# PROJECT REPORT

## AGV EMBEDDED TASK II

**Prepared By:** Soham Agarwal

**Roll No : 24CH10039**

**Dept of Chemical Engineering , IIT KGP**

**Project Name: Controls + Robot Building**

**Team : Embedded**

**Description:** The task primarily consists of building a robust and effective control system, with a focus on implementing and tuning Proportional-Integral-Derivative (PID) control. The goal is to develop an AGV capable of smoothly and accurately following a hand, relying on ultrasonic sensors for distance measurement and PID algorithms for precise motor control. For this not only understanding the theoretical foundations of PID control but also the practical challenges of applying it to a real-world robotic system, including sensor noise, motor response characteristics, and the need for careful parameter tuning to achieve optimal performance. The project further requires a comparative analysis of different control strategies (PID, PI, PD) to identify the most suitable approach for this specific application, considering factors such as response time, stability, and steady-state error.

## BRIEF INTRODUCTION

WHAT IS A CONTROLLER?

For simpler robotic applications a controller rather is not required -

- When the error in implementation is negligible
- When Speed of the machinery is not controlled and always constant
- When the command is direct and always possible

While for a much efficient and advanced applications -

- Where naturally occurring errors are not negligible.
- Speed of the machine to be controlled , so as to reduce its error.
- Command is step by step and prevents any error caused.

**COMPONENTS USED :**

| COMPONENT | UNITS |
|---|---|
| ARDUINO UNO | 1 |
| ULTRASONIC SENSOR | 2 |
| WHEELS & MOTORS | 2 |
| MOTOR DRIVER | 1 |
| CASTOR WHEEL | 2 |
| BREADBOARD | 1 |

**Ultrasonic sensor working :**

- **Trigger Pin :** A trigger pin is a digital output used to initiate an action on a sensor or module. For instance, in ultrasonic sensors, it sends a brief high-level pulse that starts the distance measurement process by triggering the emission of ultrasonic waves.
- **Echo Pin :** The echo pin serves as the input channel for receiving the ultrasonic signal that bounces back from an object. Its pulse duration is directly proportional to the distance measured, allowing the sensor to calculate how far away the object is.

**Motor Driver Working:**

| PINS | NUMBER | |
|---|---|---|
| VCC | 1 | 5V- 12V |
| 5V | 3 | Converted directly through 5A regulator |
| ENABLE PINS | 2 | Controls the speed of the motors using analogWrite command on Arduino.<br>(Make sure to use ~PWM pins for the same) |
| INPUT PINS | 4 | N1,N2,N3,N4 takes input from the arduino as a pwm signal digital output . and converting it to required voltage for the motors. |
| OUTPUT PINS | 4 | O1,O2,O3,O4 pins give output from the where each two of them are used. |

**ARDUINO UNO :**

~PWM PIN : distinguishes such pwm pins which can give analog output (3,5,6,9)

**PWM pin :**

| | |
|---|---|
| DIGITAL PWM PIN ~9 | Enable pin B |
| DIGITAL PWM PIN ~6 | Enable Pin A |
| DIGITAL PWM 2 | motor driver n4 |
| DIGITAL PWM 3 | motor driver n3 |
| DIGITAL PWM 4 | motor driver n2 |
| DIGITAL PWM 5 | motor driver n1 |
| DIGITAL PWM 10 | echo pin right |
| DIGITAL PWM 11 | Trigger pin right |
| DIGITAL PWM 12 | echo pin left |
| DIGITAL PWM 13 | Trigger pin left |

# IMPLEMENTED LOGIC

The PID Output helps in avoiding the errors of the circuit . Analyzes the past , present and future of the output and creates a symphony in total to reduce unconditional errors .

The implemented code constrains the robot to be present within the distance set in the setpoint from the hand (obstacle) . I have also introduced a deadband in the same for smoothing out the errors caused. (noting that the error is still a continuous function)

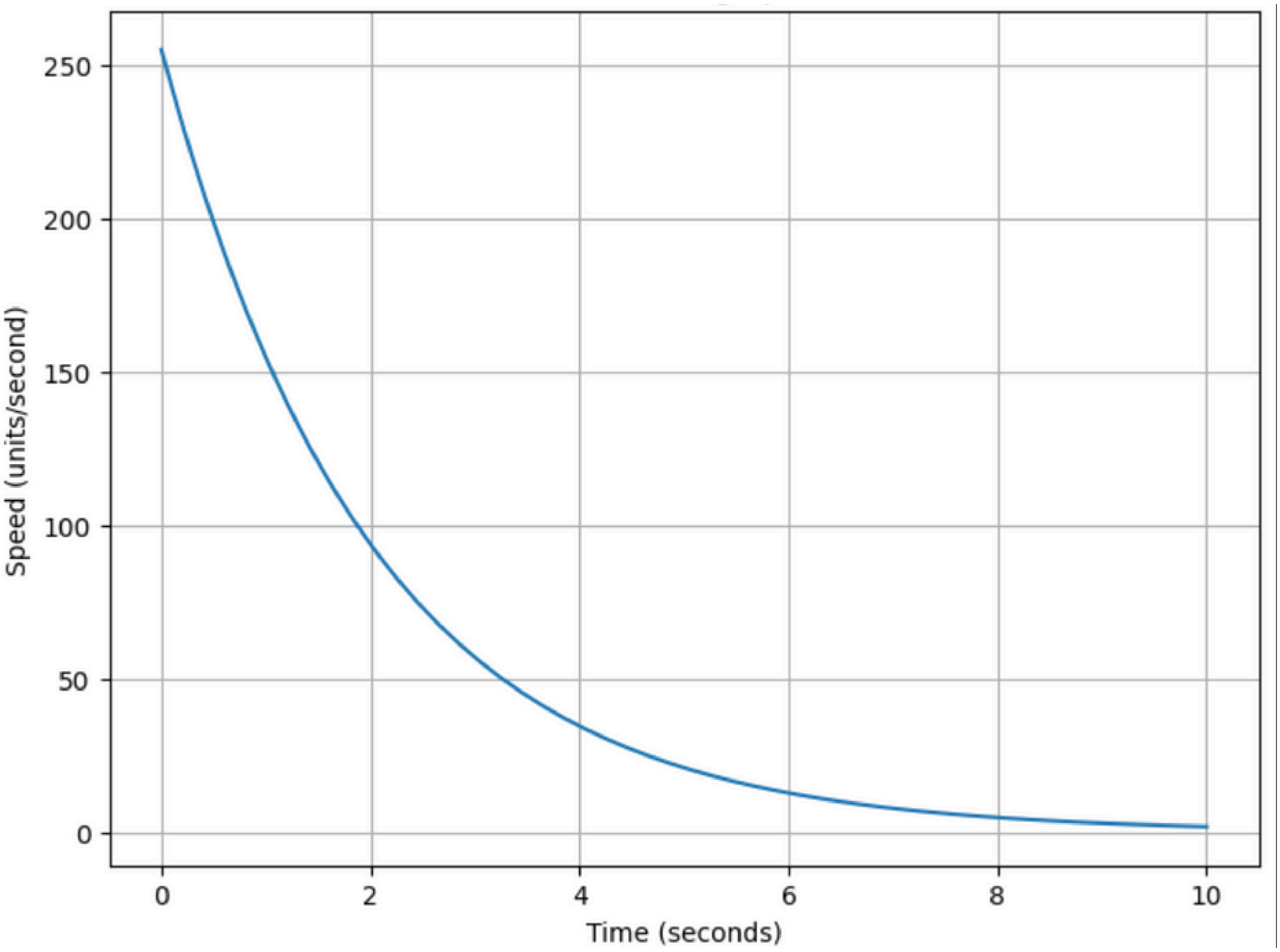
Each motor is separately controlled by each sensor individually , such that if the input distance from left < setpoint ,error is (-ve) , left motor moves forward .Similarly for the right combination.

