Index

Index of images

Introduction

The project focuses on using ultrasonic sensors (or proximity sensors) placed in each parking spot to detect whether a space is occupied or available. The information captured by these sensors is sent to a microcontroller, which processes the data and displays it on a graphical interface, updating the status in real-time. Additionally, the system is designed to potentially send this information to a mobile application or website, allowing users to check for available spaces before arriving at the parking lot.

As in the modern world everything is going automatic, we have built a system which will automatically sense the entry and exit of cars through the gate and then display the number of cars in the parking lot. Check-ins and check-outs will be handled in a fast manner without having to stop the cars so that traffic jam problems will be avoided during these processes.

This developed technology can be used in any urban areas where the car parking is most. Some of the heavy traffic places where this project can be installed and used are shopping malls, hospitals, airports, cinema halls, apartments, etc. The cost of land has grown exponentially in cities, so it becomes essential that the parking solution requires the least possible space and can accommodate the maximum number of vehicles.

Main Objective

The main objective of this project is to develop a smart parking system using sensors and microcontrollers to monitor and manage parking space availability in real-time. This system aims to optimize space utilization, reduce parking search times, and enhance user convenience by providing accurate, up-to-date information on available spots through a digital interface, ultimately contributing to efficient urban mobility solutions within the framework of smart cities.

Specific Objectives

1. **Implement Real-Time Monitoring**: Develop a system using sensors and microcontrollers to detect and display the availability of each parking spot in real-time.

2. **Optimize Space Utilization**: Design the system to efficiently manage the allocation of parking spaces, reducing idle or underutilized spaces and maximizing capacity.

3. **Improve User Experience**: Create a user-friendly interface that provides clear information on available parking spaces, accessible both on-site and remotely through a mobile app or website.

4. **Reduce Parking Search Times**: Minimize the time users spend searching for parking by guiding them directly to available spaces, thus decreasing congestion and improving traffic flow.

5. **Enable Scalable Design**: Build a flexible system that can be scaled to support a larger number of parking spaces or integrated with other smart city infrastructure in the future.

6. **Enhance System Reliability**: Ensure accurate data collection and transmission from sensors to the display interface, reducing potential errors and system downtime.
7. **Promote Sustainable Urban Mobility**: Contribute to sustainable city initiatives by reducing vehicle emissions related to searching for parking, aligning with eco-friendly smart city goals.

Theoretical framework

Feng Yuan Wang and Yi Liu presented a paper in 2017 titled ‘Mechanical Parking System’ that consisted of a rotary mechanism that allowed all the cars to travel in rotary motion. Cars were loaded and unloaded with the rotary motion of all cars. This system was preferable for 8 to 12 cars. Advantages were that it was easy to operate and easy parking of the vehicle was achieved. Limitations were that all cars need to be rotated to access one car, high initial cost and high maintenance and complicated structure.

Robin Grodi, Danda B. Rawat and Fernando RiosGutierrez published a paper in 2017 titled ‘Smart Parking Occupancy Monitoring and Visualization system for smart cities’. Robin Grodi, Danda B. Rawat and Fernando Rios-Gutierrez had done work on how the vehicle would occupy the particular allocated place. RFID sensors detected the presence of a vehicle or other objects in the allocated slot. Once a vehicle was detected, the system needed a way to notify drivers or a parking spot being occupied. The disadvantage was that the parking place would be detected only to the nearby places and there was no GPS sensor to search the parking slots from afar.

Dharmini Kanteti, D V S Srikar and T K Ramesh published paper titled ‘Smart Parking System for Commercial Stretch in Cities’ in 2017[3] . They developed a Smart Parking System. In their model pre registered IP cameras would capture the vehicle registration number and then they would proceed without interruptions. Their details like parking time estimate, their place of visit etc. would be recorded. For pre-registered users, the amount would be deducted from the e-wallet and then the users would be notified. A similar pricing system would be followed for new users but the payment would be offline. The disadvantages were that the system could serve all the parking requests but beyond the number of 80 it couldn’t accommodate more cars since the parking became full.

Problem Statement

Improper parking often leads to vehicles occupying inappropriate spaces, which can result in damage between them. Insufficient parking spaces cause traffic congestion and frustration for drivers. This leads to situations like tailgating, where one vehicle blindly follows another without proper identification.

Frequently, gates are left open, allowing all kinds of vehicles to enter the premises. Security guards are ineffective in controlling vehicle access due to the lack of a proper identification system. The gate-opening process is manual and depends on the availability of the guard. In a large parking lot with hundreds of available spaces, finding an empty spot manually is a tedious, time-consuming, and frustrating task.

The manual parking system involves various tasks such as issuing tokens, recording check-in and check-out times, calculating fares, and collecting payments. As land costs have risen dramatically in cities, it is essential that parking solutions require the least possible space while accommodating the maximum number of vehicles. On average, a person spends 10 to 15% of their travel time searching for a suitable parking spot in metropolitan cities.

Characteristics and Used Components

Arduino UNO:
Arduino UNO is a low-cost, flexible, and easy-to-use programmable open-source microcontroller board that can be integrated into a variety of electronic projects. Arduino UNO features AVR microcontroller Atmega328, 6 analog input pins, and 14 digitalI/O pins out of which 6 are used as PWM output. This board contains a USB interface i.e.,USB cable is used to connect the board with the computer and Arduino IDE (Integrated Development Environment) software is used to program the board.

- **Main Controller**: The Arduino UNO serves as the central processing unit, coordinating data from sensors, controlling outputs, and executing programmed logic to **manage** the **smart parking system**.
- **Programming Flexibility**: This microcontroller supports easy programming and integration with a wide variety of sensors and modules, allowing for a modular and adaptable system design.

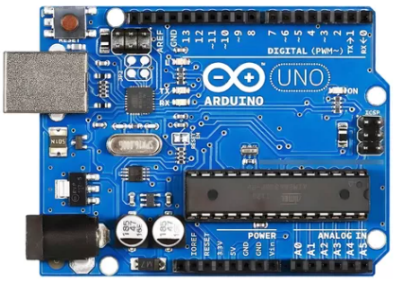


Image 1. Arduino UNO component

6 IR Sensors:
An infrared sensor is an electronic device that emits in order to sense some aspects of the surroundings. An IR sensor can measure the heat of an object as well as detect the motion. These types of sensors measure only infrared radiation, rather than emitting it that is called a passive IR sensor. The emitter is simply an IR LED (Light Emitting Diode) and the detector is simply an IR photodiode that is sensitive to IR light of the same wavelength as that emitted by the IR LED. When IR light falls on the photodiode, the resistances and the output voltages will change in proportion to the magnitude of the IR light received.

- **Parking Spot Detection**: IR sensors detect if a vehicle is **parked** in a specific spot. Each parking space is equipped with one IR sensor to monitor occupancy status.
- **Low Power Consumption**: These sensors are energy-efficient, contributing to the project's goal of building a low-power system suitable for sustained use.

