ESW PROJECT

Obstacle Avoidance

Ground Robot with Autonomous Obstacle & Edge Avoidance , Bluetooth Remote Control & Radar Unit

Contents

TimeLine

Planning

Ideation and Hardware collection

System Design

Implementation of Sensors and Motor Driver

Implementation

Integration of Sensors and Motor Driver , Implementation of Radar Unit

Testing

Adjust Threshold and Delay Values after Analysis and Handling Ambiguous Errors

UI & Bluetooth

Implemented Bluetooth Remote Control and Mounting the Radar Unit

Functionalities

RADAR UNIT

- Ultrasonic Sensor and Servo Motor Map the Area in front of the Robot
- Sweeps an angle of 165 Degrees which Detects to check Obstacle within threshold
- Processing.io

EDGE AVOIDANCE

- Infrared Sensors
- Mounted on Extended Part of Robot body at Front
- Reroutes if there is an edge with depth > threshold

OBSTACLE AVOIDANCE

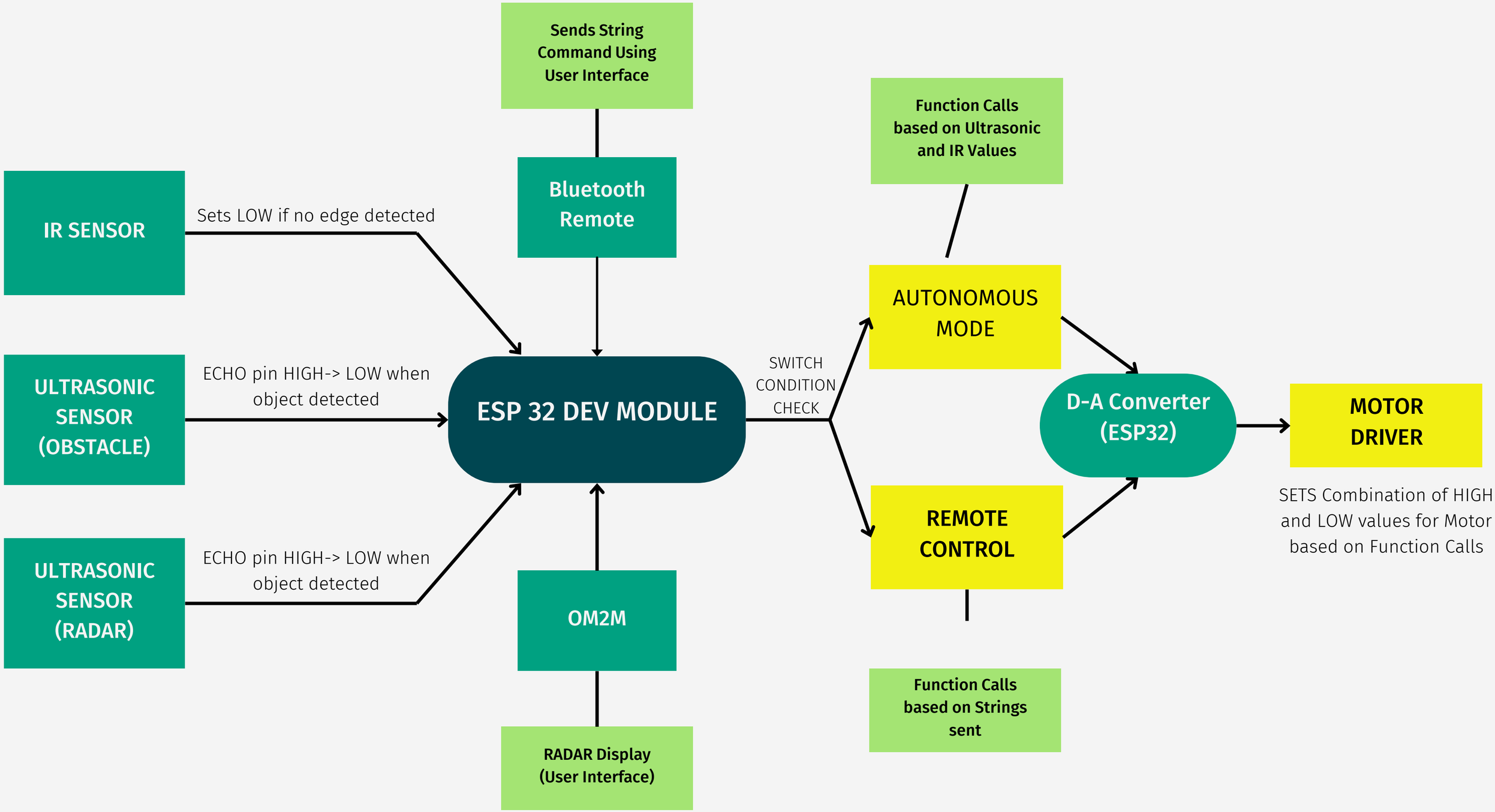
- Ultrasonic Sensors in Front
- Overrides control over the motors and pulls the robot back in case of an obstacle
- 3 Sensors to avoid Blindspots and Direction detection

BLUETOOTH REMOTE

- Switch modes between Obstacle Avoidance and Remote Control
- User Friendly Interface
- Incurs a small delay for the robot to respond

Components

- ESP 32 (x2)
- Chasis with Tyres (x4) and Motors (x2)
- Motor Driver (x1)
- HC-SR04 Ultrasonic Sensor (x4)
- IR Sensor (x2)
- Servo Motor (x1)
- Resistances & Jump Wires
- LEDs & Breadboards
- Miscellaneous Hardware & Testers