### HOW PID CONTROL HELPS?

PID output is evaluated from the error calculated . which combines the effect of the past present and future of the outputs.

But what does all this mean? Why do we need to use the past present and future?

If the motor is in high speed , due to inertia the motor moves ahead and strikes the hand. So if the past of the output that is its error is seen . and the motor speed decrements also , while coming to the hand.



## PROBLEMS FACED

- Distance smoothing : The distance output from the sensor was rough .

i added an extra line to the code to smooth out the distance changing recieved by the ultrasonic sensor

```
distance_l = (distance_l + prev_distance_l)/2;
```

```
prev_distance_l = distance_l;
```

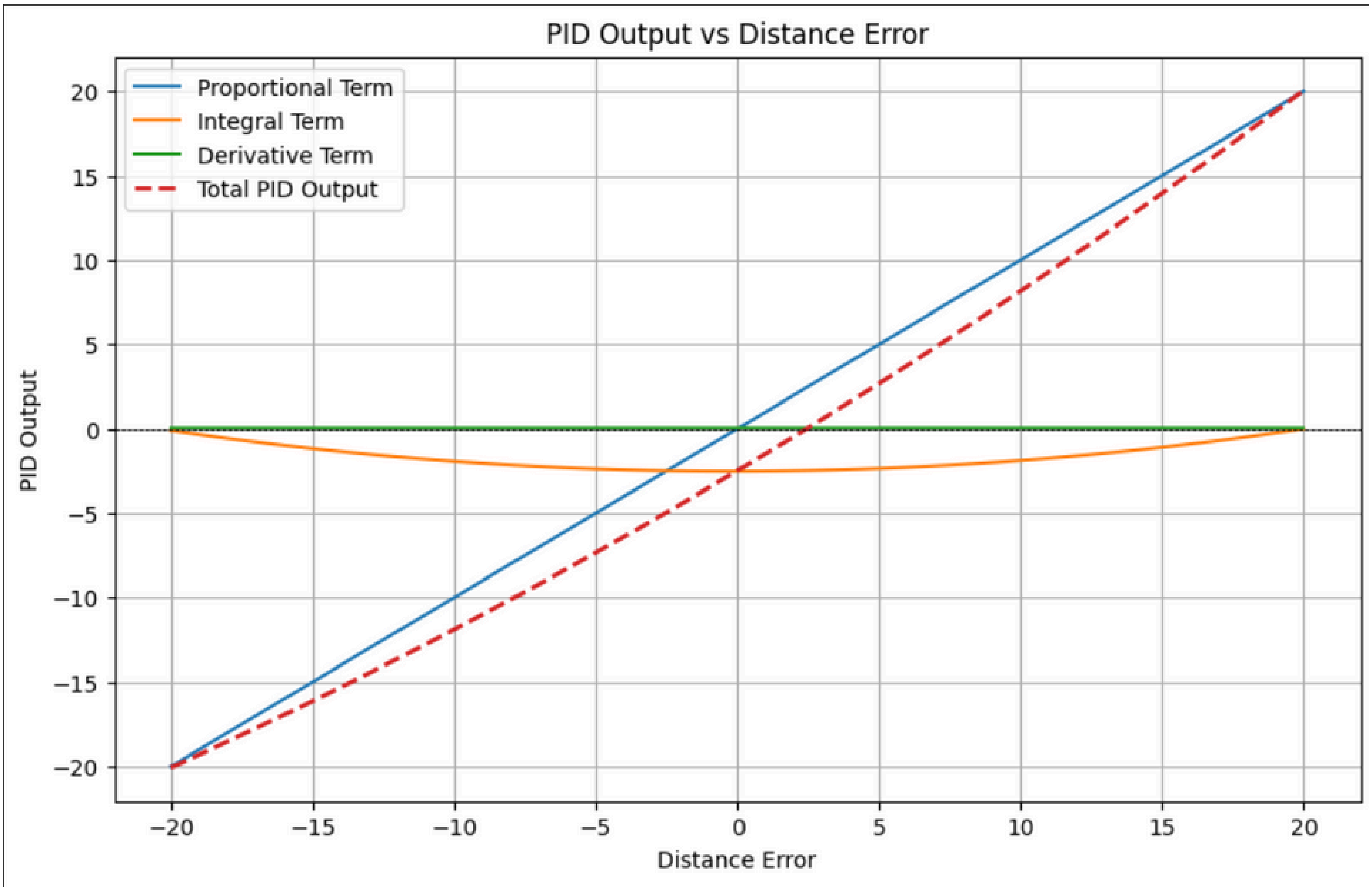
- Hardware implementation : Due to the power supply i have provided . to the motor driver and the arduino and the ultrasonic sensors , the voltage obtained at the output terminal of the motor is very less due to which the system does not move with only battery provided.

