**Lab 3 - Arduino Smart Car Kit Basic Movements**

22BRS1187 – Hrijumana Dubey

22BRS1111 – Utkarsh Pandey

22BRS1130 – Ankit Kumar

1. **Github video link:-** <https://github.com/hrijumanadubey/Cognitive-Robotics-Lab/tree/9863e6de7122851e3bc60f6176f9a700ebded34b/Arduino%20Smart%20Car>

1. **Code:-**

// Define motor control pins

#define ENA 5 // Left motor speed (PWM)

#define ENB 6 // Right motor speed (PWM)

#define IN1 9 // Left motor direction

#define IN2 11

#define IN3 7 // Right motor direction

#define IN4 8

// Define ultrasonic sensor pins

#define trigPin A5 // Trigger pin connected to A5

#define echoPin A4 // Echo pin connected to A4

// Speed ranges

const int minSpeed = 100;

const int maxSpeed = 255;

long duration;

int distance;

void setup() {

// Set motor pins as output

pinMode(ENA, OUTPUT);

pinMode(ENB, OUTPUT);

pinMode(IN1, OUTPUT);

pinMode(IN2, OUTPUT);

pinMode(IN3, OUTPUT);

pinMode(IN4, OUTPUT);

// Set ultrasonic sensor pins

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

Serial.begin(9600);

randomSeed(analogRead(0)); // Initialize random for obstacle avoid turns

}

void loop() {

// One cycle: forward, backward, left turn, right turn

performMovementWithObstacleCheck(moveForward, 3000); // Forward ~3 seconds or until obstacle

performMovementWithObstacleCheck(moveBackward, 3000); // Backward ~3 seconds or until obstacle

performMovementWithObstacleCheck(leftTurn, 1000); // Left turn ~1 second

performMovementWithObstacleCheck(rightTurn, 1000); // Right turn ~1 second

// Repeat cycle automatically

}

void performMovementWithObstacleCheck(void (\*moveFunc)(), unsigned long maxDuration) {

unsigned long startTime = millis();

// Accelerate while moving

for (int speedVal = minSpeed; speedVal <= maxSpeed; speedVal += 15) {

moveFunc();

analogWrite(ENA, speedVal);

analogWrite(ENB, speedVal);

distance = getDistance();

Serial.print("Distance during accel: ");

Serial.println(distance);

if (distance > 0 && distance < 15) {

stopMotors();

moveAwayFromObstacle();

return;

}

delay(100); // Small delay to allow PWM effect and sensor reading

if (millis() - startTime > maxDuration / 2) {

break; // Limit acceleration phase time

}

}

// Maintain max speed

unsigned long moveStart = millis();

while (millis() - moveStart < (maxDuration / 2)) {

moveFunc();

analogWrite(ENA, maxSpeed);

analogWrite(ENB, maxSpeed);

distance = getDistance();

Serial.print("Distance during max speed: ");

Serial.println(distance);

if (distance > 0 && distance < 15) {

stopMotors();

moveAwayFromObstacle();

return;

}

delay(50); // Reduced delay for more frequent checking

}

// Decelerate before stopping

for (int speedVal = maxSpeed; speedVal >= minSpeed; speedVal -= 15) {

moveFunc();

analogWrite(ENA, speedVal);

analogWrite(ENB, speedVal);

distance = getDistance();

Serial.print("Distance during decel: ");

Serial.println(distance);

if (distance > 0 && distance < 15) {

stopMotors();

moveAwayFromObstacle();

return;

}

delay(100);

}

stopMotors();

}

void moveAwayFromObstacle() {

Serial.println("Obstacle detected, moving away!");

// Move backward for 1 second to create some space

moveBackward();

analogWrite(ENA, minSpeed + 50);

analogWrite(ENB, minSpeed + 50);

delay(1000);

// Stop briefly

stopMotors();

delay(200);

// Random left or right turn to avoid obstacle

int turnDirection = random(0, 2); // 0 = left, 1 = right

if (turnDirection == 0) {

Serial.println("Turning left to avoid obstacle");

leftTurn();

} else {

Serial.println("Turning right to avoid obstacle");

rightTurn();

}

analogWrite(ENA, minSpeed + 50);

analogWrite(ENB, minSpeed + 50);

delay(800);

stopMotors();

delay(200);

}

// ========== Movement Functions ==========

void moveForward() {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

}

void moveBackward() {

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

}

void leftTurn() {

digitalWrite(IN1, LOW);

digitalWrite(IN2, HIGH);

digitalWrite(IN3, HIGH);

digitalWrite(IN4, LOW);

}

void rightTurn() {

digitalWrite(IN1, HIGH);

digitalWrite(IN2, LOW);

digitalWrite(IN3, LOW);

digitalWrite(IN4, HIGH);

}

void stopMotors() {

analogWrite(ENA, 0);

analogWrite(ENB, 0);

}

// ========== Ultrasonic Sensor Function ==========

long getDistance() {

// Send trigger pulse

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

// Read echo pulse duration with timeout (30 ms)

duration = pulseIn(echoPin, HIGH, 30000);

if (duration == 0) {

// No echo received (timeout), treat as no obstacle

return -1;

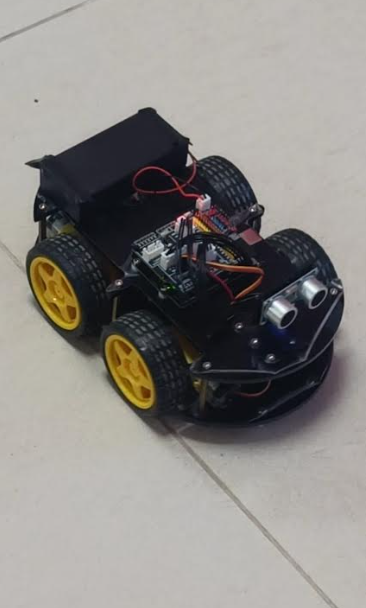
}

// Calculate distance in cm

return duration \* 0.034 / 2;

}

1. **Screenshot:-**

****