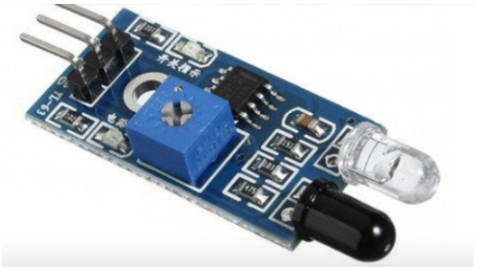


Image 2. IR sensor

2 Ultrasonic Sensors:
Ultrasonic sensors, along with stereo vision, are used on UAVs to obtain the distance of the vehicle from the ground. Ultrasonic sensors can then be used by a flight control system to maintain the vehicle at a specific altitude. In practice, an ultrasonic sensor sends ultrasonic pulses towards the ground and receives back their reflection as they bounce off of the ground. The sensor calculates the distance to the ground as the time lapses between the transmitted waves and the reflected waves.

- **Entrance and Exit Detection**: Ultrasonic sensors placed at the parking lot's entrance and exit detect when a vehicle enters or leaves, allowing the system to let cars enter and leave the parking automatically with a pair working with the RFID module.
- **High Accuracy**: Ultrasonic sensors provide reliable detection of larger distances than IR sensors, making them ideal for monitoring vehicle entry and exit.

Image 3. Ultrasonic sensors

Display LCD:
An electronic device that is used to display data and the message is known as LCD 16×2 display.As the name suggests, it includes 16 Columns & 2 Rows so it can display 32 characters (16×2=32) in total & every character will be made with 5×8 (40) Pixel Dots.

The 16 x 2 intelligent alphanumeric dot matrix display is capable of displaying 224 different characters and symbols. The basic working principle of LCD is passing the light from layer to layer through modules. These modules will vibrate line up their position on 90o that permits the polarized sheet to allow the light to pass through it.

- **Real-Time Display of Parking Availability**: A Display LCD is used to show the number of available parking spots, helping users make quick decisions.
- **Clear and User-Friendly Interface**: Positioned at the parking lot entrance, the display provides easy-to-read information, enhancing user experience and efficiency.
- **LCD Module:** The backlight provides the light source necessary for the display, and lets us work more efficiently with the display LCD via Arduino code.

Image 4. Display LCD

RFID Module:
An RFID or radio frequency identification system consists of two main components, a tag attached to the object to be identified, and a reader that reads the tag.

A reader consists of a radio frequency module and an antenna that generates a high frequency electromagnetic field. Whereas the tag is usually a passive device (it does not have a battery). It consists of a microchip that stores and processes information, and an antenna for receiving and transmitting a signal.

- **Access Control and Security**: An RFID module manages entry by requiring users to scan an RFID card to enter the parking lot, ensuring only **authorized** users can access the parking area.
- **Data Logging**: RFID can also log user entry and exit times, potentially enabling additional data analysis and record-keeping.

Image 5. RFID Module

Servomotor:
Servos are motors that allow you to precisely control physical movement because they generally move to a position rather than continuously rotating. They are simple to connect and control because the motor driver is built right into them.

Servos contain a small DC motor connected to the output shaft through gears. The output shaft drives a servo horn and is also linked to a potentiometer (pot).

The potentiometer provides position feedback to the error amplifier in the control unit, which compares the current position of the motor to the target position.

- **Barrier Control**: The servo motor controls a barrier gate at the entrance. When a valid RFID card is scanned, the servo opens the gate to allow entry and closes it after the vehicle passes.
- **Precise Positioning**: Servomotors allow for accurate control of the barrier, ensuring smooth operation with minimal delays.

Image 6. Servomotor component

Buzzer:
An audio signaling device like a beeper or buzzer may be electromechanical or piezoelectric or mechanical. The main function of this is to convert the signal from audio to sound. Generally, it is powered through DC voltage and used in timers, alarm devices, printers, alarms, computers, etc. Based on the various designs, it can generate different sounds like alarm, music, bell & siren.

- **Alert System**: The buzzer serves as an alert mechanism. It can sound when the parking lot is full or when an unauthorized RFID card is detected.
- **Enhanced Safety and Usability**: Audible alerts inform users of important events, such as entry confirmation or system errors, contributing to the overall user-friendliness and safety of the parking system.

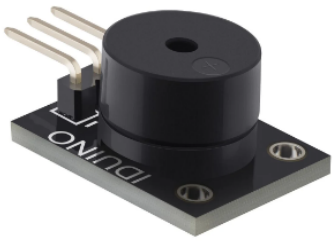


Image 7. Buzzer component

Raspberry Pico W:

The Raspberry Pi Pico W is a powerful and versatile microcontroller board developed by the Raspberry Pi Foundation. At the heart of the Pico W is the **RP2040** chip, which is designed in-house by Raspberry Pi. This chip houses a **dual-core ARM Cortex-M0+** processor, running at a clock speed of **133 MHz**, making it capable of handling a wide range of tasks efficiently. It comes equipped with **264KB of on-chip SRAM**, providing enough memory for most embedded applications, and **2MB of onboard QSPI flash storage**, which is used for storing firmware and data.

- **User Interface**: The Raspberry Pico W helps us to transfer all the information of the IR sensor to our mobile app.

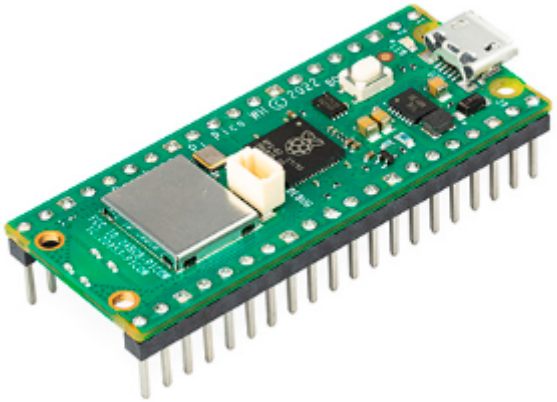


Image 8. Raspberry Pi Pico W microcontroller

Step by Step

The first thing that we did to do this project was the schematic design of the circuit that was used in the project. We decided to separate our circuits into two, due the amount of components that we used, one circuit is for the parking entrance, and the other for the parking slots, that their data is shared with a Raspberry Pi Pico W. And just one Arduino wasn't enough to combine all the components, where another factor was the power usage and needed for each component to work well.

So, the first thing that we did was working in the parking slots, because for us that was the main functionality of our project, and an extra for us was implementing a way to share the captured by the Arduino to an external application.

For this we used a Raspberry Pi Pico W microcontroller that has the capability to connect to the internet, so that was enough to create a web server in the Pico W to let us connect to it from external devices. In short, the steps to share IR sensors status to external devices was:

1. Receive signals from IR sensors with the Arduino.
2. Update state of IR sensors with Arduino.
3. Transmit managed data from Arduino to Pico W with Signal Transmission (TX and RX).
4. Create a web server in Pico W.
5. Receive data from Arduino.
6. Use WebSockets protocols to receive data from Pico W to use it in an external app.

