LAB Final Assignment Report

Course code: CSE 4418

Course Title: Internet of Things (IoT) LAB

Section: 1

Semester: Summer 2024

Submitted to:

Shakib Mahmud Dipto

Department of Computer Science and Engineering

University of Liberal Arts Bangladesh

Submitted by:

Fahim Sadik

213014027

Submission Date: 12-09-2024

1. **Requirement analysis:**

Objective: A small parking lot has a single parking space equipped with a Sonar sensor and an ESP32 microcontroller. When a car parks in the space, the Sonar sensor detects its presence and changes the status of the parking space from “Available” to “Occupied.” The system should indicate this status using an LED light:

1.1 Key Requirements:

- ■ Detect whether a parking space is occupied or available using a Sonar sensor.
- ■ Provide real-time status through LED indicators:
- ■ Green LED: Parking space available.
- ■ Red LED: Parking space occupied.
- ■ Distance measurement
- ■ Parking space log

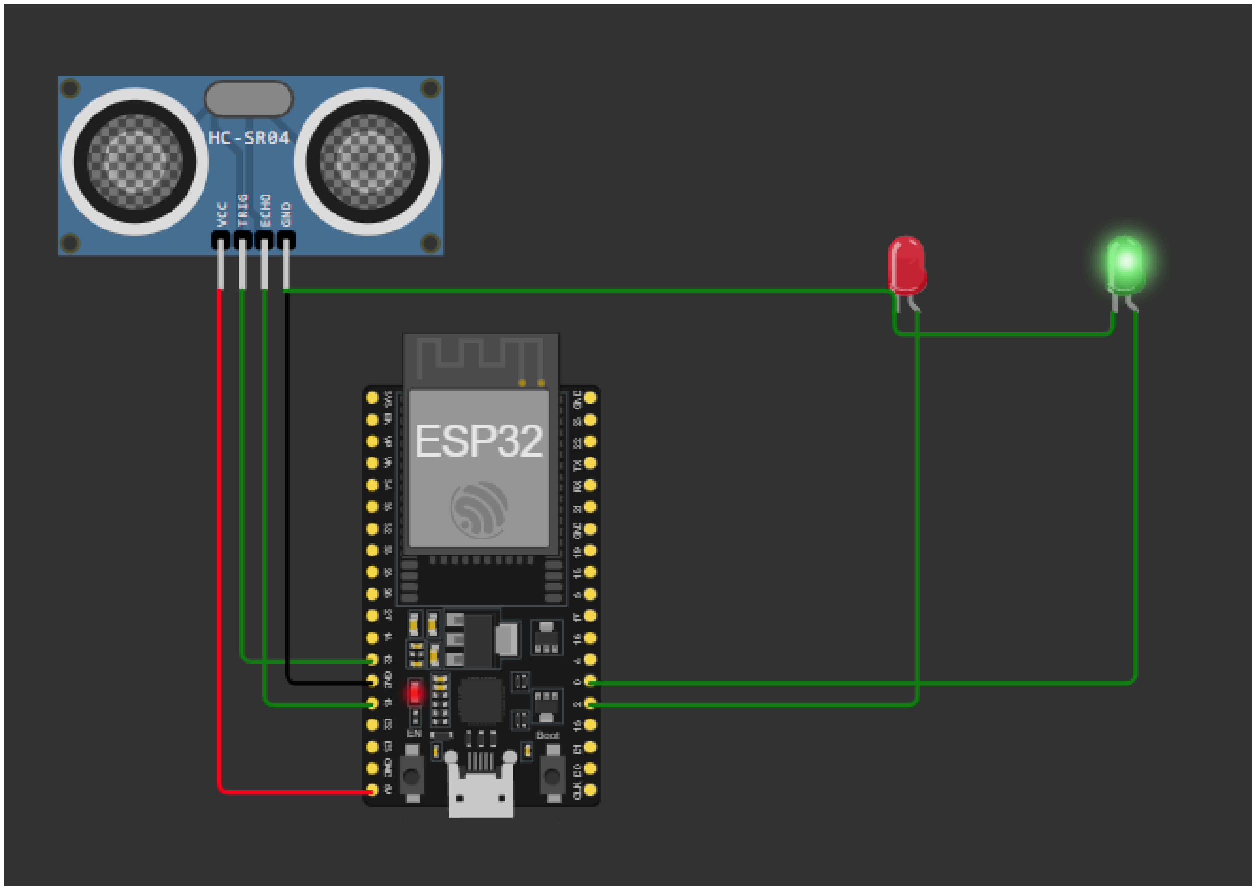
1.2 Components:
- Esp32 Microcontroller
- Sonar Sensor (HC-SR04)
- Green LED
- Red LED
- Resistors (330 Ohms)
- Wires

