**Project Title : Smart Parking System**

**Phase 5 : Project Documentation**

**Objective:**

In this project we are going to construct a smart vehicle parking system that connects to internet to help a car driver or any vehicle owner to check if there is a vacant parking spot exist in a parking lot even before the driver reach the intended parking lot destination.

**What is Smart Parking System?**

An IoT based parking system is a vehicle parking management system to ease the search for a vacant parking spot in a parking lot through a smartphone. The system utilizes various sensors and microcontrollers with internet capability for detecting parked vehicles and to update the data in real-time on internet.

**The Proposed Design:**

As mentioned above, the proposed smart parking lot circuit will be equipped with several sensors, inexpensive microcontrollers and Wi-Fi module using which a car / any vehicle owner can check if there is a vacant space in a parking lot using his / her phone or tablet or even on computer.

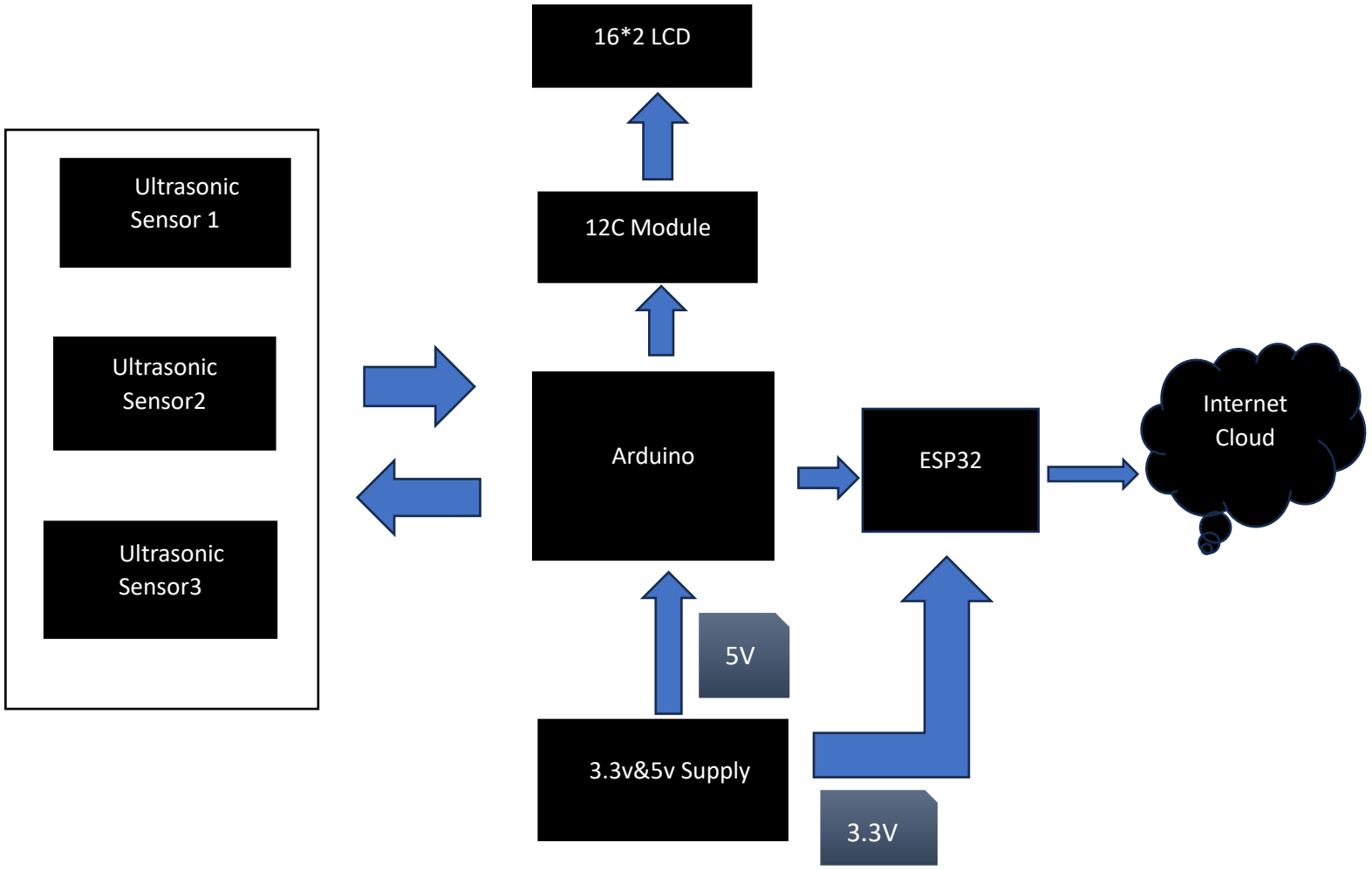
The number of vacant spaces in the smart parking lot can be viewed from anywhere in the world using a URL link or the user can scan a QR code. The scanned / shared URL can be browsed on any web browser to know how many empty parking spot exist in real time.