Sends String Command Using User Interface
Bluetooth Remote
IR SENSOR
Sets LOW if no edge detected
ULTRASONIC SENSOR (OBSTACLE)
ECHO pin HIGH-> LOW when object detected
ULTRASONIC SENSOR (RADAR)
ECHO pin HIGH-> LOW when object detected
ESP 32 DEV MODULE
OM2M
RADAR Display (User Interface)
SWITCH CONDITION CHECK
Function Calls based on Ultrasonic and IR Values
AUTONOMOUS MODE
REMOTE CONTROL
Function Calls based on Strings sent
D-A Converter (ESP32)
MOTOR DRIVER
SETS Combination of HIGH and LOW values for Motor based on Function Calls

Calibration

IR Sensor

- Direction allignment of the IR sensor
- Adjusting the IR threshold by physically rotating the screw to get the value as Height of the car
- Adjusting the screws so as to integrate both IRs

Ultrasonic Sensor

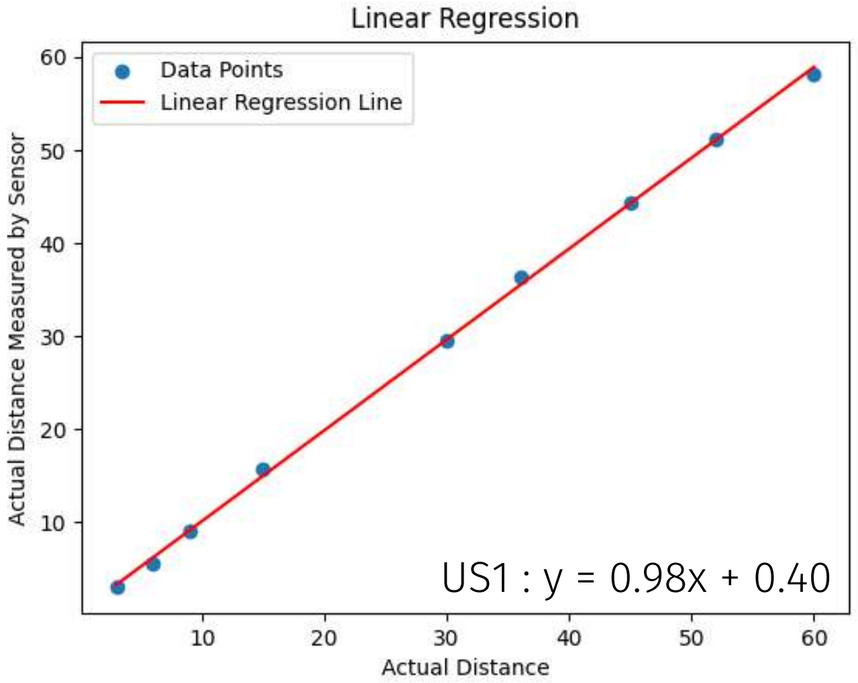
- Reading the datasheet provided by manufacturer
- Implementing a Linear regression code
- Getting the sensor values at predetermined distances
- Plotting the curve and getting the equation