1.3 Used Sites:
- Wokwi.com for Arduino Simulation
- Thingspeak.com for cloud integration

2. **Circuit Design:**

2.1 Connections:
- HC-SR04 Sonar Sensor connected to the ESP32:
  - Connect Trigpin to esp pin 12
  - Connect Echopin to esp pin 13
  - Connect GND to esp GND
  - Connect 5v to esp VCC/5V
- LEDS to ESP32
  - Connect Red LED positive to esp pin 2
  - Connect Green LED positive to esp pin 0
  - Connect all GND to esp GND

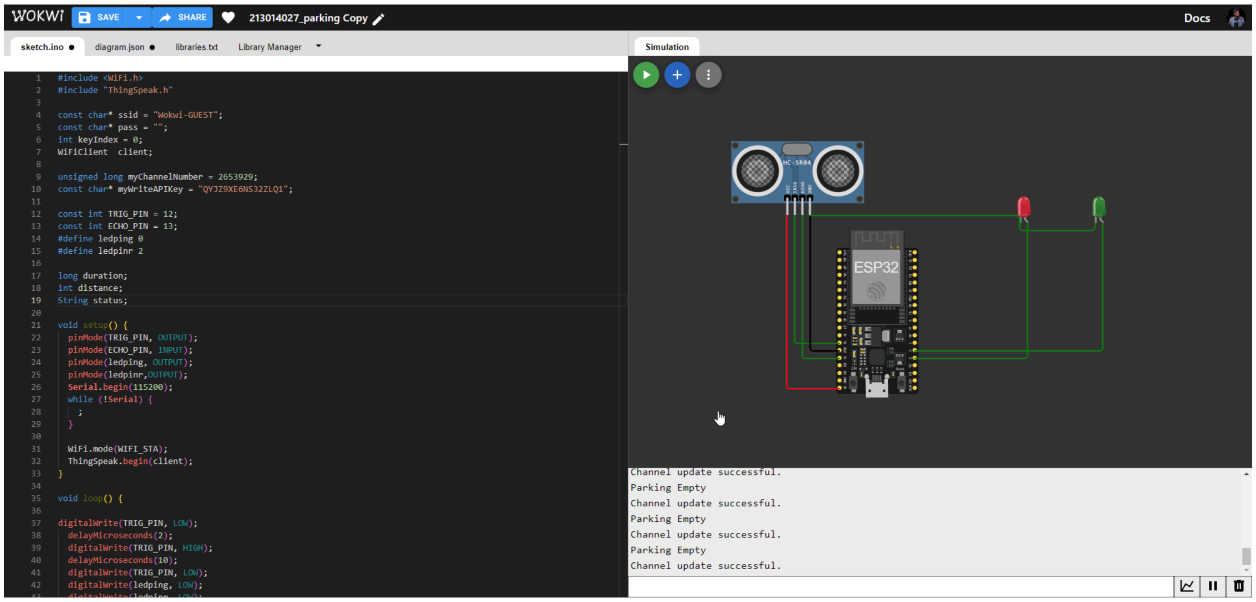
2.2 Circuit visualization:



3. Implementation:

3.1 Code Implementation:

I’ve used Worki.com for ESP32 simulation.



```
#include <WiFi.h>
#include "ThingSpeak.h"

const char* ssid = "Wokwi-GUEST";
const char* pass = "";
int keyIndex = 0;
WiFiClient  client;

unsigned long myChannelNumber = 2653929;
const char* myWriteAPIKey = "QYJZ9XE6NS32ZLQ1";

const int TRIG_PIN = 12;
const int ECHO_PIN = 13;
#define ledping 0
#define ledpinr 2

long duration;
int distance;
String status;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(ledping, OUTPUT);
  pinMode(ledpinr,OUTPUT);
  Serial.begin(115200);
  while (!Serial) {
    ;
  }

  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);
}

void loop() {

digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  digitalWrite(ledping, LOW);
```

```
Channel update successful.
Parking Empty
Channel update successful.
Parking Empty
Channel update successful.
Parking Empty
Channel update successful.
```