**Smart Parking System for Electric Vehicle Charging Station:**

The proposed smart parking system is very useful in electric vehicle charging stations and this technology is going to a boon for those who are passing beside the charging stations equipped with this system. Now the motorists can see number of vacant chargers on their smartphone and plan their journey accordingly.

In conclusion, the main purpose of a smart vehicle parking system is to save time and reduce hassle for motorists to find a parking lot with a vacant parking spot; otherwise a driver may need to spend their time to find if there are any vacant parking spot left or should they move on to an another parking lot and this situation may put many motorists to mental stress especially those who are in an urgent circumstances.

**Block Diagram:**




- The circuit we are going to build will be based on the above architecture. An inexpensive Arduino board is going to be the brain of the project.

- A 16 x 2 LCD is utilized for displaying the number of vacant spots locally (without internet). An I2C module is utilized for driving the LCD with just four wires so that GPIO pins can be saved for interfacing the sensors and other modules.

- There are three ultrasonic sensors for detecting 3 cars / vehicles on the parking spot, we are using ultrasonic sensors instead of IR based sensors because if the parking lot is situated outdoors, infrared light from sunlight may interfere with IR sensors and may give incorrect detection of the vehicle, whereas ultrasonic

sensor acts like a mini radar and environmental factors affecting its functionality is minimal.

- Please note that we are constructing a scale down version of the real project; hence we are just using three sensors.

- An ESP32 Wi-Fi module is used for internet connectivity which sends the parking lot's data to a cloud server where general public can view the data in real time. A power supply module is utilized which provides 5V and 3.3V for Arduino, ultrasonic sensors and ESP32 Wi-Fi Module.

- The internet cloud service we are going to use is called "Thingspeak" instead of blynk app where the parking lot's data to be sent, stored and displayed in real time. This concludes the block diagram.

**IOT Device Setup:**

**Arduino:**

- The Arduino in the Smart Parking System acts as the central controller, managing data from ultrasonic sensors, processing parking availability, and driving a local display.
- It enables real-time communication with the ESP32 Wi-Fi module for data transmission to Thingspeak.
- The Arduino's reliability and versatility make it suitable for various parking environments and ensure system stability.
- Its open-source nature allows for customization and adaptability to evolving requirements, ensuring long-term effectiveness.

**Ultrasonic Sensors:**

- An ultrasonic sensor utilizes sound waves to determine the distance between the sensor and an object.Ultrasonic sensors provide precise distance measurements, ensuring that vehicles are parked optimally and safely within the designated spaces.


- The ultrasonic sensor module generates ultrasonic sound at around 40 KHz, these sound waves are inaudible to human beings and propagate through air and if the ultrasonic sound wave hits an obstacle, it reflects back to sensor just like radars.

