Abstraction:

Abstraction simplifies complex systems by breaking them into key components. For the obstacle-avoiding car:

1. Sensor: Detects distance, hiding how it works internally.

2. Motor Control: Manages movement with simple commands like forward or stop, abstracting motor details.

3. Car Logic: Coordinates the sensor and motors, defining behavior like obstacle avoidance, without exposing internal mechanics.

Program:

#define trigPin 9

#define echoPin 10

#define motor1Pin1 2

#define motor1Pin2 3

#define motor2Pin1 4

#define motor2Pin2 5

#define motor1Speed 6

#define motor2Speed 7

long duration;

int distance;

void setup() {

pinMode(trigPin, OUTPUT);

pinMode(echoPin, INPUT);

pinMode(motor1Pin1, OUTPUT);

pinMode(motor1Pin2, OUTPUT);

pinMode(motor2Pin1, OUTPUT);

pinMode(motor2Pin2, OUTPUT);

pinMode(motor1Speed, OUTPUT);

pinMode(motor2Speed, OUTPUT);

Serial.begin(9600);

}

void loop() {

// Measure distance

digitalWrite(trigPin, LOW);

delayMicroseconds(2);

digitalWrite(trigPin, HIGH);

delayMicroseconds(10);

digitalWrite(trigPin, LOW);

duration = pulseIn(echoPin, HIGH);

distance = duration \* 0.034 / 2;

Serial.print("Distance: ");

Serial.println(distance);

if (distance < 20) { // If an obstacle is closer than 20 cm

stopCar();

delay(500);

turnRight(); // Turn the car right

delay(500);

} else {

moveForward(); // Move forward

}

delay(100);

}

void moveForward() {

analogWrite(motor1Speed, 255);

analogWrite(motor2Speed, 255);

digitalWrite(motor1Pin1, HIGH);

digitalWrite(motor1Pin2, LOW);

digitalWrite(motor2Pin1, HIGH);

digitalWrite(motor2Pin2, LOW);

}

void stopCar() {

digitalWrite(motor1Pin1, LOW);

digitalWrite(motor1Pin2, LOW);

digitalWrite(motor2Pin1, LOW);

digitalWrite(motor2Pin2, LOW);

}

void turnRight() {

analogWrite(motor1Speed, 255);

analogWrite(motor2Speed, 255);

digitalWrite(motor1Pin1, LOW);

digitalWrite(motor1Pin2, HIGH);

digitalWrite(motor2Pin1, HIGH);

digitalWrite(motor2Pin2, LOW);

}