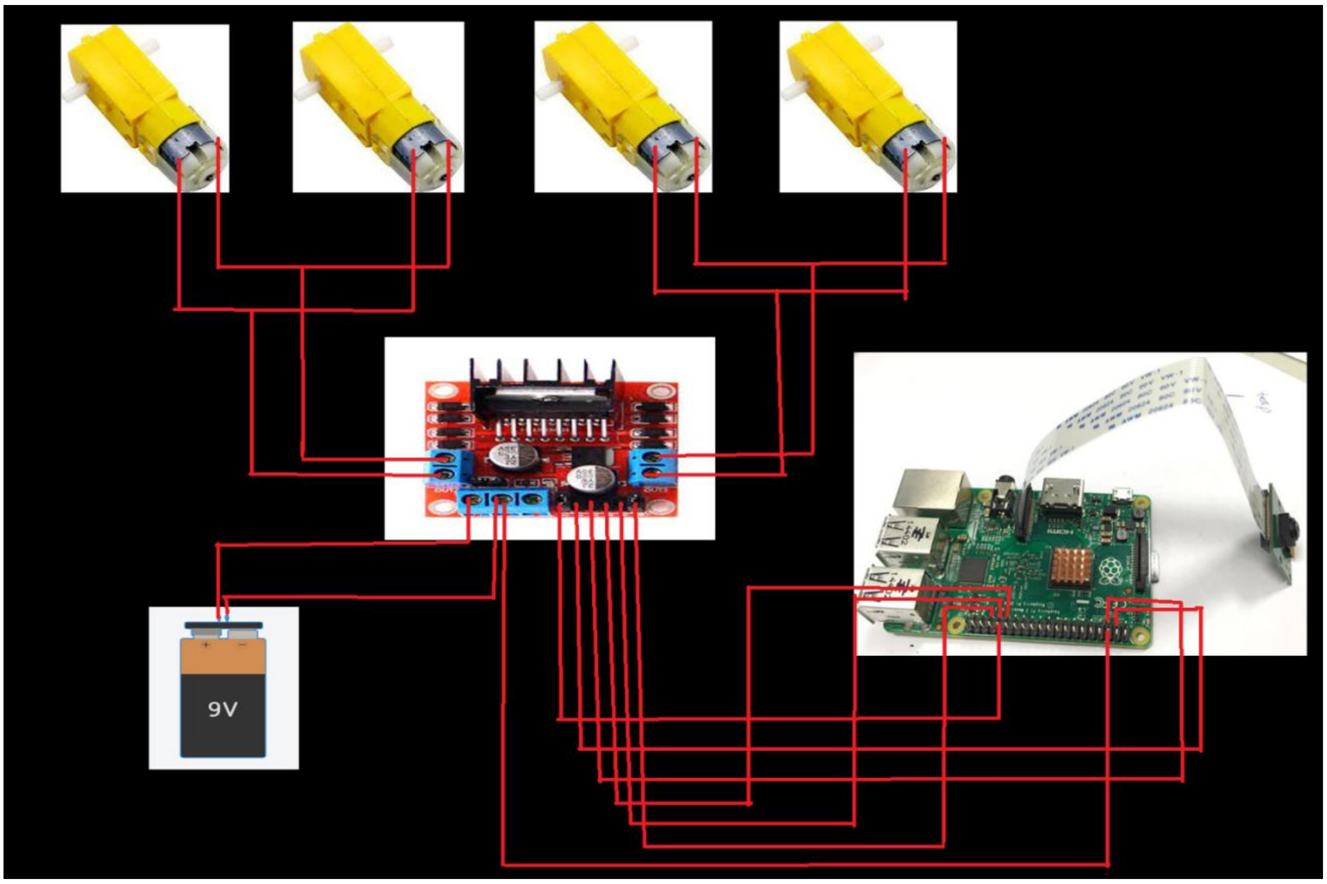


Fig 7.7: Circuit Diagram

8. Conclusion

The hybrid robotics course has been an invaluable learning experience for me as a student. Throughout the course, I have gained a comprehensive understanding of robotics concepts, along with practical hands-on experience in building and programming robotic systems.

The course curriculum covered a wide range of topics, including Arduino pinout, coding, servo motors, motor drivers, and Raspberry Pi interfacing. This allowed me to explore different aspects of robotics and develop a strong foundation in both hardware and software components.

The hybrid learning approach, combining online resources with hands-on practical sessions, has been particularly beneficial. It provided me with the flexibility to learn at my own pace while also engaging in practical applications and experiments. This hands-on experience has solidified my understanding of the theoretical concepts and enhanced my problem-solving skills.

I am particularly grateful for the opportunity to work on real-world projects, such as obstacle detection, parking assistance, and remote control operation. These projects challenged me to apply my knowledge and skills to solve practical problems, and they have given me a glimpse into the exciting possibilities of robotics.

Overall, the hybrid robotics course has equipped me with the necessary knowledge, skills, and confidence to pursue a career in robotics or related fields. I am excited about the prospects of applying what I have learned to contribute to advancements in robotics technology and make a positive impact in the world.

9. Facilitators

Facilitator Name: Mohammad Aftab Shariff
Designation: Facilitator
ComedKares Innovation Hub – JP Nagar

Facilitator Name: Vinu Chandran Y
Designation: Facilitator
ComedKares Innovation Hub – JP Nagar