And that was how the parking slots administration was made. The other main functionality was the parking entrance, this was made by combining sensors as ultrasonic sensors and RFID, and with the usage of actuators as buzzer, display LCD, and servomotors. And the way it works is like this:

1. For the entrance, firstable the car needs to be detected by an ultrasonic sensor that activates RFID lecture.
2. Once the RFID module detects a valid card, it opens the servomotor for the entrance.
3. For exit, it just needed to get close to the second ultrasonic sensor.
4. If an invalid card is detected, the buzzer emits an alarm sound.
5. We use a LCD display to display messages for user instructions and notifications.

Simulator Design

Parking entrance (initial design)

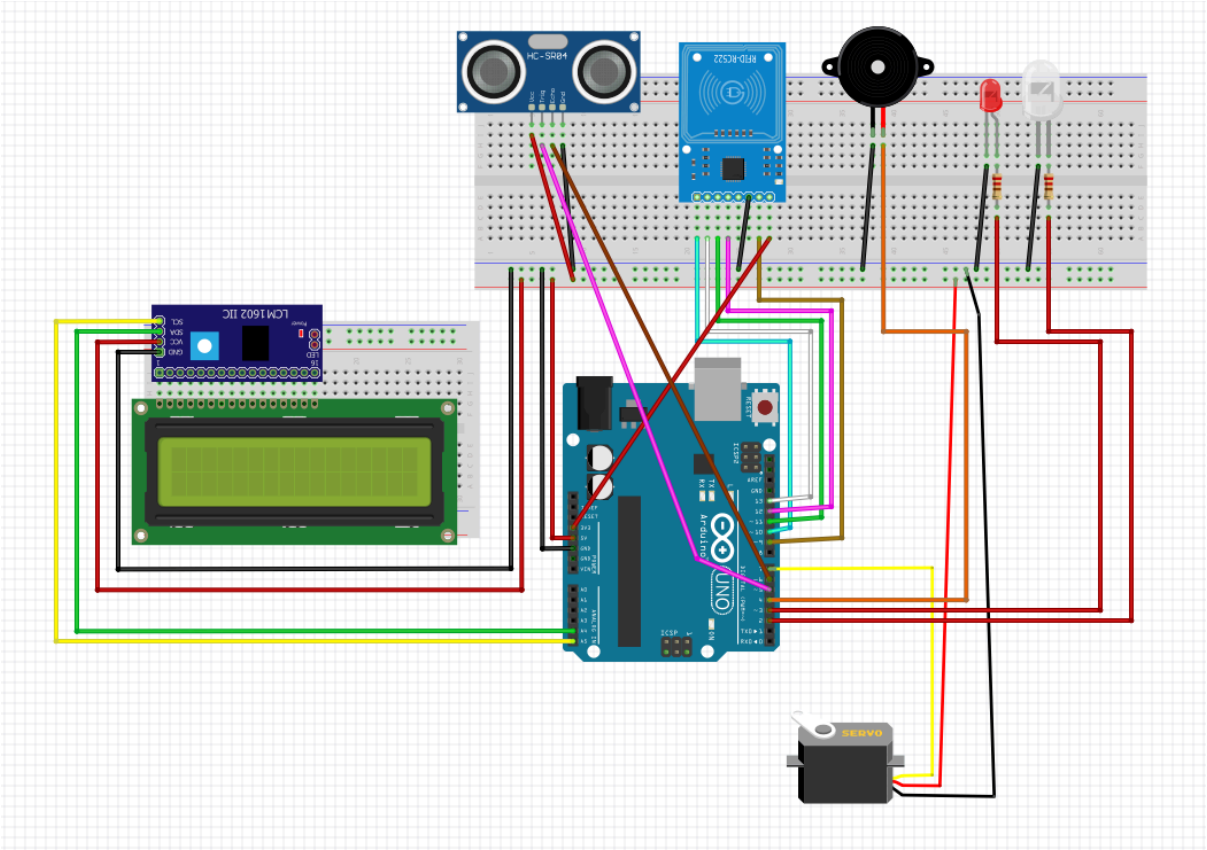


Image 9. Parking entrance simulator made in Fritzing

Parking spots

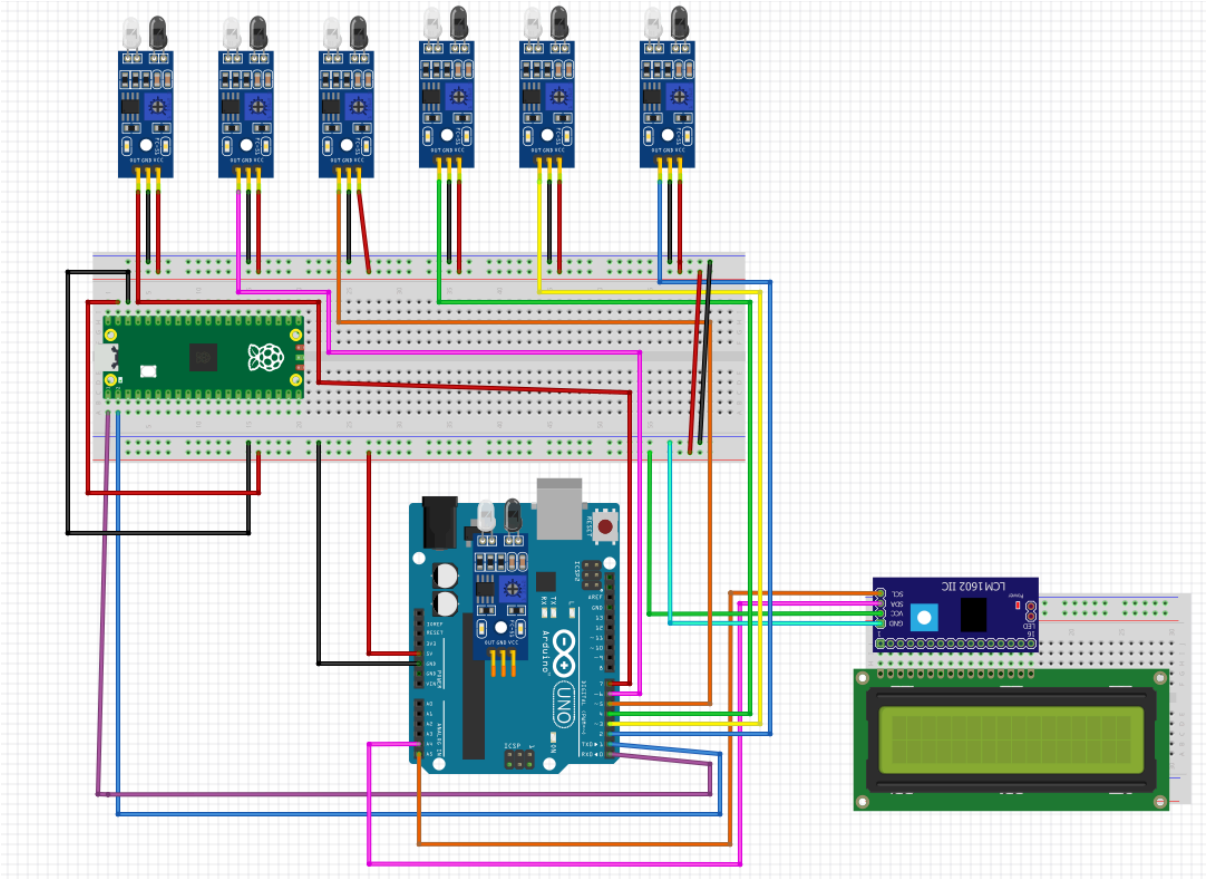


Image 10. Parking slots management simulator made in Fritzing

Utility & Usability

Utility:

Improved Parking Efficiency: The smart parking system helps optimize the usage of parking spaces, reducing congestion and better utilizing available spots.