CODE:

```
#include <WiFi.h>
#include "ThingSpeak.h" // always include thingspeak header file after other header files and custom macros

const char* ssid = "Wokwi-GUEST"; // write your "wifi name"
const char* pass = ""; // write your "wifi password"
int keyIndex = 0;            // your network key Index number (needed only for WEP)
WiFiClient  client;

unsigned long myChannelNumber = 2653929;
const char* myWriteAPIKey = "QYJZ9XE6NS32ZLQ1";

const int TRIG_PIN = 12;
const int ECHO_PIN = 13;
#define ledping 0
#define ledpinr 2

long duration;
int distance;
String status;

void setup() {
  pinMode(TRIG_PIN, OUTPUT);
  pinMode(ECHO_PIN, INPUT);
  pinMode(ledping, OUTPUT);
  pinMode(ledpinr,OUTPUT);
  Serial.begin(115200);  //Initialize serial
  while (!Serial) {
    ; // wait for serial port to connect. Needed for Leonardo native USB port only
  }

  WiFi.mode(WIFI_STA);
  ThingSpeak.begin(client);  // Initialize ThingSpeak
}

void loop() {
```

```
digitalWrite(TRIG_PIN, LOW);
  delayMicroseconds(2);
  digitalWrite(TRIG_PIN, HIGH);
  delayMicroseconds(10);
  digitalWrite(TRIG_PIN, LOW);
  digitalWrite(ledping, LOW);
  digitalWrite(ledpinr, LOW);
   duration = pulseIn(ECHO_PIN, HIGH);

  distance = duration * 0.0344  / 2;

  // Connect or reconnect to WiFi
  if(WiFi.status() != WL_CONNECTED){
    Serial.print("Attempting to connect to SSID: ");
    while(WiFi.status() != WL_CONNECTED){
      WiFi.begin(ssid, pass); // Connect to WPA/WPA2 network. Change this line if using open or WEP
network
      Serial.print(".");
      delay(5000);
    }
    Serial.println("\nConnected.");
  }
if (distance <200 ) {
    status = "Parking Occupied";
    Serial.println("Parking Occupied");
    digitalWrite(ledpinr, HIGH);
    digitalWrite(ledping, LOW);
  }

  else {
    status = "Parking Available";
    Serial.println("Parking Available");
    digitalWrite(ledping, HIGH);
    digitalWrite(ledpinr, LOW);
    }

  ThingSpeak.setField(1, distance);  // Send distance to field 1
  ThingSpeak.setField(2, status);    // Send parking status to field 2
```

```
 // Update the channel
 int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);

 if (x == 200) {
   Serial.println("Channel update successful.");
 } else {
   Serial.println("Problem updating channel. HTTP error code " + String(x));
 }

 delay(20000);
}
```

3.2 Cloud Implementation:

- First I create an account into thingspeak
- Then I log into that account.
- From the upper menu I select new channel
- I select field 1 , field 2 and save the changes
- Now create 3 widgets.
- 1. Distance in the numeric display
- 2. Parking available in the Lamp Indicator(Green Lamp)
- 3. Parking occupied in the lamp indicator (Red Lamp)
- Now we take the channel id and put it into the code at myChannelnumber
- And we take the API write key and put it into the code at myWriteAPIKey
- Now we can watch live distance and when the distance is lower than 200cm the red light will show indicating parking is occupied
- When the distance is higher than 200cm this will turn on green light indicating parking is empty hence parking is available.
- In the end in the code we sorted another field into string mode indicating the “parking occupied” and Parking available to keep a log naming them.
- Finally, we can export the log data from the export section of the thingspeak channel.