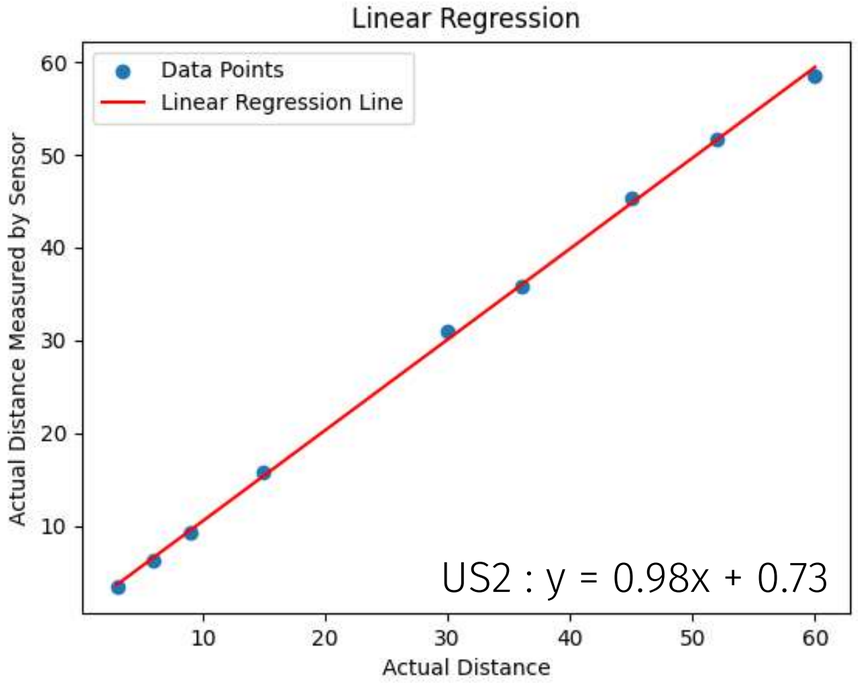
Motor Driver

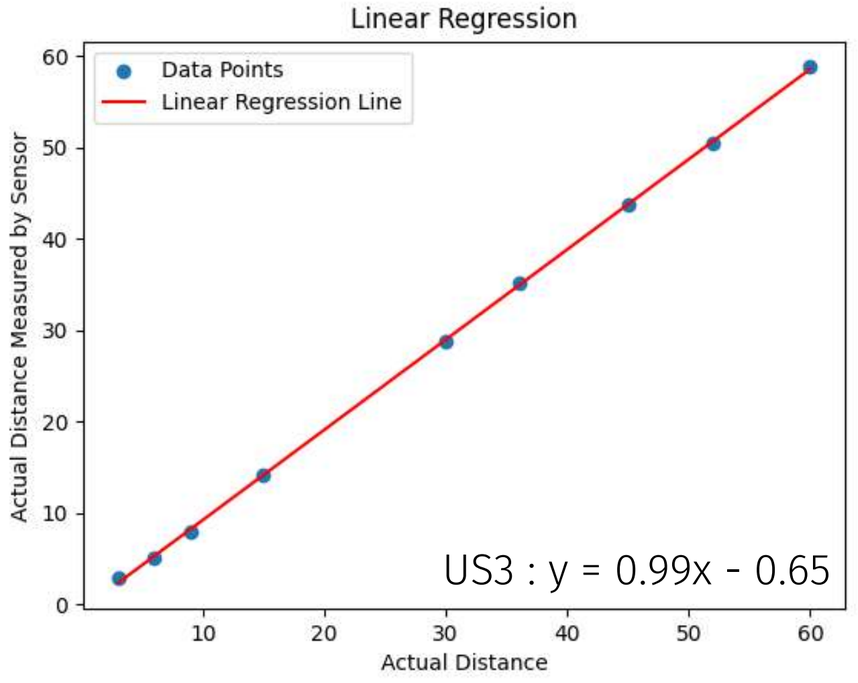
- Adjusting the voltage using the voltage regulator
- Passing optimal current to the ESP using resistor

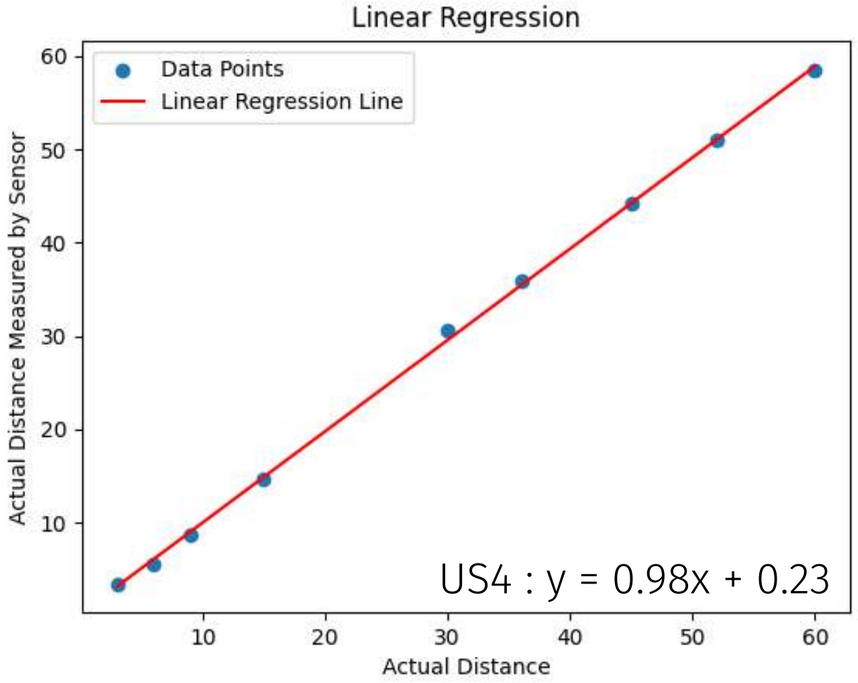
Servo Motor

- Adjusting the initial angle of the servo motor (Initial Offset)
- Mounting the servo at certain angle so as to handle the range of 15 to 165