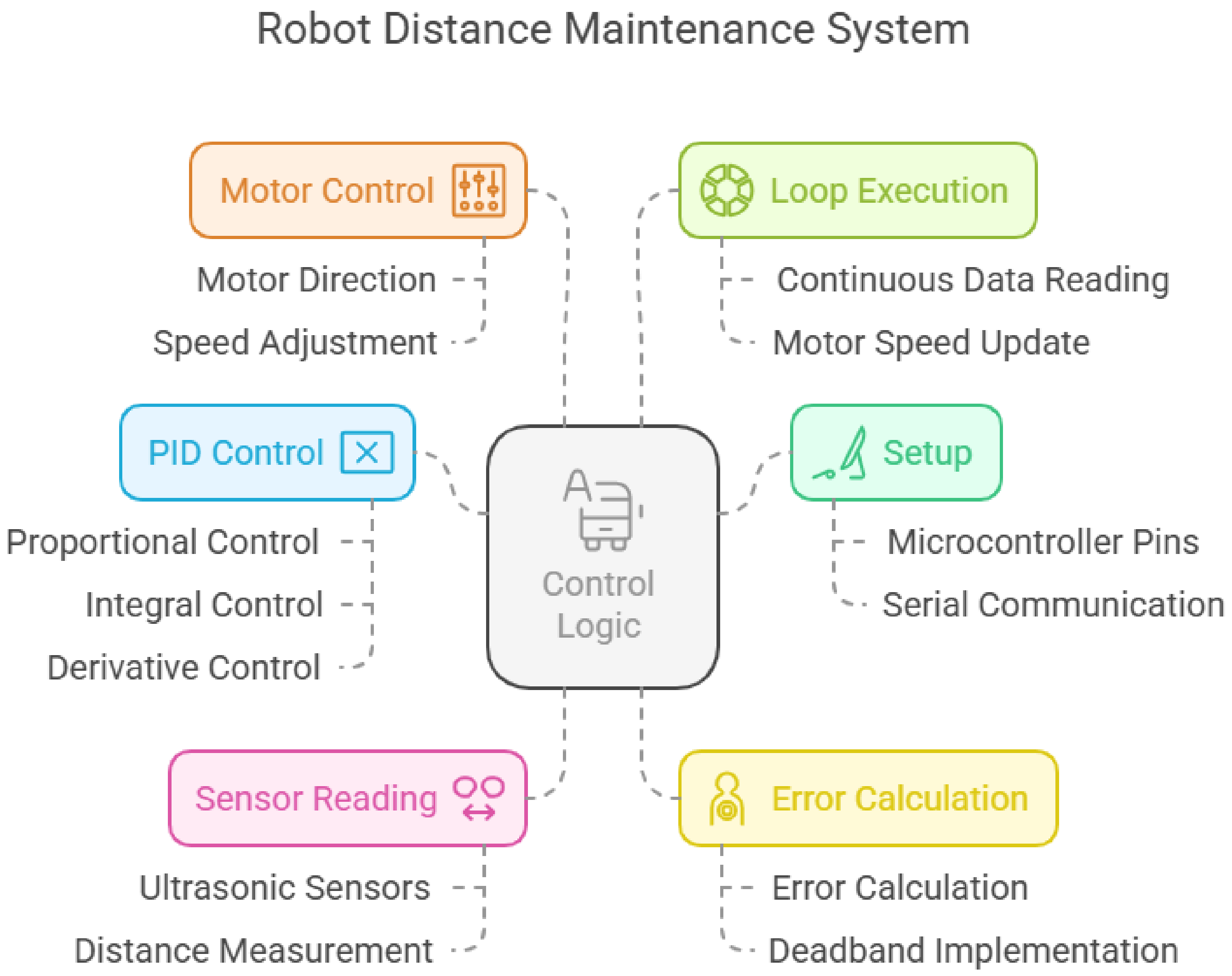
So i m still trouble shooting this problem at this point . for the lack of proper power supply . the motors are not running upto the requirement

- I HAD MOTOR DRIVER L298N ALREADY , SO INSTEAD OF BUYING THE L293D IC I USED MINE ONLY . BUT THE TINKER CAD SHOWS L293D.

# VISUAL INTERPRETATION

**PID output change with the error.**





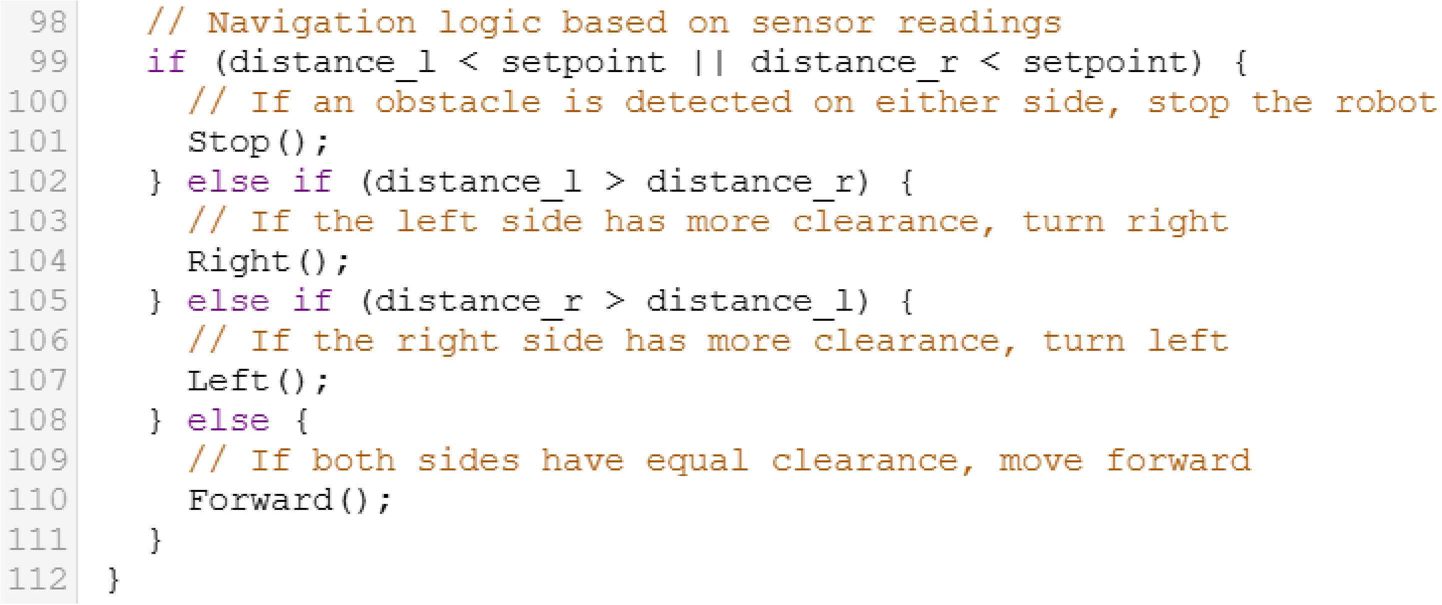
https://www.tinkercad.com/things/4YW6pinocQ5-open-loop-hand-following