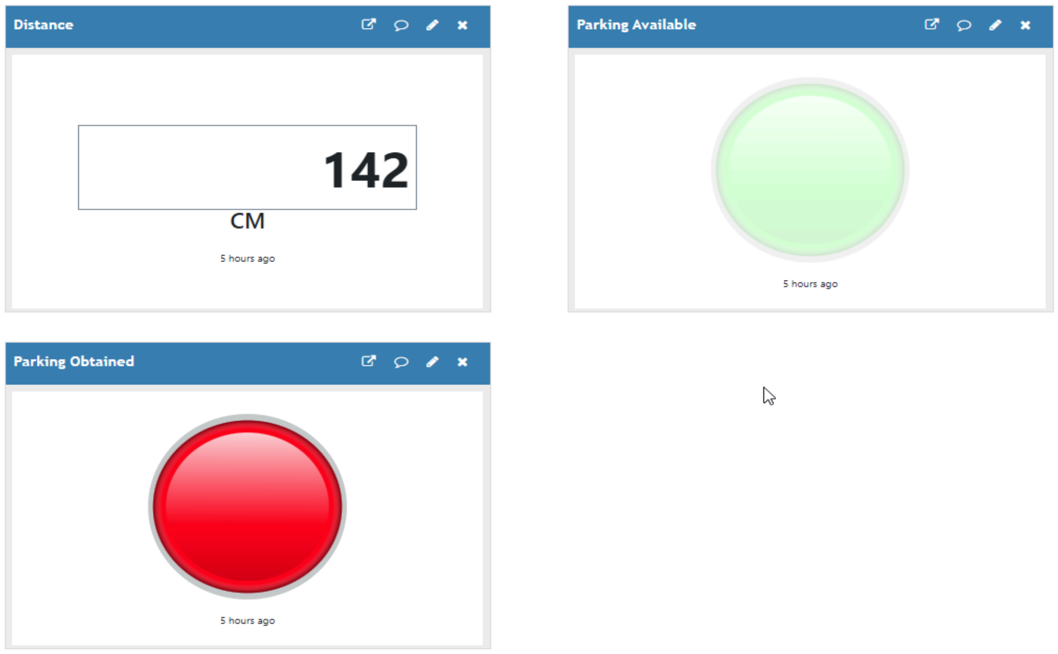


FIGURE: When there is a car this will show the distance and the red light for the car and log it into the csv file.