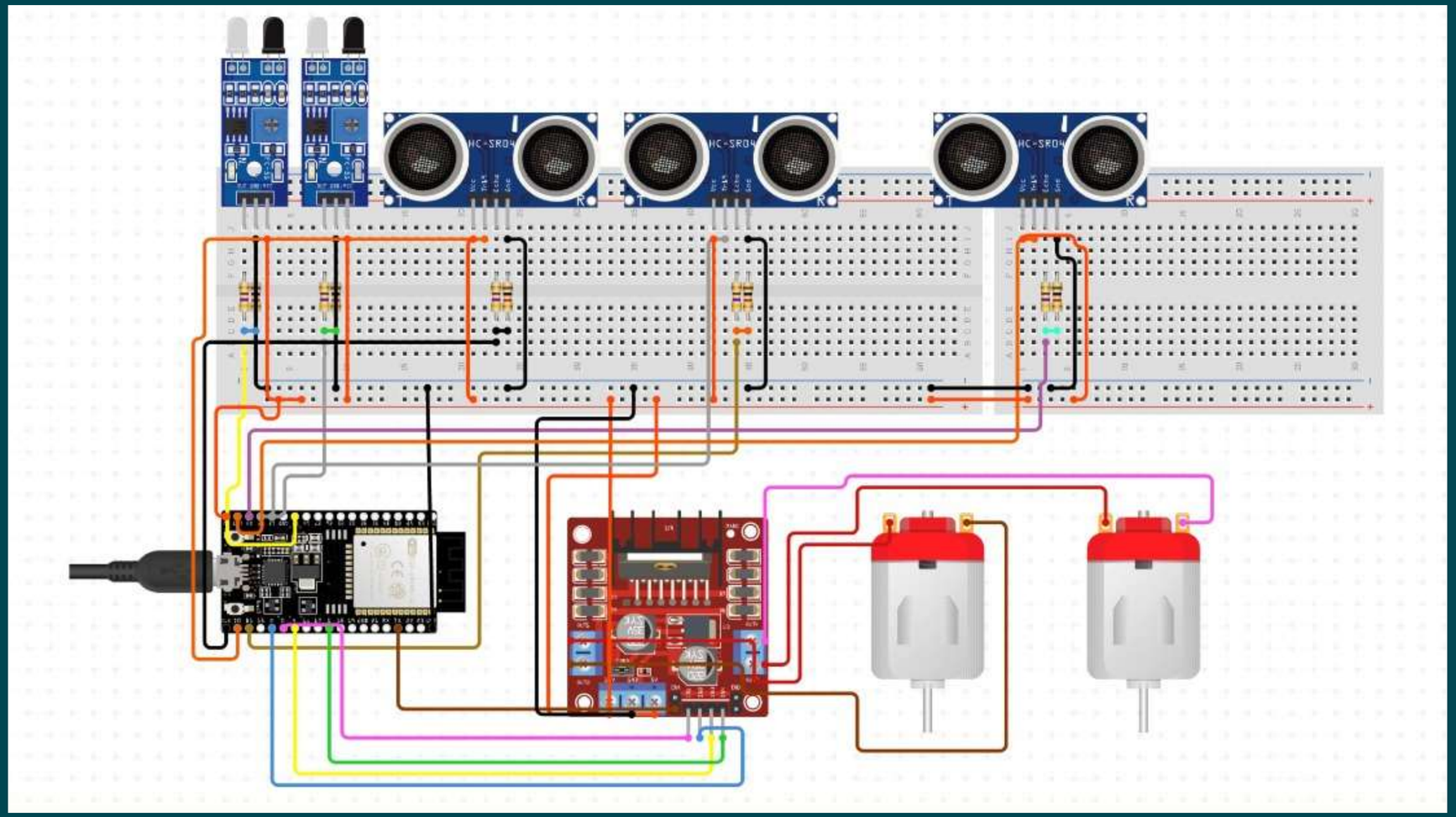

Linear Regression
Data Points
Linear Regression Line
Actual Distance Measured by Sensor
Actual Distance
US1 : y = 0.98x + 0.40


Linear Regression
Data Points
Linear Regression Line
Actual Distance Measured by Sensor
Actual Distance
US2 : y = 0.98x + 0.73

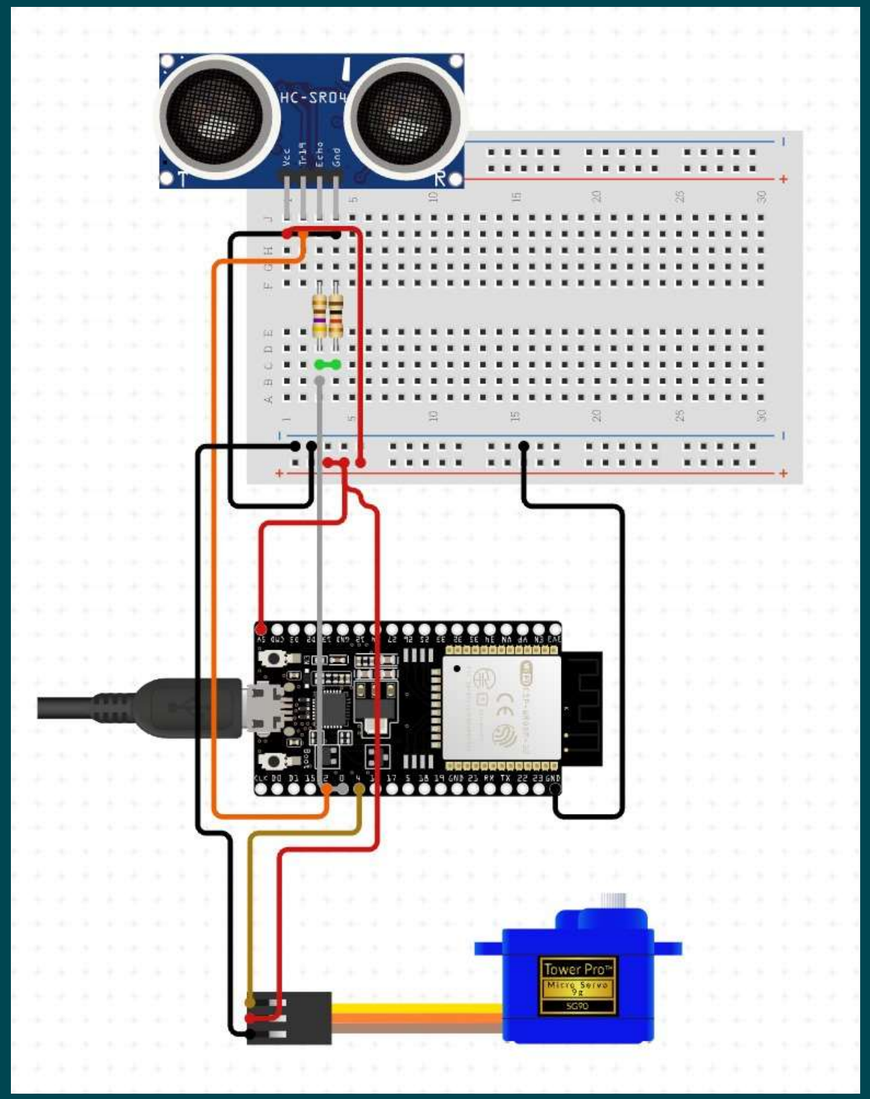

Linear Regression
Data Points
Linear Regression Line
Actual Distance Measured by Sensor
Actual Distance
US3 : y = 0.99x - 0.65

