1. **Project Problem**

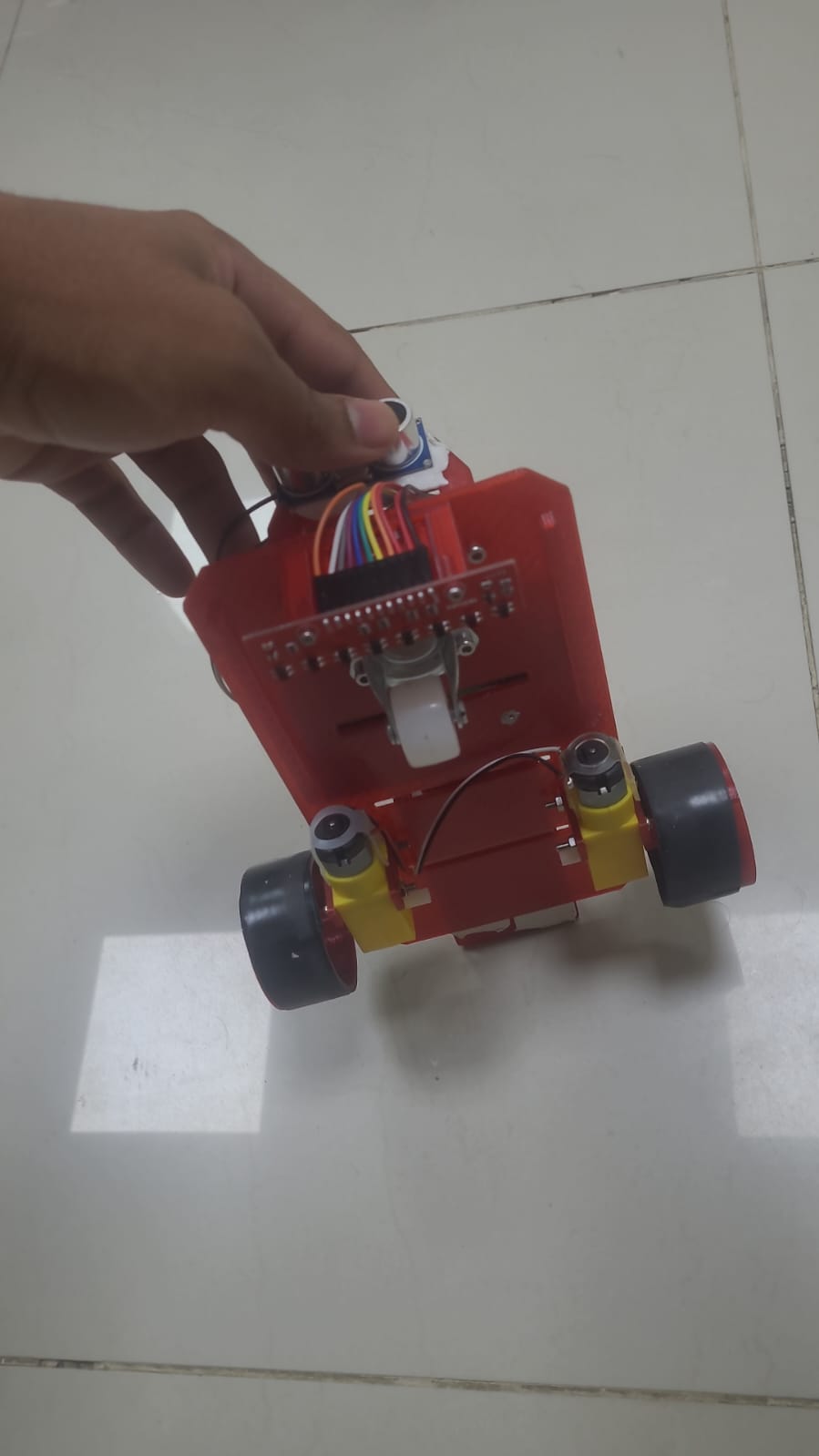
The main objective of this project is to efficiently navigate the robot along the specified track with full autonomy, earning maximum points upon reaching each milestone. This involves maximizing speed on curves and corners, consistently following the black line to achieve the quickest completion time, and automatically pausing to avoid collisions with obstacles until they are cleared, after which the robot will resume its path autonomously.

1. **Tasks**
   1. Continue moving forward if the robot is on the line.
   2. Make right or left turns as needed to remain on the line.
   3. The robot should stop when it detects an obstacle ahead.
   4. Stop and activate a buzzer upon reaching the end of the track.
2. **System Design**

The robot will only work on a line following robot arena that is black and white. We’re using 8 sensors in the qtr-8a array sensor at the bottom front of the robot to detect the line and then send the signals to the robot to follow the line. It also has an Ultrasonic sensor that will detect any object infront of the robot and send a signal to the robot to stop. The robot has two wheels each with its motor that is connected to the motor driver which is connected to the Arduino. It also has a caster wheel in the bottom front of the robot for balance.