Reduced Search Time: By providing real-time information on available spots, it minimizes the time users spend searching for parking, thereby reducing traffic and emissions caused by vehicles circling to find a space.

Enhanced User Convenience: Through a digital interface, users can quickly identify and navigate to open spots, making their parking experience smoother and less frustrating.

Urban Mobility Support: By easing parking management, the system supports the larger goals of urban mobility and the development of smart cities, making it a valuable tool for modern city infrastructure.

Data-Driven Insights: The system can collect data on parking usage patterns, peak times, and demand, providing insights for urban planners to make informed decisions regarding future infrastructure needs.

Usability:

Real-Time Availability: Users can view available parking spots in real-time, helping them make quick decisions and avoid wasted time and fuel.

User-Friendly Interface: The digital interface should be intuitive, allowing users to check availability easily from a mobile app or web platform.

Guidance Features: The system could integrate navigation assistance to guide users directly to open spots, further reducing search time.

Mobile and Remote Accessibility: Users can check availability and even reserve spaces (if applicable) remotely, which adds convenience.

System Reliability and Accuracy: The use of sensors and microcontrollers ensures that the data provided is accurate and up-to-date, fostering trust in the system and encouraging its use.

Code

IR Sensors Parking Slots Arduino Code

```
#include <Servo.h> //includes the servo library
#include <Wire.h>
// #include <LiquidCrystal_I2C.h>
#include <hd44780.h>
#include <hd44780ioClass/hd44780_I2Cexp.h> //I2C expander i/o
class header

// LiquidCrystal_I2C lcd(0x27, 2, 1, 0, 4, 5, 6, 7, 3, POSITIVE);
// LiquidCrystal_I2C lcd(0x27,16,2);
hd44780_I2Cexp lcd(0x27);

Servo myservo;

// #define ir_enter 2
// #define ir_back  4

#define ir_car1 2
#define ir_car2 3
#define ir_car3 4
#define ir_car4 5
#define ir_car5 6
#define ir_car6 7

int S1=0, S2=0, S3=0, S4=0, S5=0, S6=0;
int slot = 6;

int parkingSlots[6];

void setup(){
  Serial.begin(9600);

  pinMode(ir_car1, INPUT);
  pinMode(ir_car2, INPUT);
  pinMode(ir_car3, INPUT);
  pinMode(ir_car4, INPUT);
  pinMode(ir_car5, INPUT);
  pinMode(ir_car6, INPUT);

  lcd.begin(16, 2);
  lcd.setCursor (1,0);
  lcd.print("Car parking");
  lcd.setCursor (1,1);
  lcd.print(" System ");
  delay (2000);
```

```
  lcd.clear();

  Read_Sensor();

  int total = S1+S2+S3+S4+S5+S6;
  slot = slot-total;
}

void loop(){

  Read_Sensor();

  lcd.setCursor (0,0);
  if(S1==1){lcd.print("1:F ");}
      else{lcd.print("1:E");}

  lcd.setCursor (6,0);
  if(S2==1){lcd.print("2:F ");}
      else{lcd.print("2:E ");}

  lcd.setCursor (11,0);
  if(S3==1){lcd.print("3:F ");}
      else{lcd.print("3:E ");}


  lcd.setCursor (0,1);
  if(S4==1){lcd.print("4:F ");}
      else{lcd.print("4:E ");}

  lcd.setCursor (6,1);
  if(S5==1){lcd.print("5:F ");}
      else{lcd.print("5:E ");}

  lcd.setCursor (11,1);
  if(S6==1){lcd.print("6:F ");}
      else{lcd.print("6:E ");}

  parkingSlots[0] = S1;
  parkingSlots[1] = S2;
  parkingSlots[2] = S3;
  parkingSlots[3] = S4;
  parkingSlots[4] = S5;
  parkingSlots[5] = S6;

  for (int i = 0; i < 6; i++) {
    Serial.print(String(parkingSlots[i]));
    if (i < 2) {
      Serial.print(",");
```

```
    }
  }

  Serial.println();

  delay(200);

}

void Read_Sensor(){
  S1=0, S2=0, S3=0, S4=0, S5=0, S6=0;

  if(digitalRead(ir_car1) == 0){S1=1;}
  if(digitalRead(ir_car2) == 0){S2=1;}
  if(digitalRead(ir_car3) == 0){S3=1;}
  if(digitalRead(ir_car4) == 0){S4=1;}
  if(digitalRead(ir_car5) == 0){S5=1;}
  if(digitalRead(ir_car6) == 0){S6=1;}
}
```