# OPEN LOOP CIRCUIT

**You can Refer to GitHub link for the full code**

**(kP = 1 | kI = 0 | kD = 0) also works as an open loop circuit**

**but for the first reference and creation , i tried to implement the simplest circuit possible**



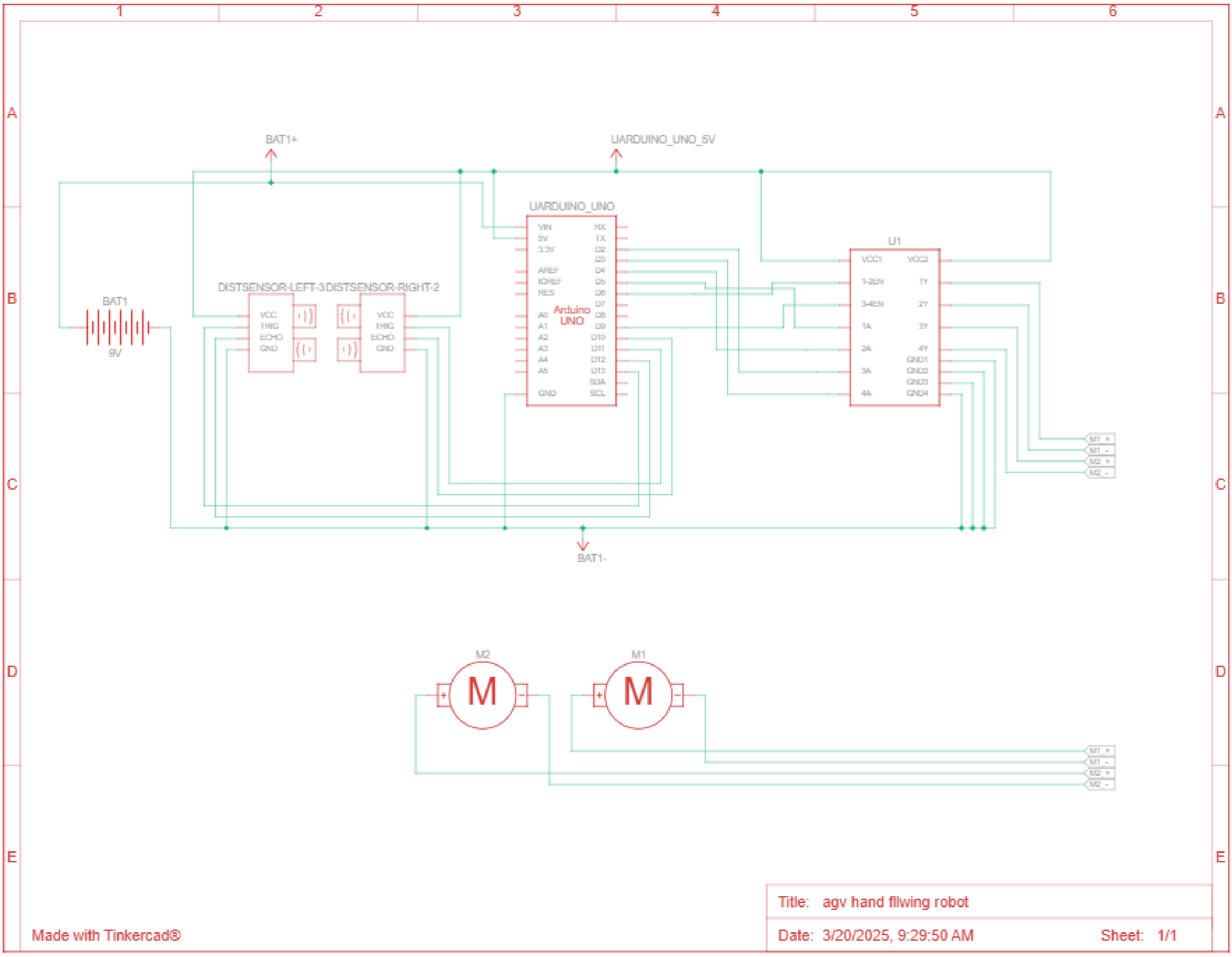
```
  // Navigation logic based on sensor readings
  if (distance_l < setpoint || distance_r < setpoint) {
    // If an obstacle is detected on either side, stop the robot
    Stop();
  } else if (distance_l > distance_r) {
    // If the left side has more clearance, turn right
    Right();
  } else if (distance_r > distance_l) {
    // If the right side has more clearance, turn left
    Left();
  } else {
    // If both sides have equal clearance, move forward
    Forward();
  }
}
```