**[Pictures of the robot]**

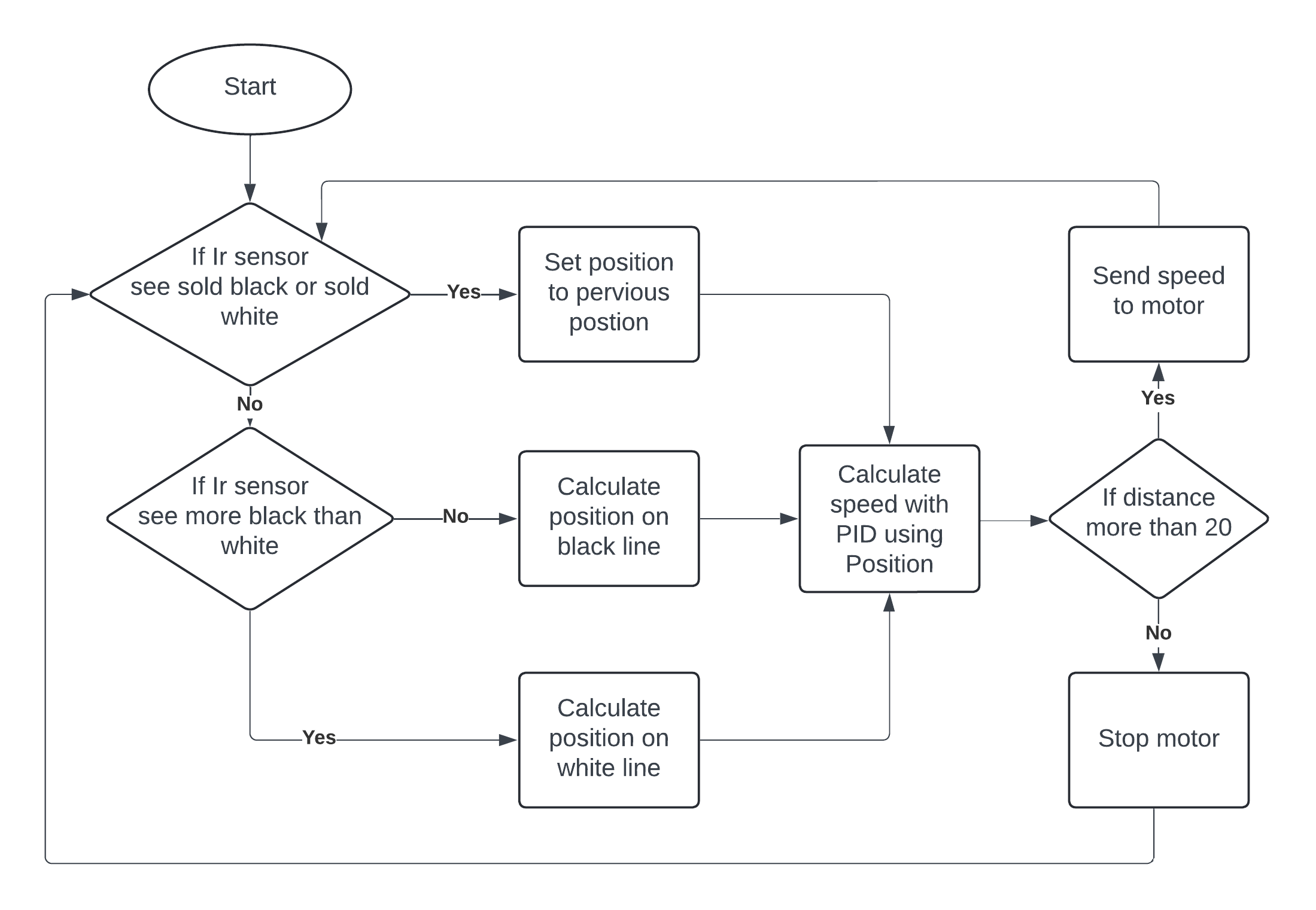
A red toy car with wires and wires

Description automatically generated****

**VIDEO** to a demonstration video of a robot completing the steps

Is in blackboard sublimation.

**Flow Chart:**

****

1. **Design Analysis**

motorDriver();

Set the pin of the motor Driver.

Ultrasonic();

Set the pin of the ultrasonic.

rotateMotor (int rightMotorSpeed, int leftMotorSpeed);

this method helps in converting the speed and direction we want to something that the motor driver will understand.

moves();

this method is used for phase 1 for the 50 cm and the rotation.

follow();

this method is for calculating the distance of the ultrasonic and pos(which is the position of the sensor to the line) and d(which is the percentage of the sensor that see black) of the Ir sensor and then it use this values with PID to calculate the speed of each motor.

1. **Conclusion**

In summary, the robotics competition was a valuable learning experience for all involved. By programming the robot to react autonomously to sensor inputs in different scenarios, students gained practical knowledge of programming and robot navigation. The competition also fostered a spirit of competitiveness and dedication among participants, motivating them to invest significant time and effort in tackling challenges and finding solutions. Overall, the event not only enhanced technical skills but also instilled a passion for innovation and teamwork within the participating students.