Parking Entrance Arduino Code}

```
#include <LiquidCrystal_I2C.h>

//RFID LIBRAIES
#include <SPI.h>
#include <MFRC522.h>
#include <Wire.h>

// #include <LiquidCrystal_I2C.h>//inlude lcd library
#include <Servo.h>//include servo library
#include <HCSR04.h>

// #include <hd44780.h>
// #include <hd44780ioClass/hd44780_I2Cexp.h> //I2C expander
i/o class header

#define POSITIVE 0

//ultrasonic connection

#define TRIG_PIN 5
#define ECHO_PIN 6

#define TRIG_PIN2 2
#define ECHO_PIN2 3

//rfid connection
#define SS_PIN 10
#define RST_PIN 9

// Timing constants
const unsigned long SERVO_OPEN_TIME = 5000;    // Time to keep
servo open
const unsigned long LCD_REFRESH_TIME = 1000;   // LCD refresh
interval
const unsigned long DISPLAY_MESSAGE_TIME = 3000; // Message
display duration
const unsigned long BUZZER_BEEP_TIME = 1000;    // Buzzer beep
duration

// Global variables for timing
unsigned long lastServo1Time = 0;
```

```
unsigned long lastServo2Time = 0;
unsigned long lastLcdRefresh = 0;
unsigned long messageStartTime = 0;
unsigned long buzzerStartTime = 0;

// State variables
boolean RFIDMode = false;
boolean servo1Opening = false;
boolean servo2Opening = false;
boolean buzzerActive = false;
String currentMessage = "";
int currentMessageLine = 0;

MFRC522 mfrc522(SS_PIN, RST_PIN);

LiquidCrystal_I2C lcd (0x27, 2, 1, 0, 4, 5, 6, 7);
// hd44780_I2Cexp lcd(0x27);

UltraSonicDistanceSensor distanceSensor(TRIG_PIN, ECHO_PIN);
UltraSonicDistanceSensor distanceSensor2(TRIG_PIN2,
ECHO_PIN2);

Servo myservo;
Servo myservo2;

int buzzer=8;

String tagUID="23 CA C5 26";
//another tag UID  i use aa a tag
//String tagUIDddd="DD B5 A3 59";

void setup()
{
  lcd.setBacklightPin(3, POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.begin(16, 2);

  SPI.begin();      // Initiate  SPI bus
  mfrc522.PCD_Init();   // Initiate MFRC522

  Serial.begin(9600);
  myservo.attach(4);
  myservo2.attach(7);
```

```
  pinMode(buzzer,OUTPUT);

  pinMode(TRIG_PIN, OUTPUT); // Sets the trigPin as an Output
  pinMode(ECHO_PIN, INPUT); // Sets the echoPin as an Input

  pinMode(TRIG_PIN2, OUTPUT); // Sets the trigPin as an Output
  pinMode(ECHO_PIN2, INPUT); // Sets the echoPin as an Input

  lcd.setCursor (1,0);
  lcd.print("Car parking");
  lcd.setCursor (1,1);
  lcd.print(" System ");
  delay (2000);
  lcd.clear();

  myservo.write(0);
  myservo2.write(0);
}

void displayTemporaryMessage(String line1, String line2) {
  currentMessage = line1;
  lcd.clear();
  lcd.setCursor(0, 0);
  lcd.print(line1);
  lcd.setCursor(0, 1);
  lcd.print(line2);
  // messageStartTime = millis();
}

String getCardUID() {
  String content = "";
  for (byte i = 0; i < mfrc522.uid.size; i++) {
    content.concat(String(mfrc522.uid.uidByte[i] < 0x10 ? " 0"
: " "));
    content.concat(String(mfrc522.uid.uidByte[i], HEX));
  }
  content.toUpperCase();
  return content.substring(1);
}

void handleRFID() {
  if (!mfrc522.PICC_IsNewCardPresent() ||
!mfrc522.PICC_ReadCardSerial()) {
    return;
```

```
  }

  // Get card UID
  String content = getCardUID();

  // Print UID for debugging
  Serial.print("Card UID: ");
  Serial.println(content);

  if (content == tagUID) {
    displayTemporaryMessage("Tag Matched", "");
    servo1Opening = true;
    lastServo1Time = millis();

    RFIDMode = false;
  } else {
    displayTemporaryMessage("Wrong Tag Shown", "Access
Denied");
    buzzerActive = true;
    buzzerStartTime = millis();

    // RFIDMode = false;
  }
}

void handleServos() {
  unsigned long currentTime = millis();

  // Handle Servo 1 (RFID controlled)
  if (servo1Opening && (currentTime - lastServo1Time >=
SERVO_OPEN_TIME)) {
    myservo.write(0);
    servo1Opening = false;
    RFIDMode = false;
    displayTemporaryMessage("Door Closed", "");
    delay(3000);
    currentMessage = "";
  } else if (servo1Opening) {
    myservo.write(90);
  }

  // Handle Servo 2 (Distance controlled)
  if (servo2Opening && (currentTime - lastServo2Time >=
SERVO_OPEN_TIME)) {
```

```
    myservo2.write(0);
    servo2Opening = false;
  } else if (servo2Opening) {
    myservo2.write(90);
  }
}

void handleBuzzer() {
  if (buzzerActive) {
    if (millis() - buzzerStartTime < BUZZER_BEEP_TIME) {
      tone(buzzer, BUZZER_BEEP_TIME);
    } else if (millis() - buzzerStartTime < BUZZER_BEEP_TIME *
2) {
      noTone(buzzer);
    } else {
      buzzerActive = false;
      digitalWrite(buzzer, LOW);
      currentMessage = "";
    }
  }
}

void handleDisplay() {
  if (millis() - lastLcdRefresh >= LCD_REFRESH_TIME) {
    if (currentMessage == "") {
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("   Bienvenido  ");
      lcd.setCursor(0, 1);
      lcd.print("  Get close...");
    }
    lastLcdRefresh = millis();
  }
}

void loop()
{
  float distance = distanceSensor.measureDistanceCm();
  float distance2 = distanceSensor2.measureDistanceCm();

  Serial.println(distance);

  // Handle distance-based triggers
  if (RFIDMode != true) {
```

```
    if (distance < 5 && distance >= 0) {
      RFIDMode = true;
      if (RFIDMode) {
        displayTemporaryMessage("RFID Door Lock", "Show Your
Tag");
        handleRFID();
      }
    }
  }
  else {
    handleRFID();
  }

  if (distance2 < 5 && distance2 >= 0 && !servo2Opening) {
    servo2Opening = true;
    lastServo2Time = millis();
  }

  // Handle all concurrent tasks
  handleServos();
  handleBuzzer();
  handleDisplay();

  // Small delay to prevent excessive CPU usage
  delay(10);
}
```

Raspberry Pi Pico W Code

```
import machine
from machine import Pin,UART

import time
import network

from microdot import Microdot, send_file
from microdot.websocket import with_websocket

ssid = ""
password = ''

def connect_to_wifi():
    connection = network.WLAN(network.STA_IF)
    if not connection.isconnected():
        print("Connecting to the network...")
        connection.active(True)
        connection.connect(ssid, password)
        while not connection.isconnected():
            pass
        print("Connected to IP: ", connection.ifconfig()[0])
    else:
        print("Connected to Internet")
        print("Connected to IP: ", connection.ifconfig()[0])

connect_to_wifi()

uart = UART(0, baudrate=9600, tx=Pin(0), rx=Pin(1))

html = """<!DOCTYPE html>
<html>
<head> <title>Pico W</title> </head>
<body> <h1>Pico W HTTP Server</h1>
<p>Hello, World!</p>
</body>
</html>
"""

app = Microdot()

@app.route("/")
async def index(request):
```

```
    return html, {"Content-Type": "text/html"}

@app.route("/test")
@with_websocket
async def index(request, ws):
    while True:
        try:
            data = None

            if uart.any():
                data = uart.readline()  # Read data sent from
Arduino
                time.sleep(0.2)
                print(data)  # Print to console

            else:
                data = "No data available from UART"

            await ws.send(data)

        except OSError as e:
            print('connection closed')
            pass

app.run(port=80, debug=True)
```