- If a car or any vehicle is parked, the ultrasonic sound waves hit the parked vehicle and the sensor module detects the reflection and thus existence of a vehicle on a parking spot is detected.

- The ultrasonic sensor module has four pins, Vcc, GND, trigger and echo. The Vcc is connected to 5V supply and GND is connected to GND of the supply. When we apply “HIGH” signal to trigger pin for 10 microseconds, the module generates ultrasonic sound from one of the transducers, when the sound wave hit back the other transducer, the echo pin gets “HIGH” and this signal is detected by Arduino.

- The time taken between generating and detecting the sound wave is calculated and thus a parked vehicle is detected.

**Connectivity Option:**

- The WiFi module is responsible for establishing a wireless network connection within the smart parking infrastructure. It allows sensors, cameras, and other devices deployed in the parking area to communicate with each other and with a central server.

- Here we use ESP32 for wifi Connection.
- This project utilizes a generic ESP32 Wi-Fi module for internet connectivity. The ESP32 is actually a miniature microcontroller board and just like Arduino the ESP32 need a program code to perform its intended function.

**LCD Display Module 16×2 I2C:**

- In this project we are using a 16 x 2 LCD display for displaying parking lot's data locally without the need for internet. The LCD is driven by an I2C adapter module to reduce the number of wires to four; otherwise you need to connect up to 16 wires to Arduino just to drive the display. If the LCD occupies most of the I/O pins, then there won't be any pins left for the sensors.

- The I2C module has 16 pins at the output and just four at the input: Vcc, GND, SDA and SCL. The SDA and SCL are I2C bus pins which are connected to A4 and A5 pins of Arduino respectively and it operates on 5V.

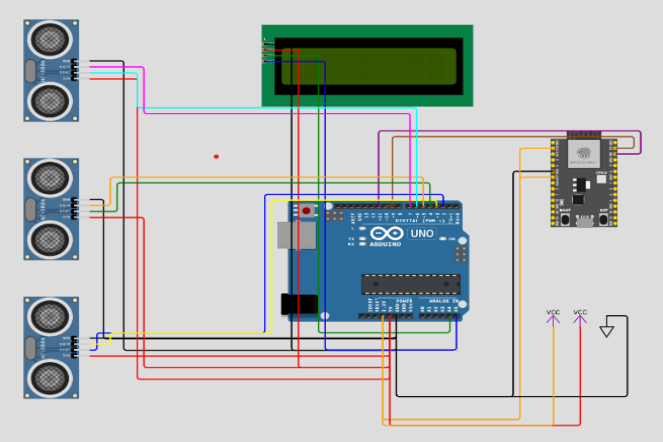
- You can control the contrast of the display by adjusting the trim pot on the I2C adapter module. This concludes about the circuit diagram.

**Communication:**

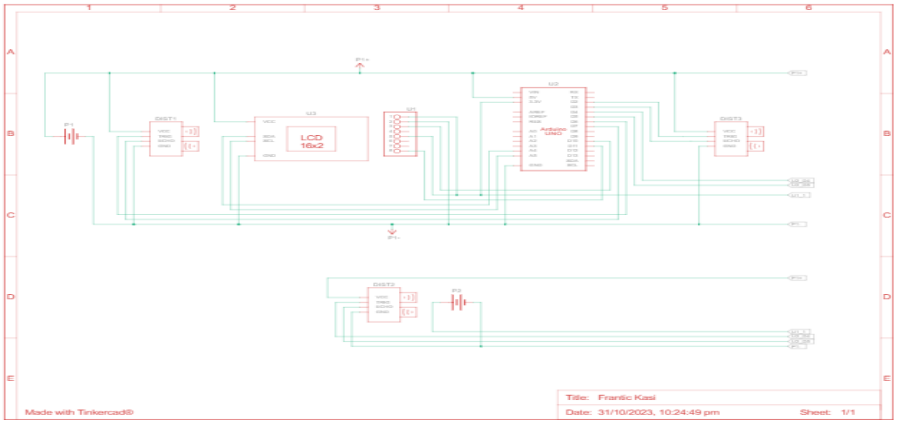
- We are using a (free) cloud service called Thingspeak where we will send parking lot's data to share it with public.
- Steps:
  - First you need to sign up for Thingspeak: Click here
  - Enter the credentials it asks for and create a new channel and do the following to your new channel:
  - Go to **channel settings** and enter the things as shown above and take note of your channel ID which we need to enter it in the program code.
  - Scroll down and **click save** to save the changes.
  - Now click on **API keys tab** and you will see your keys as illustrated below. API keys are responsible for writing and reading the data to your Thingspeak account.
  - Go to **channel settings** and enter the things as shown above and take note of your channel ID which we need to enter it in the program code.
  - Scroll down and **click save** to save the changes.