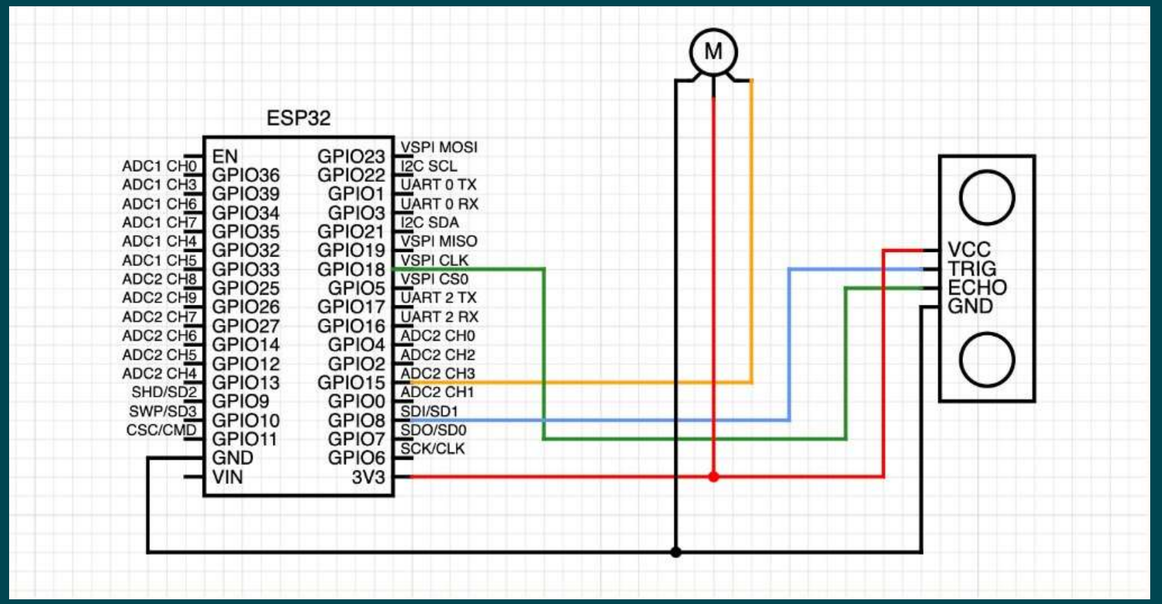

Linear Regression
Data Points
Linear Regression Line
Actual Distance Measured by Sensor
Actual Distance
US4 : y = 0.98x + 0.23

CIRCUIT



RADAR UNIT


HC-SR04
Vcc
Trig
Echo
Gnd
T
R
Tower Pro
Micro Servo
9g
SG90


M
ESP32
EN
GPIO36
GPIO39
GPIO34
GPIO35
GPIO32
GPIO33
GPIO25
GPIO26
GPIO27
GPIO14
GPIO12
GPIO13
GPIO9
GPIO10
GPIO11
GND
VIN
GPIO23
GPIO22
GPIO1
GPIO3
GPIO21
GPIO19
GPIO18
GPIO5
GPIO17
GPIO16
GPIO4
GPIO2
GPIO15
GPIO0
GPIO8
GPIO7
GPIO6
3V3
ADC1 CH0
ADC1 CH3
ADC1 CH6
ADC1 CH7
ADC1 CH4
ADC1 CH5
ADC2 CH8
ADC2 CH9
ADC2 CH7
ADC2 CH6
ADC2 CH5
ADC2 CH4
SHD/SD2
SWP/SD3
CSC/CMD
VSPI MOSI
I2C SCL
UART 0 TX
UART 0 RX
I2C SDA
VSPI MISO
VSPI CLK
VSPI CS0
UART 2 TX
UART 2 RX
ADC2 CH0
ADC2 CH2
ADC2 CH3
ADC2 CH1
SDI/SD1
SDO/SD0
SCK/CLK
VCC
TRIG
ECHO
GND

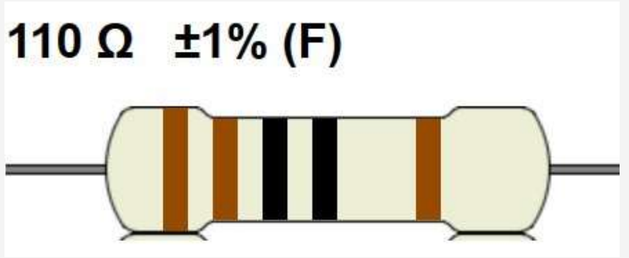
ERROR ANALYSIS

“Error is not a fault of our knowledge, but a mistake of our judgment giving assent to that which is not true”

- Excess Current to ESP
- Adjusting the Center of Mass
- Delay due to OM2M
- MD5 File Error
- Handling Infinite Loops

Excess Current to ESP

Sample resistor value (in lab) = 110 ohms



Calculation:

Required current through ESP32 = (0.02 , 0.04) Amps

Actual current flow through ESP32 = 0.2 Amps

By using the given relation:

$$V = I.R$$

$$\frac{1}{I_{total}} = \frac{1}{I_1} + \frac{1}{I_2}$$

Required Resistor value in range

108.5 - 111.5 ohms

$$R = \frac{V}{I_{resistor}} = \frac{V}{I_{req}} - \frac{V}{I_{obs}}$$

To divert the current from the ESP we connect a resistor in Parallel and allow a part of the current to flow into the ground