# PID Control CIRCUIT

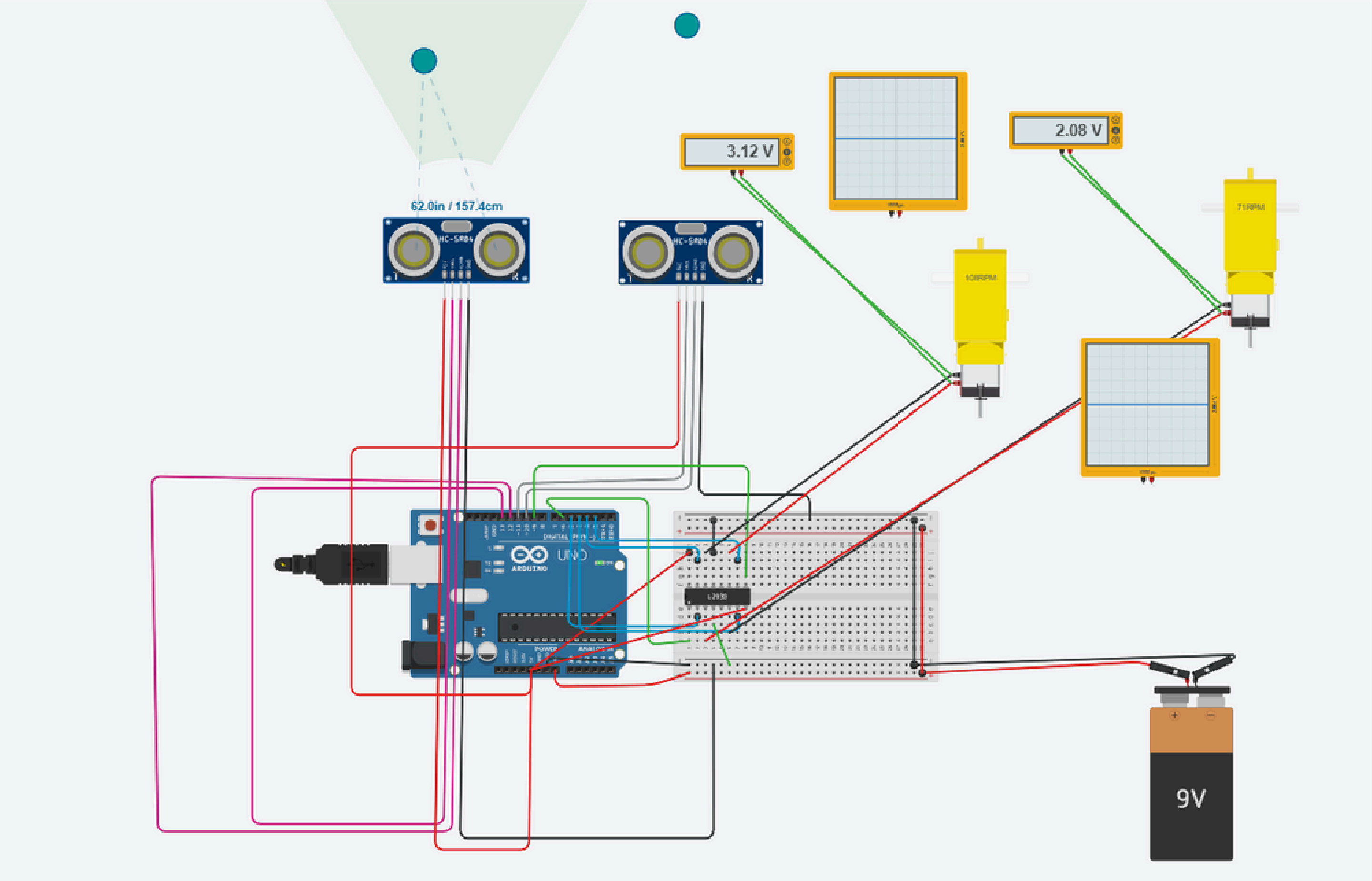
**You can Refer to GitHub link for the full code**