Now click on **API keys tab** and you will see your keys as illustrated below. API keys are responsible for writing and reading the data to your Thingspeak account.

**Circuit Diagram for Smart Parking Monitoring System:**




**Schematic Diagram:**




**Arduino Code:**

```
#include <LiquidCrystal_I2C.h>
#include <SoftwareSerial.h>
SoftwareSerial mySerial(10, 11);
LiquidCrystal_I2C lcd(0x27, 16, 2); // set the LCD address to 0x27 for a 16 chars and 2 line display
const int trig_1 = 2;
const int echo_1 = 3;
const int trig_2 = 4;
const int echo_2 = 5;
const int trig_3 = 6;
const int echo_3 = 7;
float distanceCM_1 = 0, resultCM_1 = 0;
float distanceCM_2 = 0, resultCM_2 = 0;
float distanceCM_3 = 0, resultCM_3 = 0;
long Time_1, Time_2, Time_3;
float car_1, car_2, car_3;
float Dist_1 = 8.0, Dist_2 = 8.0, Dist_3 = 8.0;
int total = 0, timer_cnt = 0;
void setup()
{
  mySerial.begin(115200);
  pinMode(trig_1, OUTPUT);
  pinMode(trig_2, OUTPUT);
  pinMode(trig_3, OUTPUT);
  pinMode(echo_1, INPUT);
  pinMode(echo_2, INPUT);
  pinMode(echo_3, INPUT);
  digitalWrite(trig_1, LOW);
  digitalWrite(trig_2, LOW);
```

```
  digitalWrite(trig_3, LOW);
  lcd.init();
  lcd.backlight();
  lcd.setCursor(0, 0);
  lcd.print("  SMART PARKING SYSTEM");
  lcd.setCursor(0, 1);
  lcd.print(" USING IOT");
  delay(2000);
  lcd.clear();
}

void loop()
{
  total = 0;
  car_1 = sensor_1();
  car_2 = sensor_2();
  car_3 = sensor_3();
  lcd.setCursor(0, 0);
  lcd.print("CAR1:");
  if (car_1 <= Dist_1)
  {
    lcd.print("OK ");
  }
  else
  {
    total += 1;
  }
  if (car_1 > Dist_1)   lcd.print("NO ");
  lcd.print("CAR2:");
  if (car_2 <= Dist_2)
  {
    lcd.print("OK ");
  }
  else
  {
    total += 1;
  }
  if (car_2 > Dist_2)   lcd.print("NO ");
  lcd.setCursor(0, 1);
  lcd.print("CAR3:");
  if (car_3 <= Dist_3)
  {
    lcd.print("OK ");
  }
  else
  {
```

```
    total += 1;
   }
   if (car_3 > Dist_3)   lcd.print("NO ");
   lcd.print("FREE:");
   lcd.print(total);
   if (timer_cnt >= 50)
   {
    mySerial.print('*');
    mySerial.print(total);
    mySerial  .println('#');
    timer_cnt = 0;
   }
   timer_cnt += 1;
   delay(200);
}

float sensor_1(void)
{
   digitalWrite(trig_1, HIGH);
   delayMicroseconds(10);
   digitalWrite(trig_1, LOW);
   Time_1 = pulseIn(echo_1, HIGH);
   distanceCM_1 = Time_1 * 0.034;
   return resultCM_1 = distanceCM_1 / 2;
}

float sensor_2(void)
{
   digitalWrite(trig_2, HIGH);
   delayMicroseconds(10);
   digitalWrite(trig_2, LOW);
   Time_2 = pulseIn(echo_2, HIGH);
   distanceCM_2 = Time_2 * 0.034;
   return resultCM_2 = distanceCM_2 / 2;
}

float sensor_3(void)
{
   digitalWrite(trig_3, HIGH);
   delayMicroseconds(10);
   digitalWrite(trig_3, LOW);
   Time_3 = pulseIn(echo_3, HIGH);
   distanceCM_3 = Time_3 * 0.034;
   return resultCM_3 = distanceCM_3 / 2;
}
```