Error Analysis

MD5 FILE ERROR

- After esptool.py wrote the new binary to the flash it read back the contents and it didn't match. File corruption, or write protection prevented the flash from being updated.
- Also, while drawing current for ESP from Motor Driver , such an error occured as powering the ESP directly may lead to internal damages

DELAY DUE TO OM2M

- Radar Unit had an implementation using OM2M initially but due to realtime graph not being smooth (Due to the Latency of OM2M) , We dropped this initial plan

RUNNING INTO INFINITE LOOP

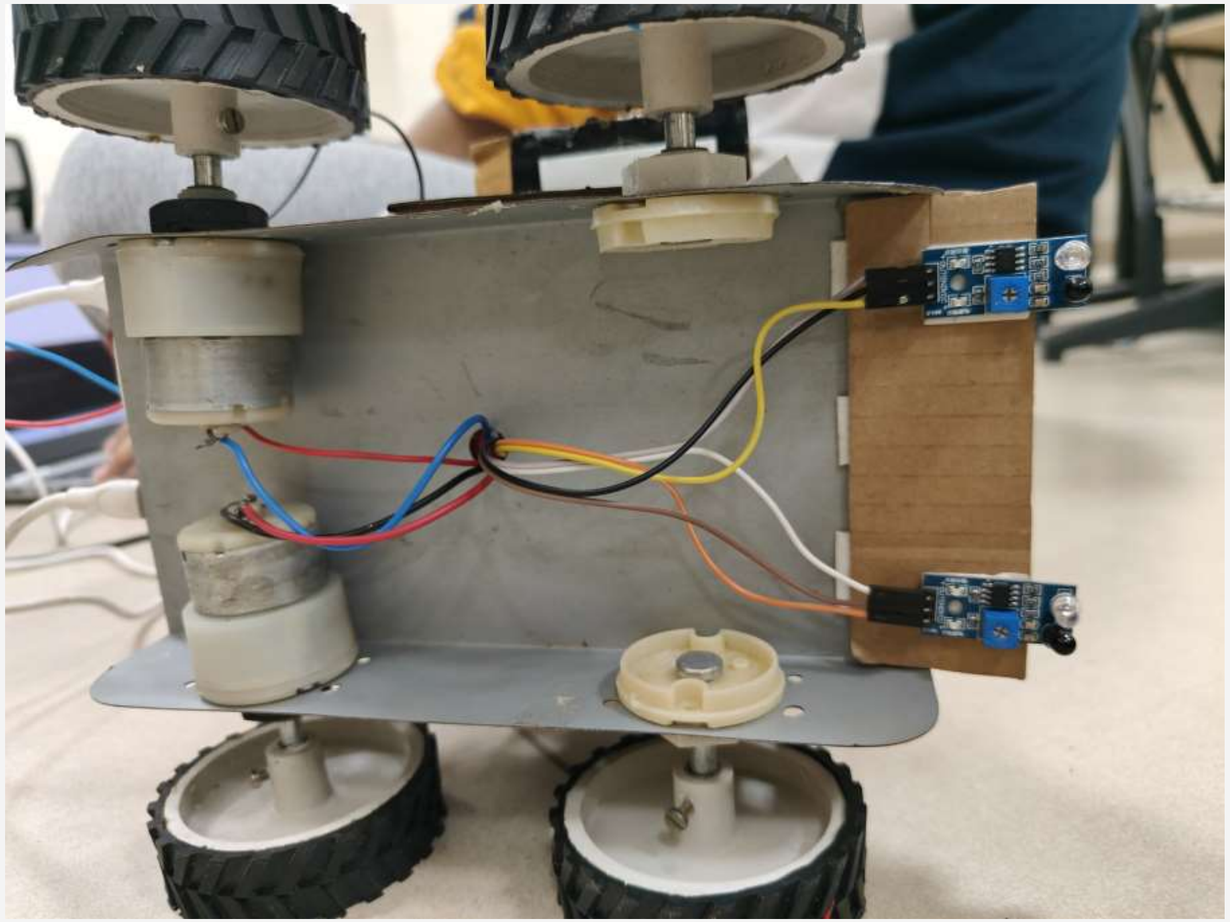
- The robot looks for the longest free path available between left and right and turns accordingly
- This leads to a possibility of infinite loop (recursively turning left and right by same amount)
- This was handled by making the Robot turn left and right for different angles (i.e. the delays for left and right is not same)