Photos of the process

Parking Entrance

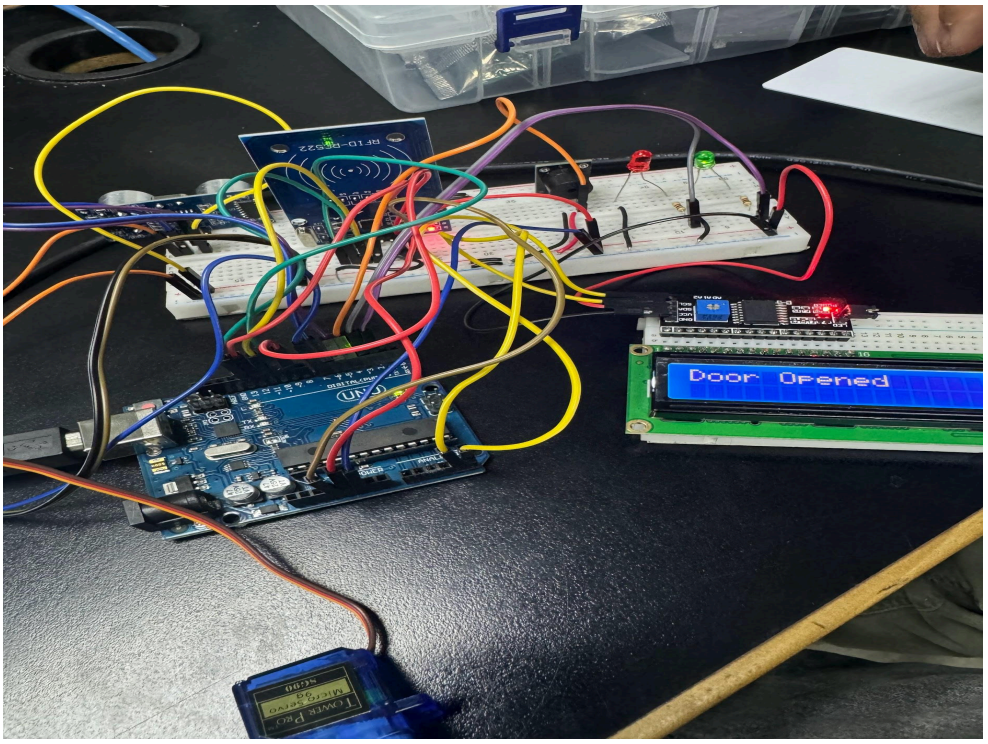


Image 11. Parking entrance circuit first implementation (Door opened)

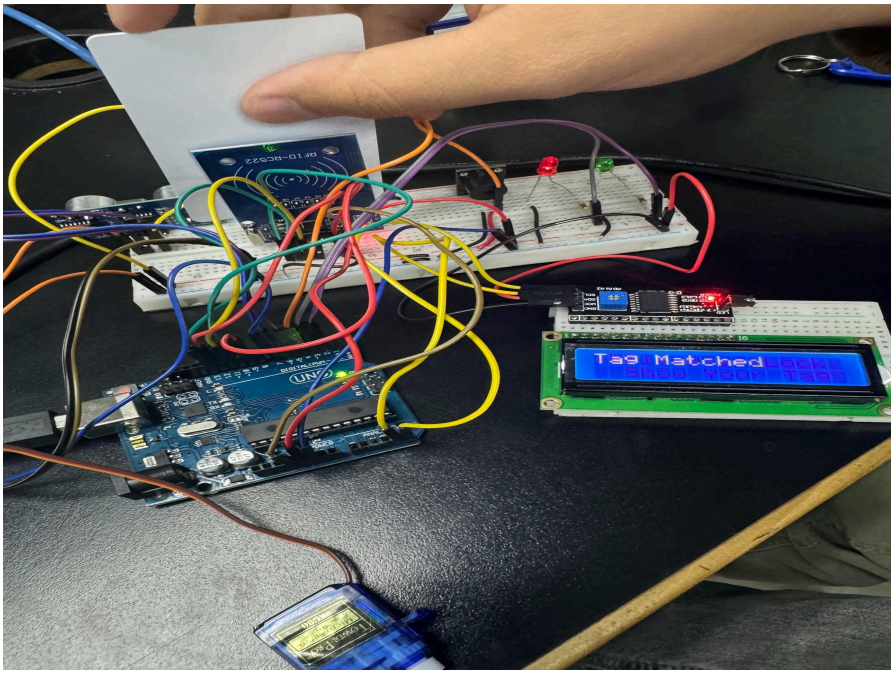


Image 12. Parking entrance circuit first implementation (RFID usage)

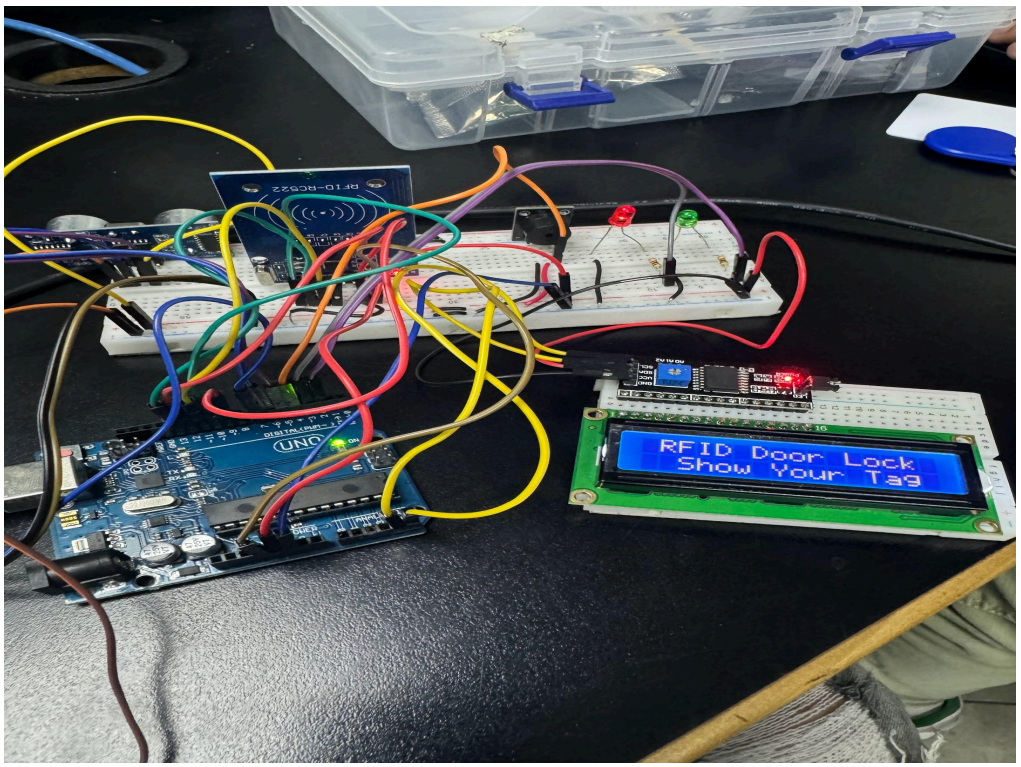


Image 13. Parking entrance circuit first implementation (RFID activated)