**Wokwi Platform Address:** https://wokwi.com/projects/380115860873524225

**Data Simulation and Thingspeak Communication:**

```
#include <ThingSpeak.h>

// Variables for data and ThingSpeak configuration
const unsigned long Channel_ID = 1234567; // Your ThingSpeak Channel ID
const char* myWriteAPIKey = "ABCDE012345"; // Your ThingSpeak Write API Key
const int Field_Number_1 = 1;

int value_1 = 0;

void setup() {
  Serial.begin(115200); // For debugging in the simulation
}

void loop() {
  // Simulate data retrieval (you can use Serial input for data simulation)
  if (Serial.available() > 0) {
    value_1 = Serial.parseInt();
    Serial.print("Value received: ");
    Serial.println(value_1);
  }

  // Simulate sending data to ThingSpeak
  Serial.print("Sending data to ThingSpeak - Channel ID: ");
  Serial.print(Channel  ID);
  Serial.print(", Field: ");
  Serial.print(Field_Number_1);
  Serial.print(", Value: ");
  Serial.println(value_1);

  // Delay for simulation purposes
  delay(15000);
}
```

**Real-time Parking Availability in Flutter:**

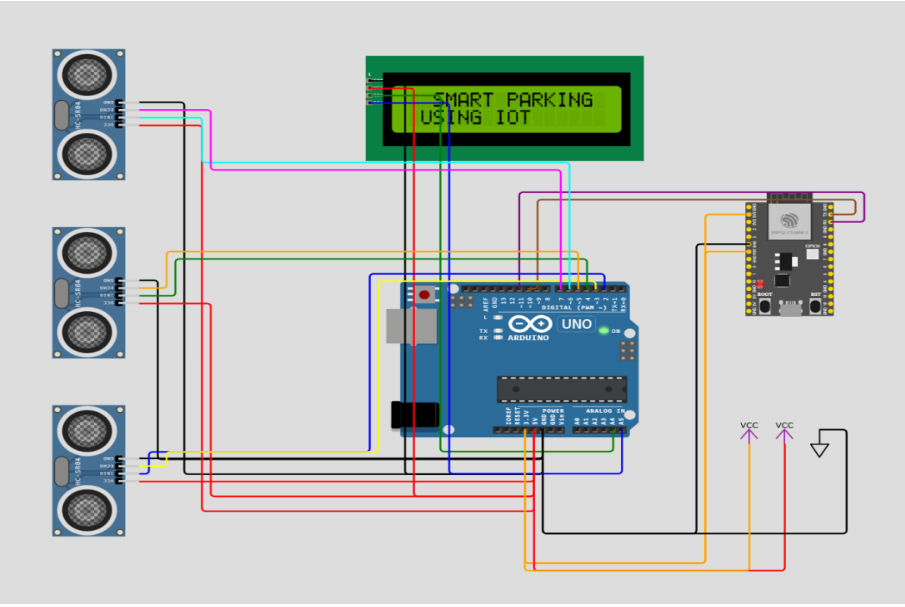
```
import 'package:flutter/material.dart';
import 'package:http/http.dart' as http;
import 'dart:convert';
```