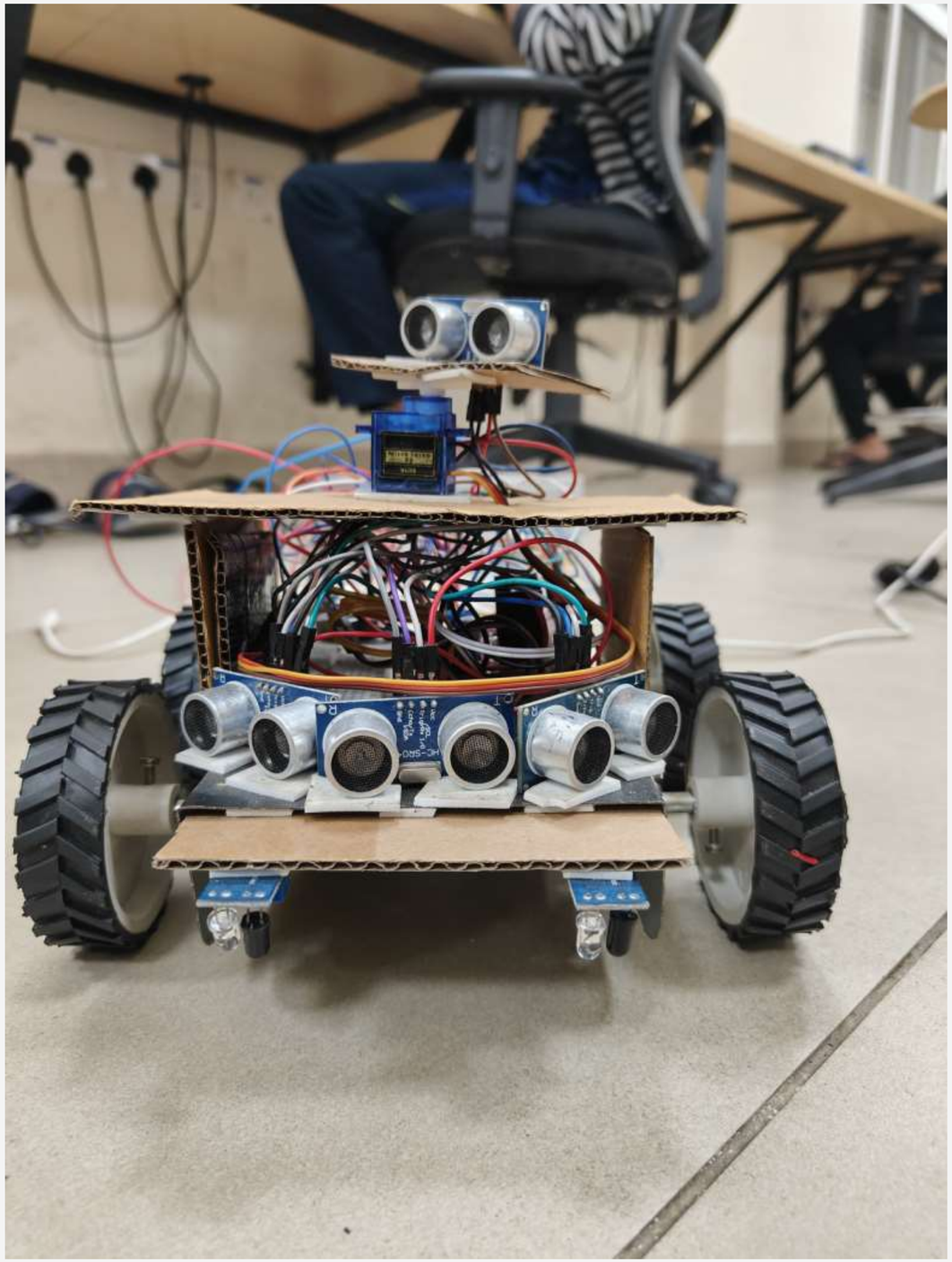
Adjusting COM

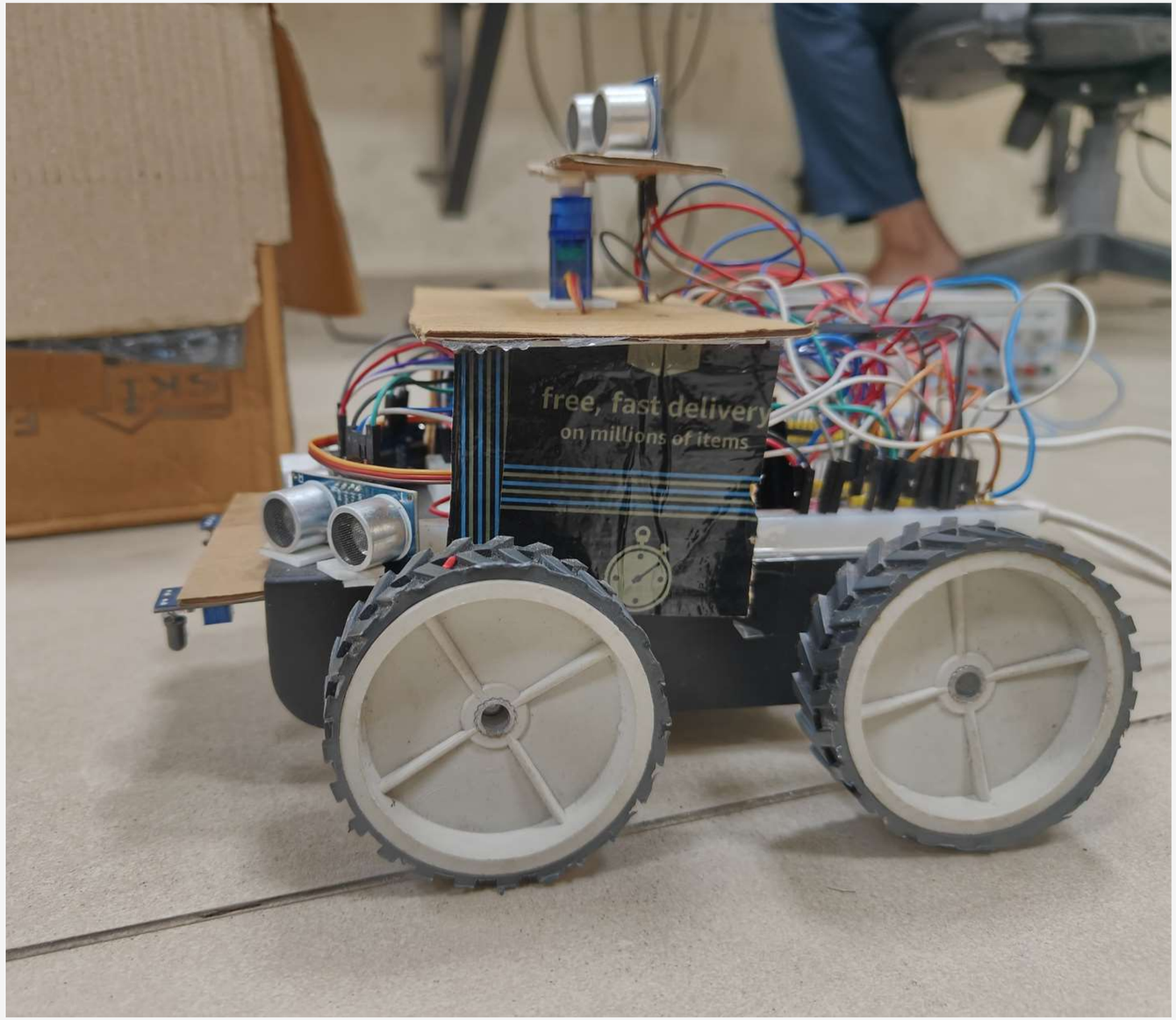
One side of the robot car was experiencing insufficient contact with the ground due to an uneven weight distribution, leading to instability during movement

