Lab name : Introduction to Integrated Robotic Car System

Description : This lab focuses on understanding the integration of various components required to build a functional robotic car. Throughout the session, participants will learn to program the car's movement, control sensors, and manage power systems. Key learning objectives include: Sensor Integration: Participants will configure and calibrate sensors like ultrasonic and infrared to help the car navigate obstacles. Motor Control: Detailed programming of motor movements allows the robotic car to achieve complex movement patterns, including turns and speed adjustments. Microcontroller Programming: The microcontroller acts as the car's "brain," enabling control over inputs and outputs to achieve autonomous operation. By the end of this lab, students will have a functional model that can navigate a basic track autonomously.

Level : hard

Setting Up and Calibrating Sensors for Obstacle Detection

Objective:
Learn to set up and calibrate ultrasonic sensors on the robotic car to accurately detect obstacles at various distances.
Materials Needed:

- Robotic car chassis
- Ultrasonic sensors
- Jumper wires
- Microcontroller (e.g., Arduino or Raspberry Pi)
- Breadboard

Steps:

1. **Assemble the Robotic Car**:
   Begin by attaching the wheels, motors, and sensor mounts to the chassis. Secure all components firmly to ensure stable movement during testing.
2. **Connect Ultrasonic Sensors**:
   Mount the ultrasonic sensors on the front of the car. Use jumper wires to connect the sensor’s VCC, GND, Trigger, and Echo pins to the microcontroller.

3. **Calibrate Sensors**:
   Write a basic program to measure the distance between the car and obstacles. An example code in Arduino is provided below:

```
const int trigPin = 9;
const int echoPin = 10;
long duration;
int distance;

void setup() {
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.begin(9600);
}

void loop() {
    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.print(distance);
    Serial.println(" cm");
    delay(500);
}
```

Basic to Integrated System

Objective:
Gain foundational knowledge and practical skills in building an integrated robotic car. By the end of this lab, participants will understand sensor integration, motor control, and basic microcontroller programming to create a simple robotic car that responds to its environment.

Materials Needed:

- Robotic car chassis with wheels and motor mounts
- Motors with motor driver (e.g., L298N or similar)
- Ultrasonic sensor (for obstacle detection)
- Microcontroller (e.g., Arduino Uno or Raspberry Pi)
- Jumper wires, breadboard
- Battery pack for power supply
- Miscellaneous mounting hardware (screws, bolts)

Part 1: Setting Up the Chassis and Motors

- **Steps**:
- **Assemble the Chassis**:
  Attach the wheels, motors, and any mounts onto the robotic car chassis according to the instructions provided in your kit.
- **Wiring Motors to Motor Driver**:
  - Connect each motor to the motor driver module. Typically, two wires from each motor go to the output pins of the motor driver.
  - Connect the motor driver’s VCC and GND pins to the power source and ground of the microcontroller.

- **Coding Basic Motor Control**:

```
const int motorPin1 = 3;
const int motorPin2 = 4;

void setup() {
    pinMode(motorPin1, OUTPUT);
    pinMode(motorPin2, OUTPUT);
}

void moveForward() {
    digitalWrite(motorPin1, HIGH);
    digitalWrite(motorPin2, LOW);
}

void stopCar() {
    digitalWrite(motorPin1, LOW);
    digitalWrite(motorPin2, LOW);
}

void loop() {
    moveForward();
    delay(2000);  // Move forward for 2 seconds
    stopCar();
    delay(1000);  // Stop for 1 second
}
```

- **Testing**:
  Power on the car and run the code. Verify that the car moves forward and stops as programmed. Adjust wiring or code if any issues arise.

Part 2: Integrating the Ultrasonic Sensor for Obstacle Detection

- **Steps**:
- **Mounting the Ultrasonic Sensor**:
  Attach the ultrasonic sensor to the front of the car to allow for obstacle detection.
- **Wiring the Ultrasonic Sensor**:
  - Connect the VCC and GND of the ultrasonic sensor to the microcontroller's power and ground pins.
  - Connect the Trigger and Echo pins to two digital I/O pins on the microcontroller (e.g., pin 9 for Trigger and pin 10 for Echo).
- **Programming the Sensor for Distance Measurement**:

```
const int trigPin = 9;
const int echoPin = 10;

void setup() {
    pinMode(trigPin, OUTPUT);
    pinMode(echoPin, INPUT);
    Serial.begin(9600);
}

int getDistance() {
    long duration;
    int distance;
    digitalWrite(trigPin, LOW);
    delayMicroseconds(2);
    digitalWrite(trigPin, HIGH);
```

```
    delayMicroseconds(10);
    digitalWrite(trigPin, LOW);

    duration = pulseIn(echoPin, HIGH);
    distance = duration * 0.034 / 2;  // Convert to cm
    return distance;
}

void loop() {
    int distance = getDistance();
    Serial.print("Distance: ");
    Serial.print(distance);
    Serial.println(" cm");
    delay(500);
}
```

- **Testing the Sensor**:
  Upload the code to the microcontroller and open the Serial Monitor to check if the sensor accurately measures distances. Adjust sensor positioning if readings are inconsistent.

Part 3: Integrating Sensor and Motor for Autonomous Movement

Objective:
Program the robotic car to autonomously move forward and stop when an obstacle is detected within a specified range.

Steps:

1. **Combining Motor and Sensor Code**:
   Update the code to include motor movement based on sensor distance data.

```
void loop() {
    int distance = getDistance();

    if (distance < 15) {  // If obstacle is within 15 cm
        stopCar();
        delay(500);  // Stop briefly
    } else {
        moveForward();
    }

    delay(100);
}
```

- **Testing Autonomous Functionality**:
  Place obstacles at varying distances in front of the car. Observe the car's behavior, ensuring it stops when close to an object and resumes moving when the path is clear.
- **Debugging and Adjustments**:
  - Modify the threshold distance (e.g., 15 cm) in the code if needed.
  - Adjust the motor speed, turning angles, or stopping time as per the car's response.

- ## Wrap-Up

- **Review**:

Review your code and document how each component (motor, sensor, microcontroller) works together in the integrated robotic car.

- **Future Enhancements**:
  Consider adding features like turning when an obstacle is detected, adding multiple sensors, or enabling Bluetooth control.