**(kP = 1 | kI = opt(kd) | kD = opt(kd))**

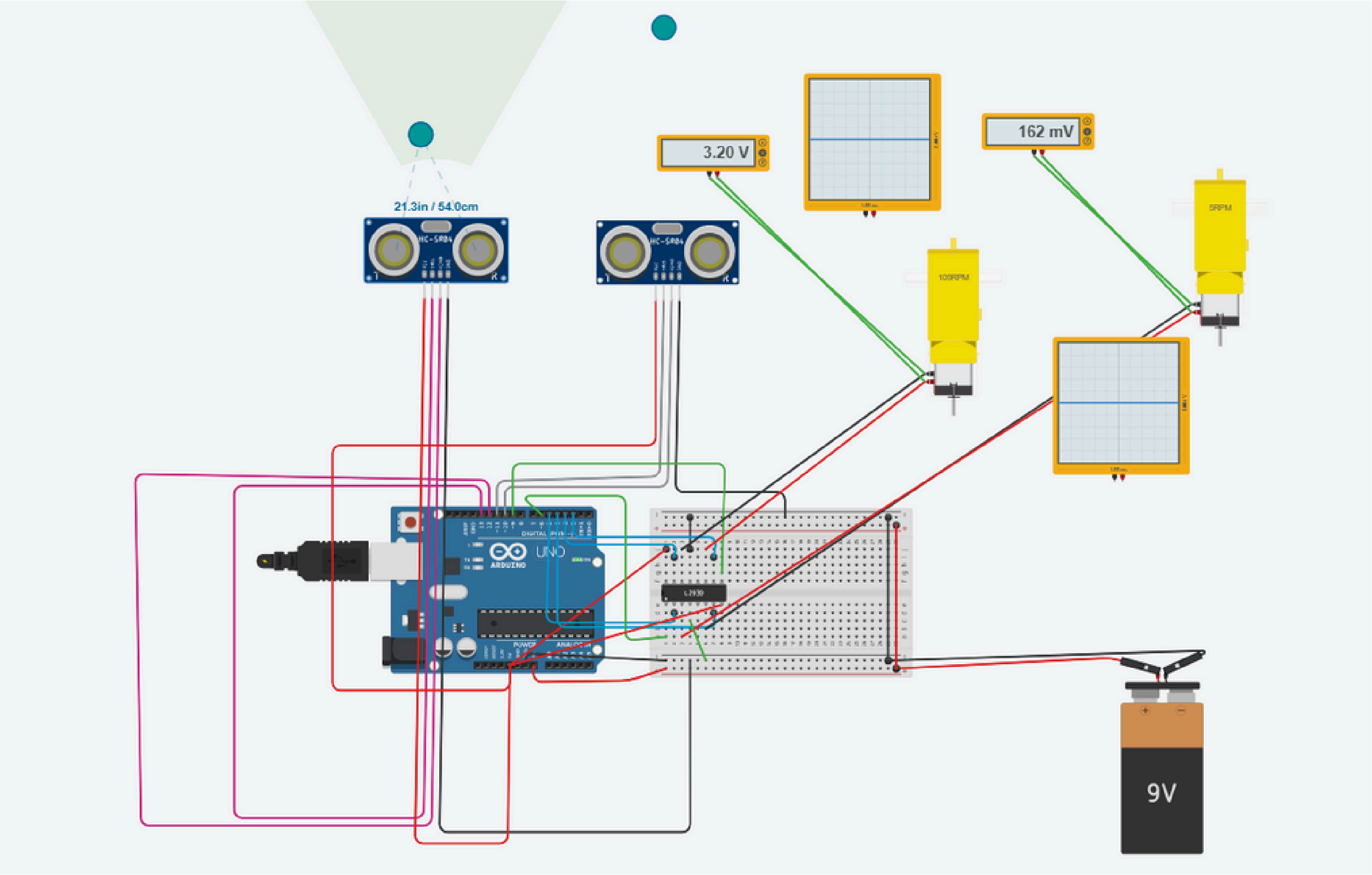
**Circuitry**



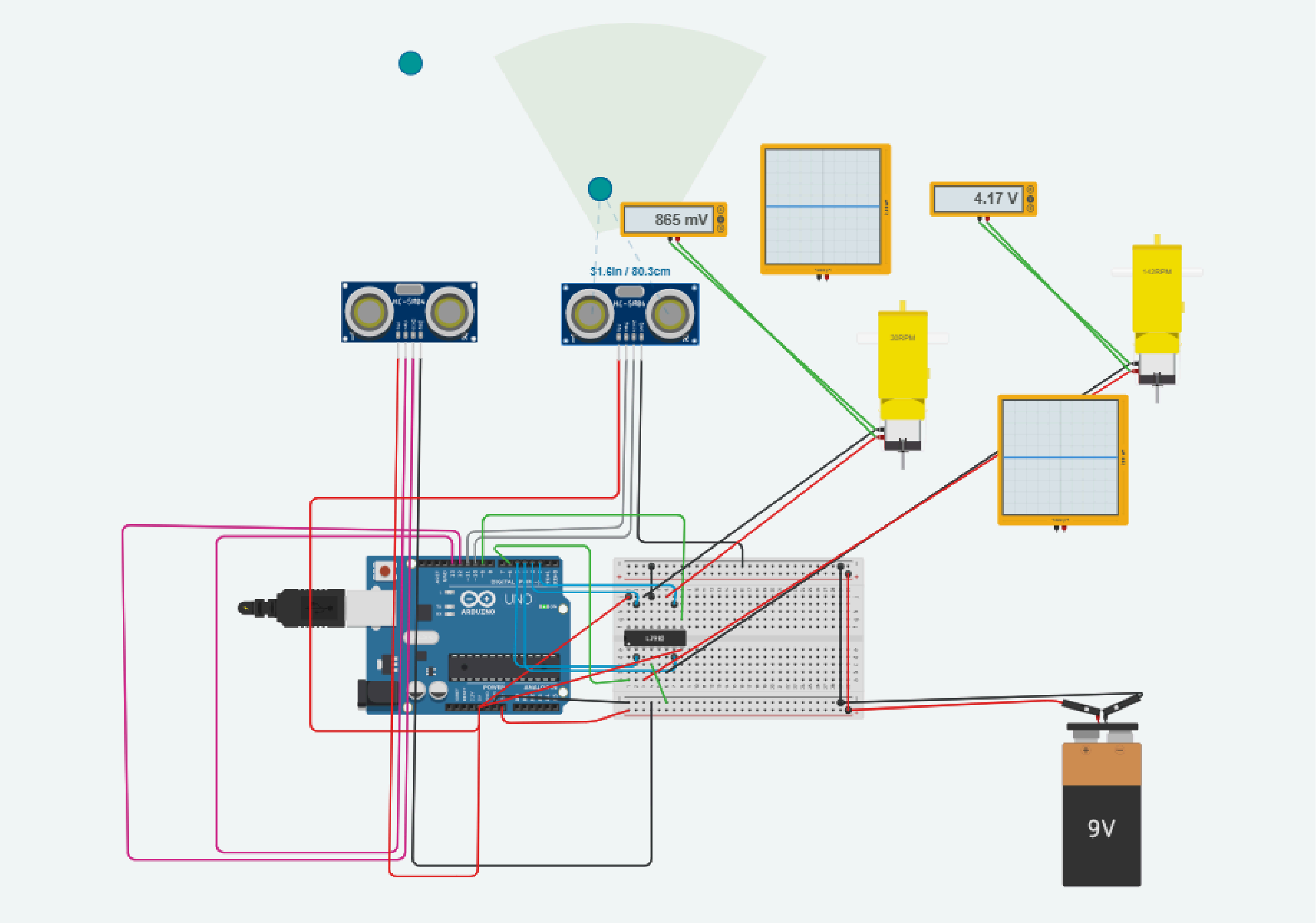
**If Right Distance > Left Distance,**



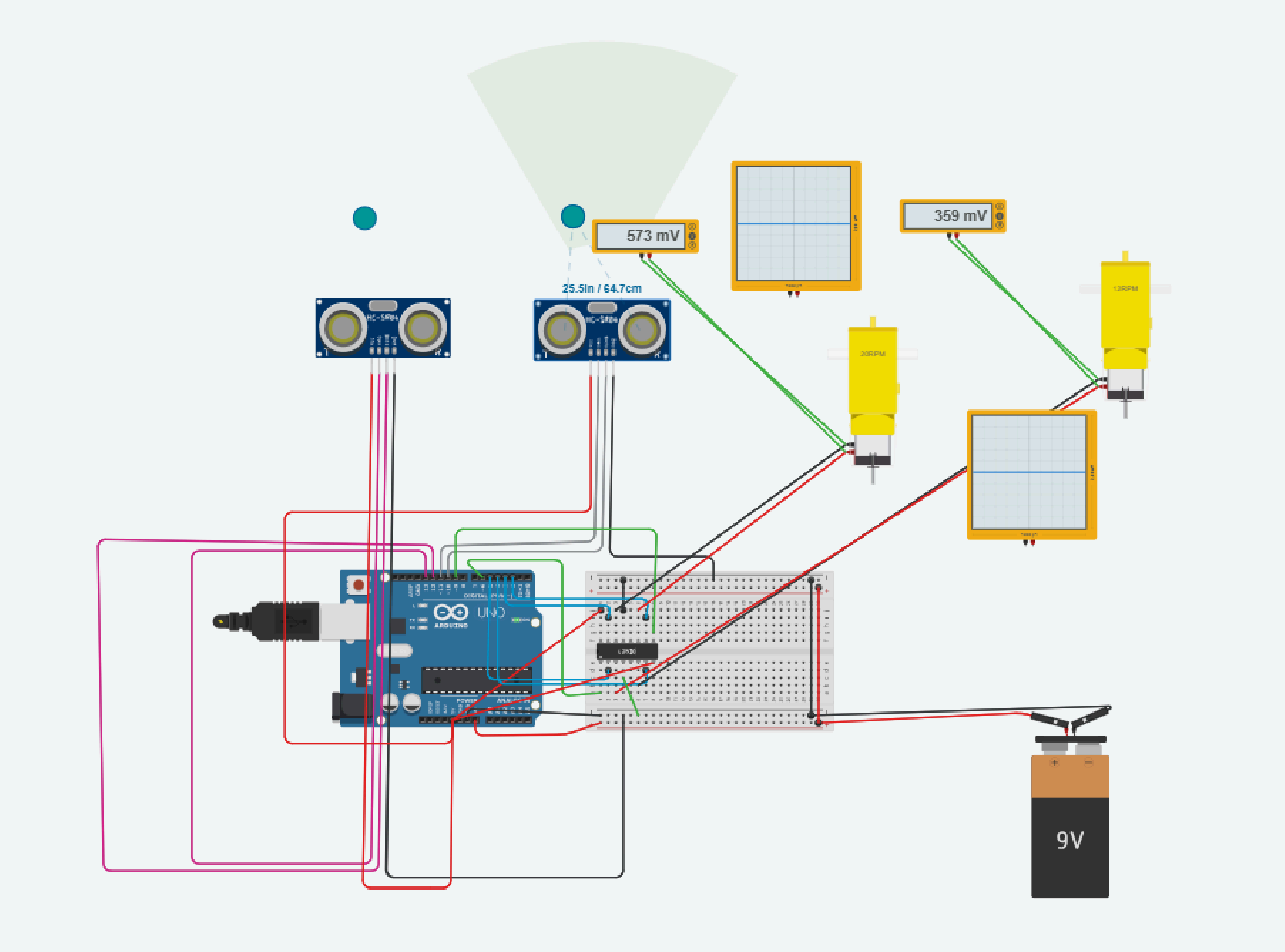
## If Right Distance < Left Distance,

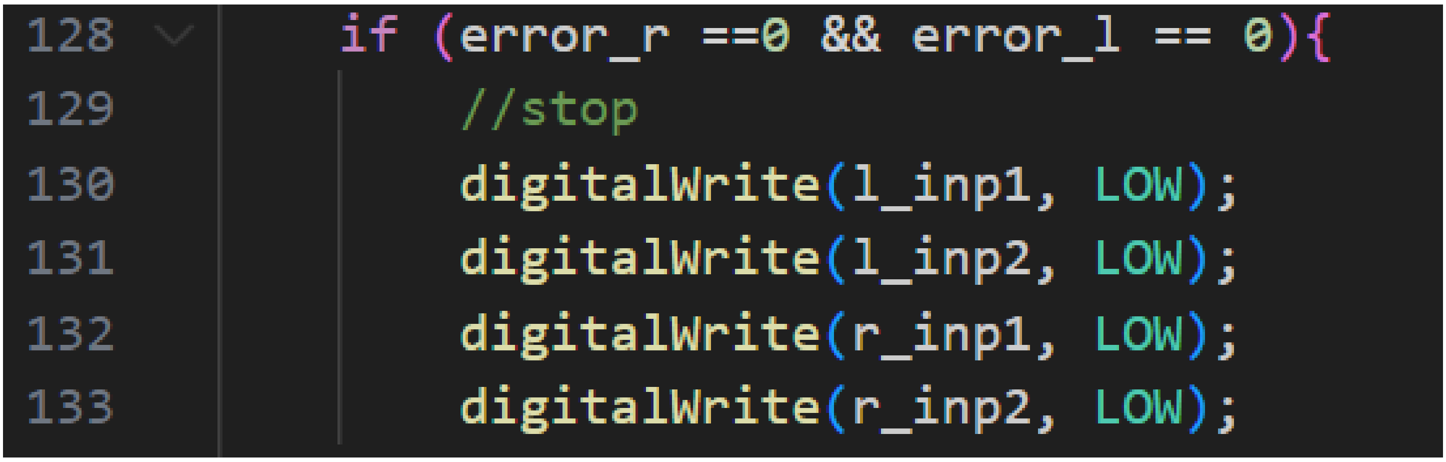


### If Left Distance > Right Distance ,

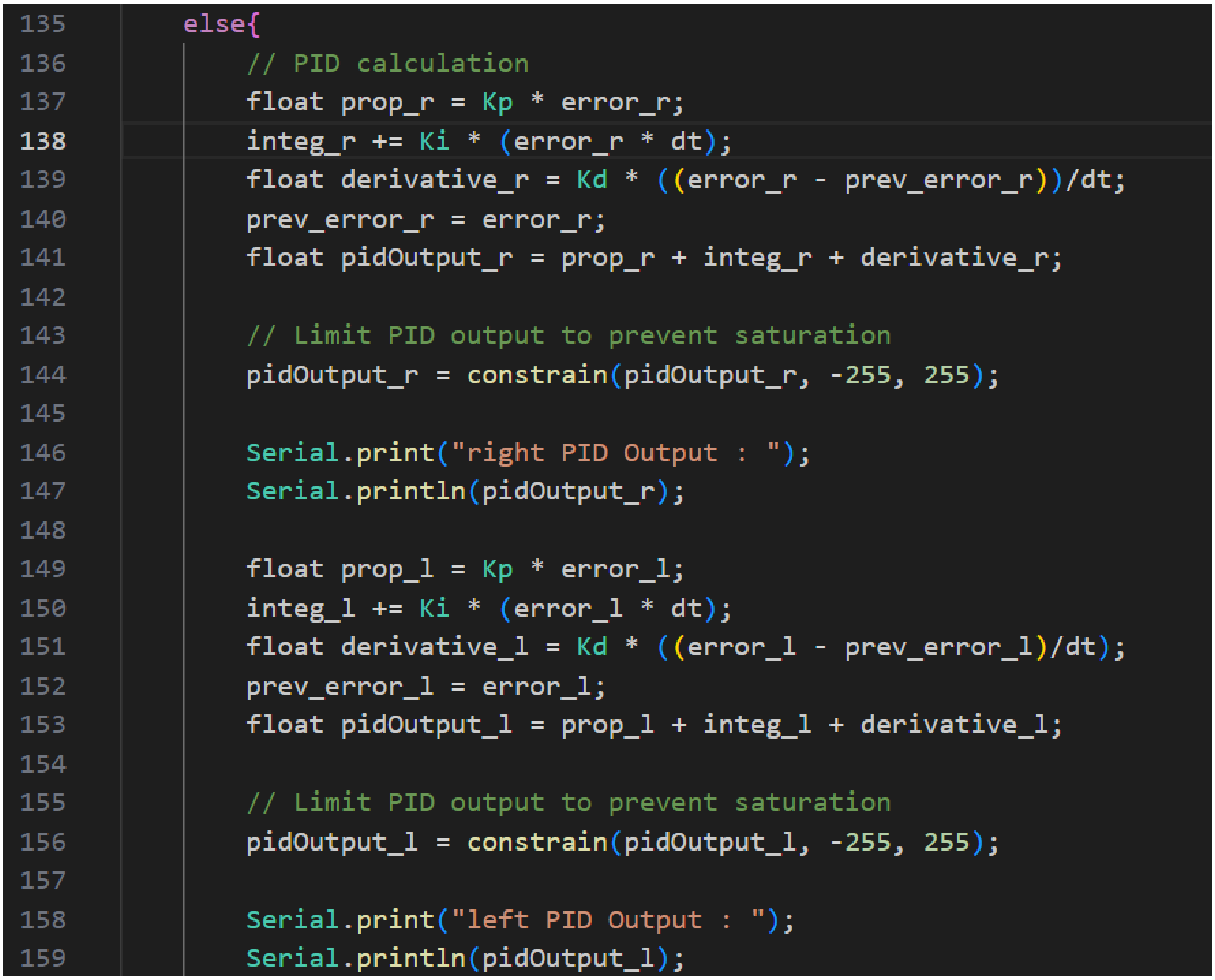


### If Left Distance < Right Distance ,

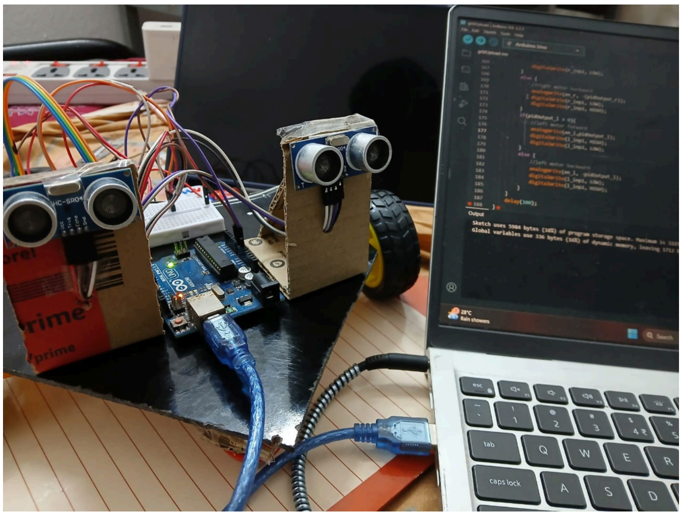


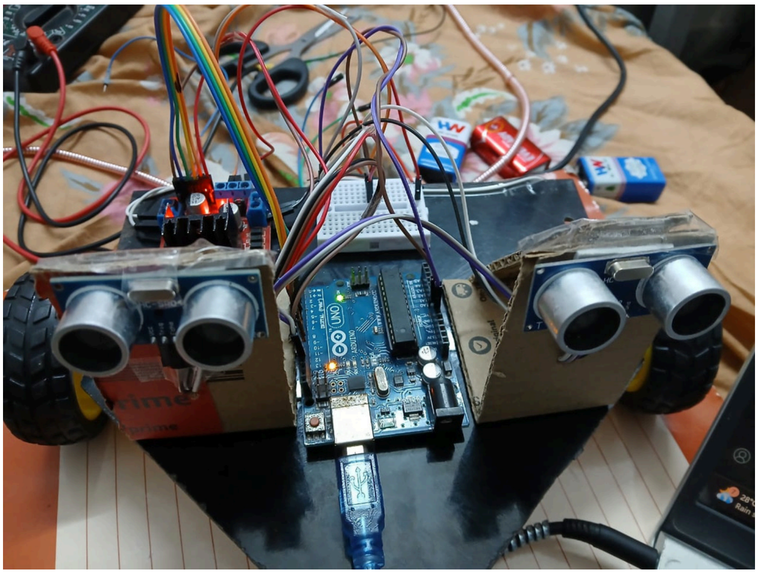


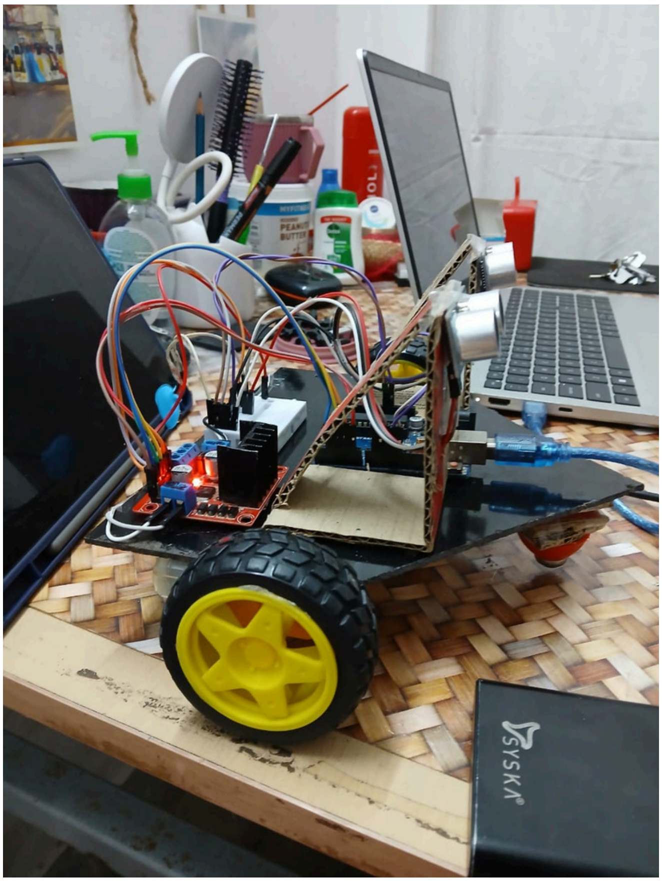
```
    if (error_r ==0 && error_l == 0){
        //stop
        digitalWrite(l_inp1, LOW);
        digitalWrite(l_inp2, LOW);
        digitalWrite(r_inp1, LOW);
        digitalWrite(r_inp2, LOW);
```



```
    else{
        // PID calculation
        float prop_r = Kp * error_r;
        integ_r += Ki * (error_r * dt);
        float derivative_r = Kd * ((error_r - prev_error_r))/dt;
        prev_error_r = error_r;
        float pidOutput_r = prop_r + integ_r + derivative_r;

        // Limit PID output to prevent saturation
        pidOutput_r = constrain(pidOutput_r, -255, 255);

        Serial.print("right PID Output : ");
        Serial.println(pidOutput_r);

        float prop_l = Kp * error_l;
        integ_l += Ki * (error_l * dt);
        float derivative_l = Kd * ((error_l - prev_error_l)/dt);
        prev_error_l = error_l;
        float pidOutput_l = prop_l + integ_l + derivative_l;

        // Limit PID output to prevent saturation
        pidOutput_l = constrain(pidOutput_l, -255, 255);

        Serial.print("left PID Output : ");
        Serial.println(pidOutput_l);
```




UNO
prime


PEANUT BUTTER
SYSKA