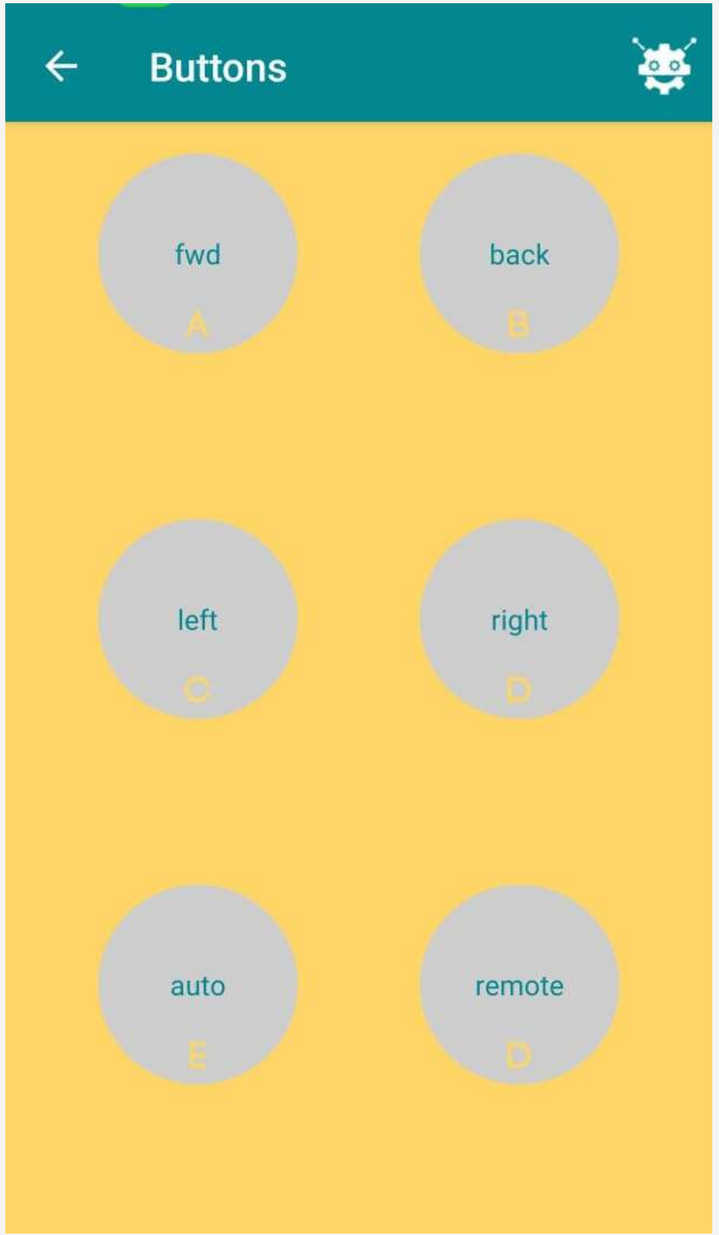
To rectify the instability, a small weight (ΔW) was added to the lighter side. The goal was to introduce a counter torque that would balance the torque causing the instability.

I*dentify Unstable Side*: Determine the side of the car where instability is observed.
Calculate Torque: Assess the torque (τ) causing the instability due to uneven weight distribution.
Determine Counter Torque: Calculate the counter torque (τ_counter) required to stabilize the car.
Add a small weight (ΔW) to the lighter side.
Testing: Assess the stability of the robot car after each weight addition.
Results:The addition of a small weight on the lighter side successfully improved the stability of the robot car. The weight was carefully selected to provide the necessary counter torque, ensuring better contact of all wheels with the ground






free, fast delivery
on millions of items


Buttons
fwd
A
back
B
left
C
right
D
auto
E
remote
D

Code Base

https://github.com/divyanash911/ESW_TeamProject

Owner of repository - divyanash911

Contributors - hemang-n00b , MohakSomani , AanvikBh

References

https://www.researchgate.net/publication/224173055_A_novel_potential_field_method_for_obstacle_avoidance_and_path_planning_of_mobile_robot
https://www.researchgate.net/publication/313389747_Potential_field_methods_and_their_inherent_approaches_for_path_planning
https://journals.sagepub.com/doi/pdf/10.1177/1729881418799562
https://www.cs.cmu.edu/~motionplanning/papers/sbp_papers/integrated1/borenstein_potential_field_limitations.pdf
https://app.diagrams.net/#G1txN8pTdBOoUyEYBJykSd_4Jii9mSY0FL

Thank You