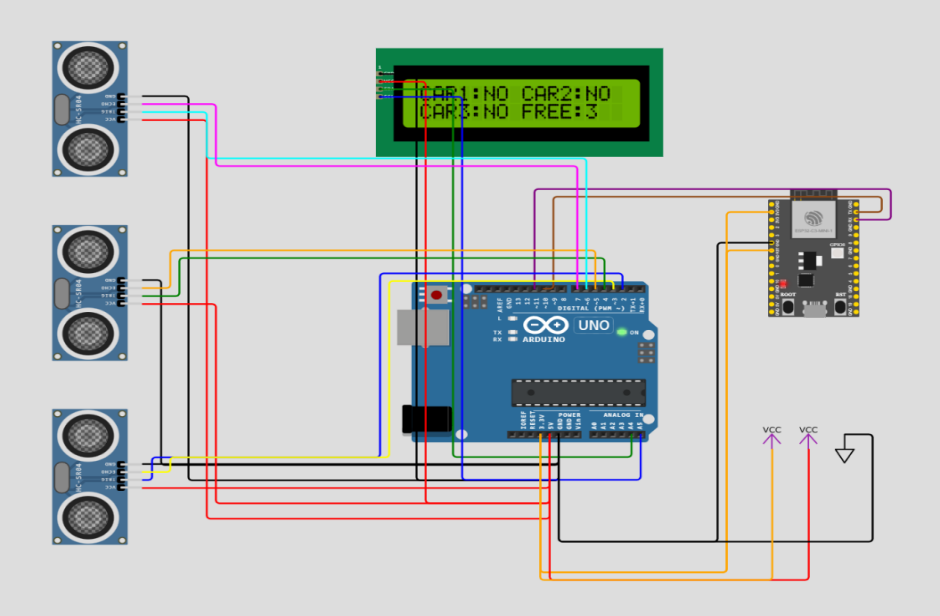
```
void main() {
  runApp(ThingSpeakApp());
}

class ThingSpeakApp extends StatefulWidget {
  @override
  _ThingSpeakAppState createState() => _ThingSpeakAppState();
}

class _ThingSpeakAppState extends State<ThingSpeakApp> {
  String value = 'Loading...';

  Future<void> fetchData() async {
    try {
      final response = await http.get(
        Uri.parse('https://api.thingspeak.com/channels/ Thingspeak_Channel_ID
/feeds.json?api_key=Your_API_key&results=1'),
      );

      if (response.statusCode == 200) {
        Map<String, dynamic> data = json.decode(response.body);
        setState(() {
          value = data['feeds'][0]['field1'].toString();
        });
      } else {
        setState(() {
          value = 'Error';
        });
      }
    } catch (e) {
      setState(() {
        value = 'Error';
      });
    }
  }

  @override
  void initState() {
    super.initState();
    fetchData();
  }
```

```
  @override
  Widget build(BuildContext context) {
    return MaterialApp(
      home: Scaffold(
        appBar: AppBar(
          title: const Text('ThingSpeak App'),
        ),
        body: Center(
          child: Text('Value from ThingSpeak: $value'),
        ),
      ),
    );
  }
}
```