Parking Slots monitored using Raspberry Pico W and Arduino

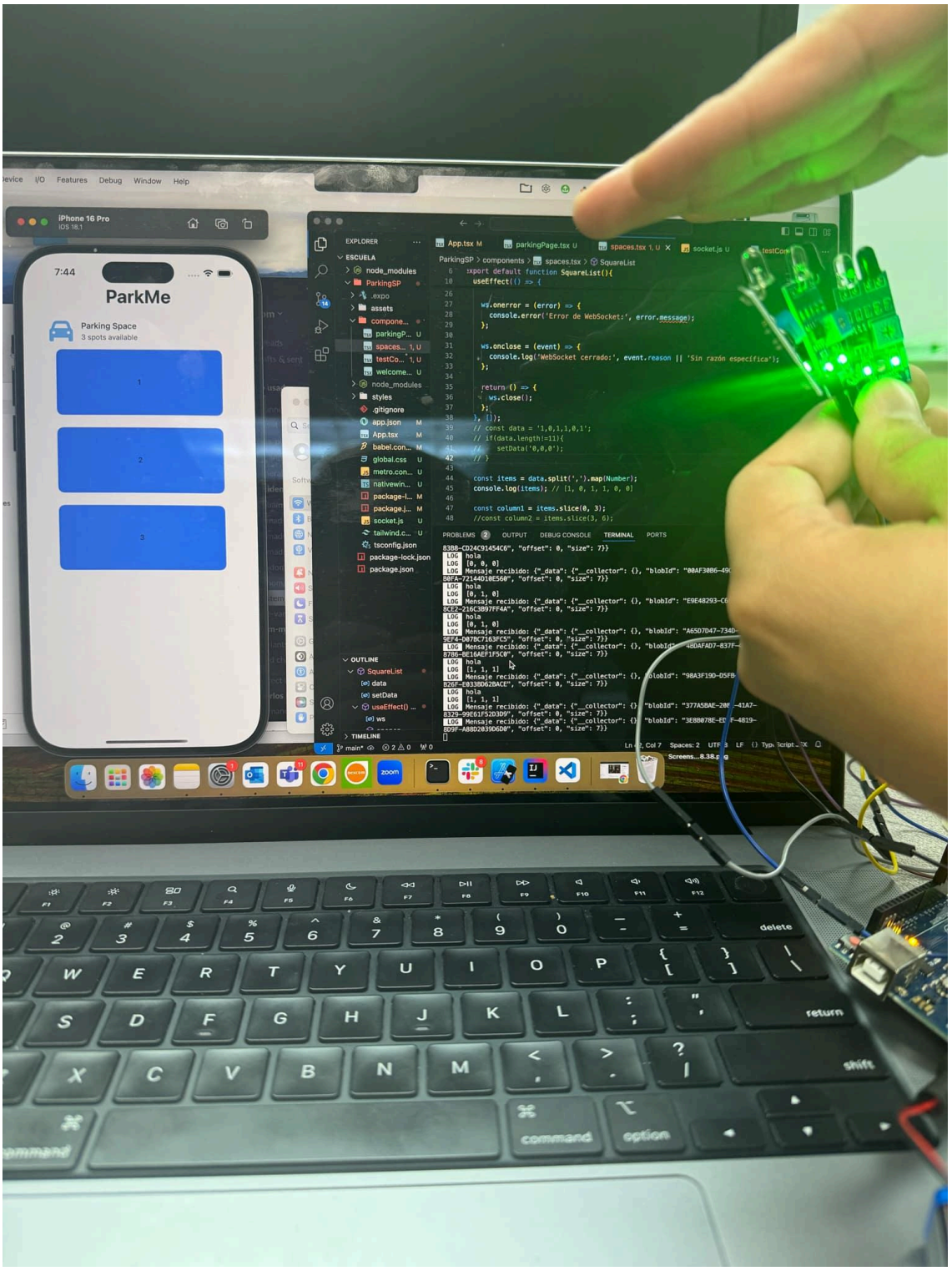


Image 14. Parking slots viewed in external application (Full)

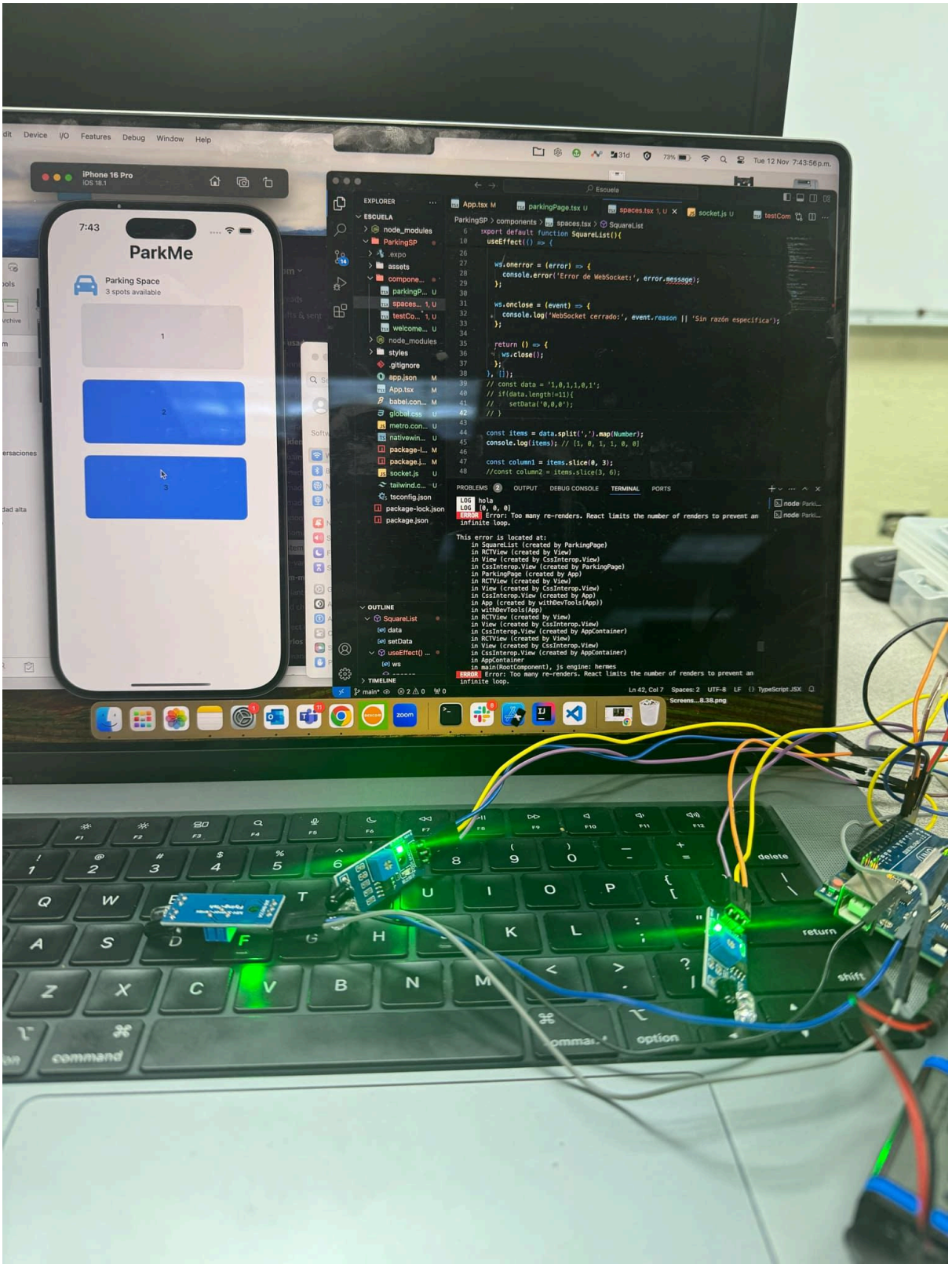
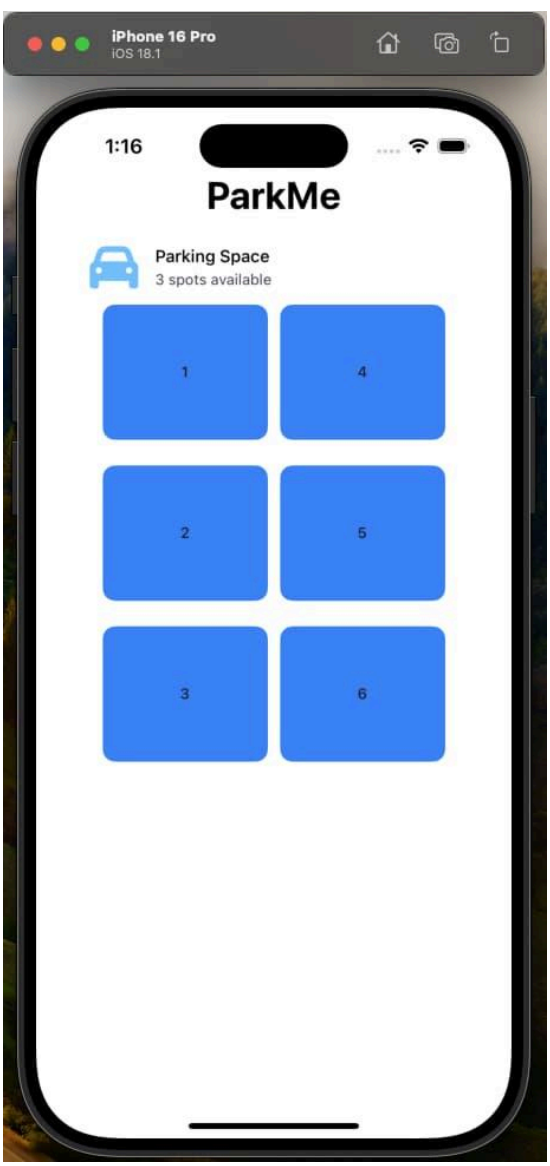