1. **References**

* Help in toning the PID

(Controlling Self Driving Cars)

<https://youtu.be/4Y7zG48uHRo?feature=shared>

* help in wiring the qtr8 a Ir sensor and implementing PID

(How to Write a Simple PID Line Follow Algorithm)

<https://youtu.be/PP4fvBVe3rI?feature=shared>

1. **Appendix**

Our Code:

#define MOTOR\_SPEED 105

double perpos = 0;

double sensors[8];

bool W = false;

bool B = true;

// PID Properties

const double KP = 27;

const double KD = 243;

double lastError = 0;

const double Goal = 3.5;

//left motor

int rightMotorPin1 = 10;

int rightMotorPin2 = 11;

//Right motor

int leftMotorPin1 = 5;

int leftMotorPin2 = 6;

//Ultrasonic

const int trigPin = 3;

const int echoPin = 2;

long duration;

int distance;

void setup()

{

  //pinMode(13,OUTPUT);

  motorDriver();

  Ultrasonic();

  //Serial.begin(9600);

  //delay(10000);

  //rotateMotor(255, 255);

  //moves();

}

void loop()

{

  follow();

}

void follow(){

  digitalWrite(trigPin, LOW);

  delayMicroseconds(2);

  // Sets the trigPin on HIGH state for 10 micro seconds

  digitalWrite(trigPin, HIGH);

  delayMicroseconds(10);

  digitalWrite(trigPin, LOW);

  // Reads the echoPin, returns the sound wave travel time in microseconds

  duration = pulseIn(echoPin, HIGH);

  // Calculating the distance

  distance = duration \* 0.034 / 2;

  int sensors[] = {analogRead(A0), analogRead(A1), analogRead(A2), analogRead(A3), analogRead(A4), analogRead(A5), analogRead(A6), analogRead(A7)};

  double pos = 0;

  double d = 0;

  double sen[8];

  for (uint8\_t i = 0; i < 8; i++)

  {

    double value = sensors[i];

    Serial.print(value);

    Serial.print(" ");

    if (value >=900){

      value = 900;

    }

    if (value < 800){

      value = 800;

    }

    value = (value-800)/100;

    d = d + value;

    sen[i] =value;

  }

  if (d < 0.1 ||d > 7.9){

    pos = perpos;

  }else if(d > 5){

    pos = ((1-sen[1])\*1+(1-sen[2])\*2+(1-sen[3])\*3+(1-sen[4])\*4+(1-sen[5])\*5+(1-sen[6])\*6+(1-sen[7])\*7)/((1-sen[0])+(1-sen[1])+(1-sen[2])+(1-sen[3])+(1-sen[4])+(1-sen[5])+(1-sen[6])+(1-sen[7]));

    W = true;

    B = false;

  }else{

    pos = (sen[1]\*1+sen[2]\*2+sen[3]\*3+sen[4]\*4+sen[5]\*5+sen[6]\*6+sen[7]\*7)/d;

    B = true;

  }

  perpos  = pos;

  double error = Goal - pos;

  double adjustment = KP\*error + KD\*(error - lastError);

  lastError = error;

  if(d > 5 and (W and B)){rotateMotor(0,0);}

  else if(distance > 20){

    rotateMotor(constrain(MOTOR\_SPEED + adjustment, 0, MOTOR\_SPEED),constrain(MOTOR\_SPEED - adjustment, 0, MOTOR\_SPEED));

  }else{rotateMotor(0,0);}

  //Serial.println(pos);

}

void Ultrasonic(){

  pinMode(trigPin, OUTPUT);

  pinMode(echoPin, INPUT);

}

void motorDriver(){

  pinMode(rightMotorPin1, OUTPUT);

  pinMode(rightMotorPin2, OUTPUT);

  pinMode(leftMotorPin1, OUTPUT);

  pinMode(leftMotorPin2, OUTPUT);

}

void moves(){

  delay(3000);

  rotateMotor(150,150);

  delay(900);

  rotateMotor(0,0);

  delay(3000);

  rotateMotor(150,-150);

  delay(1400);

  rotateMotor(0,0);

  delay(900);

}

void rotateMotor(int rightMotorSpeed, int leftMotorSpeed)

{

  if (rightMotorSpeed < 0)

  {

    digitalWrite(rightMotorPin1, LOW);

    analogWrite(rightMotorPin2, abs(rightMotorSpeed));

  }

  else if (rightMotorSpeed > 0)

  {

    analogWrite(rightMotorPin1, abs(rightMotorSpeed));

    digitalWrite(rightMotorPin2, LOW);

  }

  else

  {

    digitalWrite(rightMotorPin1, LOW);

    digitalWrite(rightMotorPin2, LOW);

  }

  if (leftMotorSpeed < 0)

  {

    digitalWrite(leftMotorPin1, LOW);

    analogWrite(leftMotorPin2, abs(leftMotorSpeed));

  }

  else if (leftMotorSpeed > 0)

  {

    analogWrite(leftMotorPin1, abs(leftMotorSpeed));

    digitalWrite(leftMotorPin2, LOW);

  }

  else

  {

    digitalWrite(leftMotorPin1, LOW);

    digitalWrite(leftMotorPin2, LOW);

  }

}