Triple Ultrasonic Sensor-Based Reverse Car Parking System with OLED Display

This document details the development of an Arduino-based reverse car parking system that uses three ultrasonic sensors to enhance safety and accuracy when parking. The system integrates an OLED display for real-time distance feedback and a buzzer for audible alerts. This project aims to mitigate the challenges of parking in tight spaces and low-visibility conditions by providing drivers with comprehensive obstacle detection.

The project team consists of-

Adrika Sikidar
Baljinder Kaur
Diya Seal
Gunjan Verma (SPOC)
Jahnavi Sharma

Components and Their Uses

HC-SR04 Ultrasonic Sensor (x3)

Detects the distance between the vehicle and nearby objects by sending an ultrasonic pulse and listening for its echo. Three sensors are used: center, left, and right.

Microcontroller ESP32 WROOM

The "brain" of the system. It processes sensor input, controls output devices, reads distance values, and sends signals to the buzzer and OLED display.

Motor Driver L298N

Drives the motors by supplying power and directional control signals based on input from the ESP32, simulating reverse motion.

BO Motors

The driving motors for the vehicle, rotating based on signals from the motor driver to simulate the car's motion during parking.

5V Buzzer

Provides an audio alert when an obstacle is detected. Activated by a HIGH signal from the ESP32, with the beeping frequency increasing as the object gets closer.

OLED Display

Shows the distance to the obstacle in a visual format, displaying real-time data, alerts like "Object Detected," and distance values.

Resistors

Protect sensitive components by controlling the current flow, mainly used in sensor circuits to ensure stable operation.

Mini Breadboard

Facilitates easy and temporary circuit connections without soldering, used for prototyping and testing sensor, display, and ESP32 connections.

Connecting Wires: Physically connect the components in the circuit, allowing the flow of electricity and data between components such as the ESP32, sensors, display, buzzer, motors, and breadboard connections.

Arduino Code Overview

The Arduino code integrates sensor readings, motor control, and display functionalities to create a comprehensive parking assistance system. The code includes libraries for the OLED display and defines pins for various components such as ultrasonic sensors, IR sensors, motors, and the buzzer.

Key functions in the code:

- **readDistanceCM:** Measures distance using ultrasonic sensors.
- **stopMotors:** Halts motor movement.
- **moveForward:** Initiates forward motor movement.
- **setup:** Initializes serial communication, configures pin modes, and initializes the OLED display.
- **loop:** Continuously reads sensor data, displays information on the OLED screen, and controls motor and buzzer functionalities based on sensor inputs.

The loop function checks for obstacle presence and alignment using IR sensors and ultrasonic sensors. It controls the motors to simulate parking and activates the buzzer as an alert mechanism.

```
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

#define SCREEN_WIDTH 128
#define SCREEN_HEIGHT 64
#define OLED_RESET -1
#define SCREEN_ADDRESS 0x3C
Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &SPI, 22, 21, 5);

#define trig1 27
#define echo1 26
#define trig2 25
#define echo2 33
#define trig3 32
#define echo3 35

#define ir1 34
#define ir2 36

#define in1 14
#define in2 12
#define in3 13
#define in4 15

#define buzzer 4

long readDistanceCM(int trigPin, int echoPin) {
  digitalWrite(trigPin, LOW);
  delayMicroseconds(2);
  digitalWrite(trigPin, HIGH);
  delayMicroseconds(10);
  digitalWrite(trigPin, LOW);
  long duration = pulseIn(echoPin, HIGH);
  return duration * 0.034 / 2;
}

void stopMotors() {
  digitalWrite(in1, LOW);
  digitalWrite(in2, LOW);
  digitalWrite(in3, LOW);
  digitalWrite(in4, LOW);
}

void moveForward() {
  digitalWrite(in1, HIGH);
  digitalWrite(in2, LOW);
  digitalWrite(in3, HIGH);
  digitalWrite(in4, LOW);
}

void setup() {
  Serial.begin(115200);

  pinMode(trig1, OUTPUT);
  pinMode(echo1, INPUT);
  pinMode(trig2, OUTPUT);
  pinMode(echo2, INPUT);
  pinMode(trig3, OUTPUT);
  pinMode(echo3, INPUT);

  pinMode(ir1, INPUT);
  pinMode(ir2, INPUT);

  pinMode(in1, OUTPUT);
  pinMode(in2, OUTPUT);
  pinMode(in3, OUTPUT);
  pinMode(in4, OUTPUT);

  pinMode(buzzer, OUTPUT);

  if (!display.begin(SSD1306_SWITCHCAPVCC, SCREEN_ADDRESS)) {
    Serial.println(F("OLED failed"));
    for (;;);
  }

  display.clearDisplay();
  display.setTextSize(1);
  display.setTextColor(SSD1306_WHITE);
  display.setCursor(0, 0);
  display.println("Parking Assistant Ready");
  display.display();
  delay(1000);
}

void loop() {
  long distRear = readDistanceCM(trig1, echo1);
  long distLeft = readDistanceCM(trig2, echo2);
  long distRight = readDistanceCM(trig3, echo3);

  bool irLeft = digitalRead(ir1);
  bool irRight = digitalRead(ir2);

  display.clearDisplay();
  display.setCursor(0, 0);
  display.print("Rear: ");
  display.println(distRear);
  display.print("Left: ");
  display.println(distLeft);
  display.print("Right: ");
  display.println(distRight);
  display.display();

  if (irLeft == LOW && irRight == LOW && distRear > 15) {
    moveForward();
    digitalWrite(buzzer, LOW);
  } else {
    stopMotors();
    digitalWrite(buzzer, HIGH); // Alert when obstacle or not aligned
  }

  delay(100);
}
```

Pin Configuration

Detailed below are the pin configurations used in the Arduino project. These configurations specify the connections between the ESP32 microcontroller and the various sensors, motor drivers, and output devices used in the reverse car parking system.

- **Ultrasonic Sensor 1:** Trig - GPIO 27, Echo - GPIO 26
- **Ultrasonic Sensor 2:** Trig - GPIO 25, Echo - GPIO 33
- **Ultrasonic Sensor 3:** Trig - GPIO 32, Echo - GPIO 35
- **IR Sensor 1:** OUT - GPIO 34
- **IR Sensor 2:** OUT - GPIO 36
- **Motor Driver:** IN1 (left motor A) - GPIO 14, IN2 (left motor A) - GPIO 12, IN3 (left motor B) - GPIO 13, IN4 (left motor B) - GPIO 15
- **Buzzer:** +ve - GPIO 4
- **OLED (SPI):** RES - GPIO 22, DC - GPIO 21, CS - GPIO-5, CLK - GPIO 18