**Sample Output:**

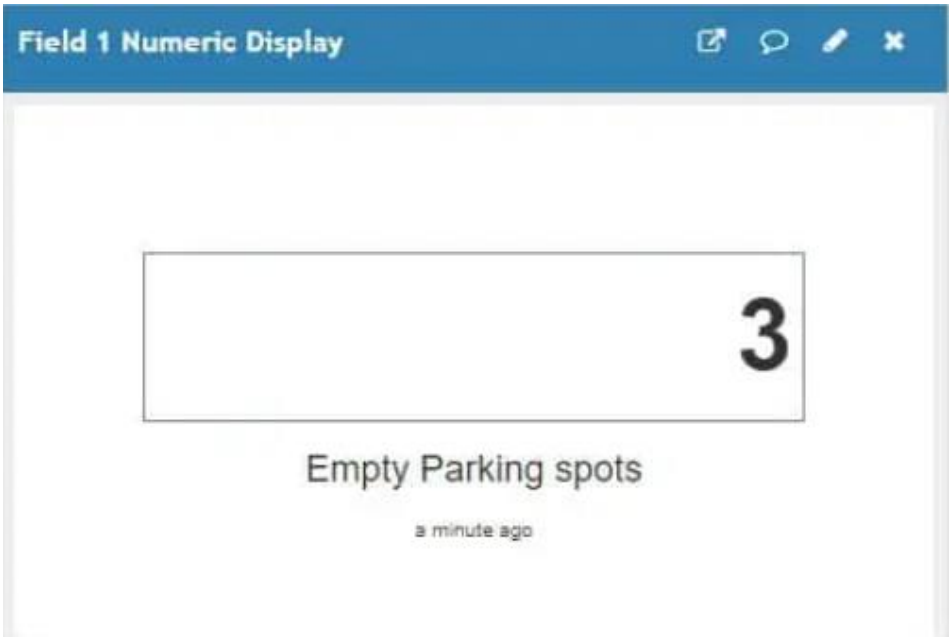
**Wokwi View:**






**Thingspeak Private View:**

**ThingSpeak Public View:**




**Overview of the Project:**

**Ultrasonic Sensors :** Sensors detect the presence of vehicles in parking slots using sound waves.

**Sensor Configuratio :** Each sensor has pins for power, trigger, and echo signals.

**Sound wave Emission :** Ultrasonic sensors emit sound waves towards vehicles in the parking slots.

**Reflection Detection :** If a vehicle is present, sensors receive sound wave reflections.

**Distance Calculation :** The time taken for sound waves to travel and return is used to calculate the distance to the vehicle.

**Data Transmission :** Occupancy status of parking slots is collected and recorded.

**Local Display :** The occupancy status is displayed locally on an LCD.

**Total vacant spots :** The system calculates the number of vacant parking slots.

**Data to the Cloud :** The occupancy data is sent to a cloud server, ThingSpeak.

**Public Access :** Users can check parking availability remotely through a URL or QR code.

**Cloud Visualization :** Parking data is accessible via ThingSpeak for public viewing.

**Continuous Monitoring :** The system updates occupancy status in real-time, providing accurate information for users.

**Conclusion:**

In the Smart Parking System project, we have successfully designed and implemented an IoT-based solution that revolutionizes the way vehicle owners find and access parking spaces. This system addresses the common issue of searching for available parking spots by providing real-time information via the internet.

**Objective Acheivement :** The project effectively fulfills its core objective of helping drivers identify vacant parking spaces before reaching their intended destination.

**Iot Technology :** The system leverages Internet of Things (IoT) technology, using a network of sensors, microcontrollers, and Wi-Fi connectivity to monitor and transmit parking spot data.

**User Convenience :** Vehicle owners can conveniently check parking spot availability using their smartphones, tablets, or computers from anywhere in the world.

**Electric Vehicle Support :** The smart parking system is especially useful for electric vehicle charging stations, enabling motorists to plan their journeys based on available charging slots.

**Stress Reduction :** By reducing the stress and time wasted in searching for parking spaces, this system contributes to a smoother and more convenient parking experience for drivers, especially those in urgent situations.

The project represents a step toward more efficient and user-friendly parking solutions. It also demonstrates the power of IoT and cloud-based platforms in enhancing everyday experiences. With the Smart Parking System, we have paved the way for smarter, technology-driven urban mobility solutions.