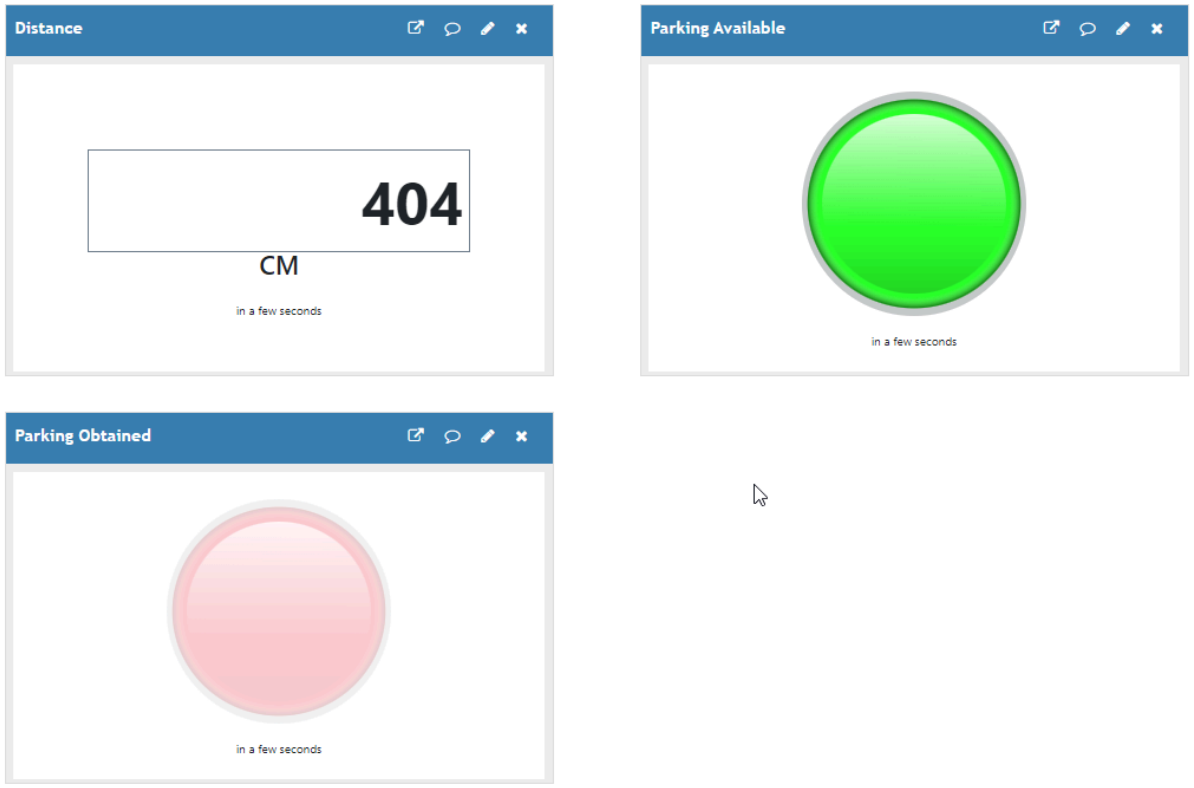
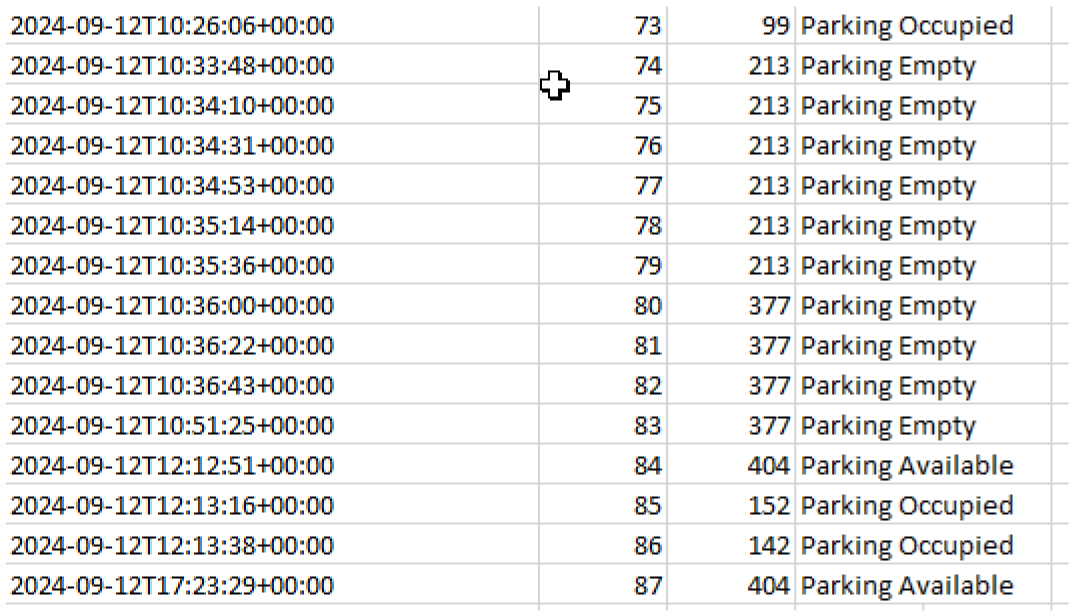


FIGURE: When there is no car it will show the distance and the green light and log it into the csv.



2024-09-12T10:26:06+00:00	73	99	Parking Occupied
2024-09-12T10:33:48+00:00	74	213	Parking Empty
2024-09-12T10:34:10+00:00	75	213	Parking Empty
2024-09-12T10:34:31+00:00	76	213	Parking Empty
2024-09-12T10:34:53+00:00	77	213	Parking Empty
2024-09-12T10:35:14+00:00	78	213	Parking Empty
2024-09-12T10:35:36+00:00	79	213	Parking Empty
2024-09-12T10:36:00+00:00	80	377	Parking Empty
2024-09-12T10:36:22+00:00	81	377	Parking Empty
2024-09-12T10:36:43+00:00	82	377	Parking Empty
2024-09-12T10:51:25+00:00	83	377	Parking Empty
2024-09-12T12:12:51+00:00	84	404	Parking Available
2024-09-12T12:13:16+00:00	85	152	Parking Occupied
2024-09-12T12:13:38+00:00	86	142	Parking Occupied
2024-09-12T17:23:29+00:00	87	404	Parking Available

FIGURE: the CSV file log

4. Explanation of your approach to handling multiple parking spaces:

If there were multiple parking slots, I would use multiple sonar sensors and multiple lights indicating and naming them according to their position. Also, for the cloud, we would need multiple fields and statuses to show them on there. In the end we could get Excel file log from all of them in a single file. And obviously we could see the live time feed there in the Thingspeak channel dashboard or the Thingspeak mobile application.