Image 15. Parking slots viewed in external application (1 space left)

Parking Slots monitoring application



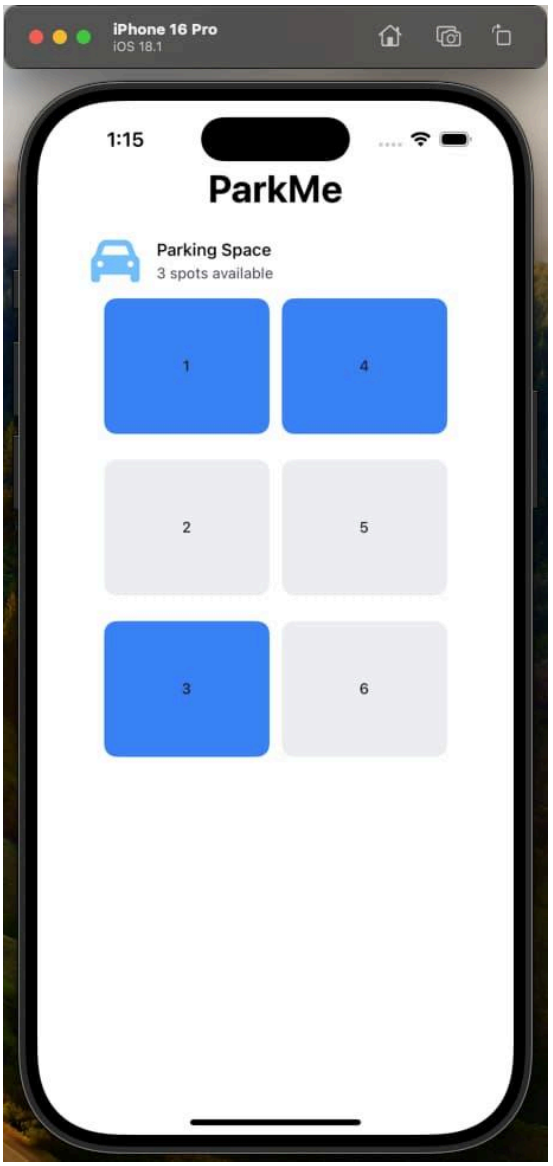


Image 16. Parking app (full occupied)

Image 17. Parking app (3 slots left)

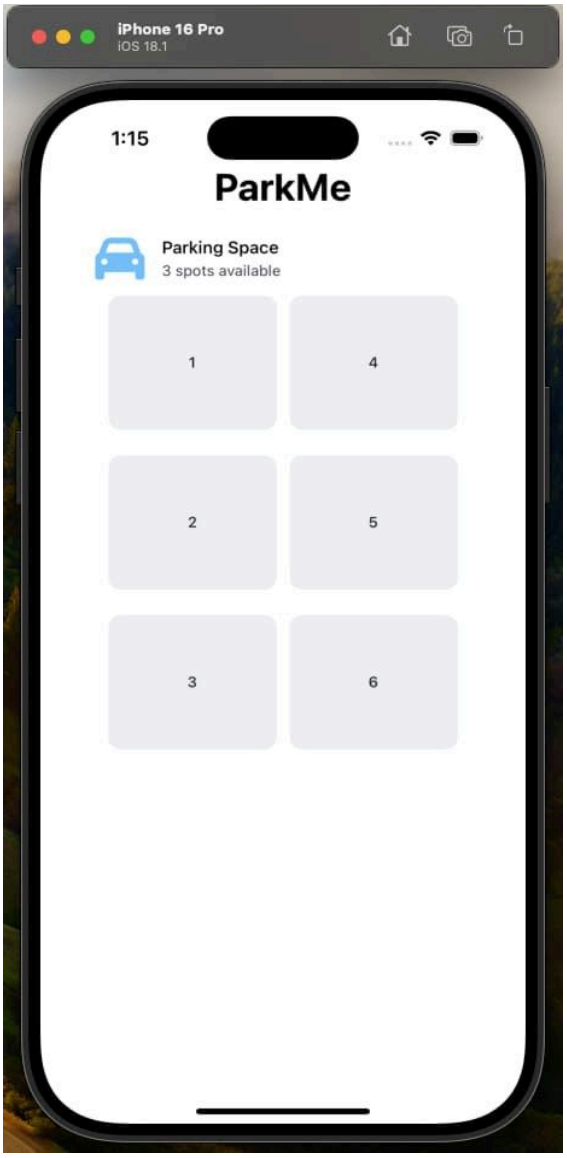


Image 18. Parking app (no spots occupied)

Parking physical design

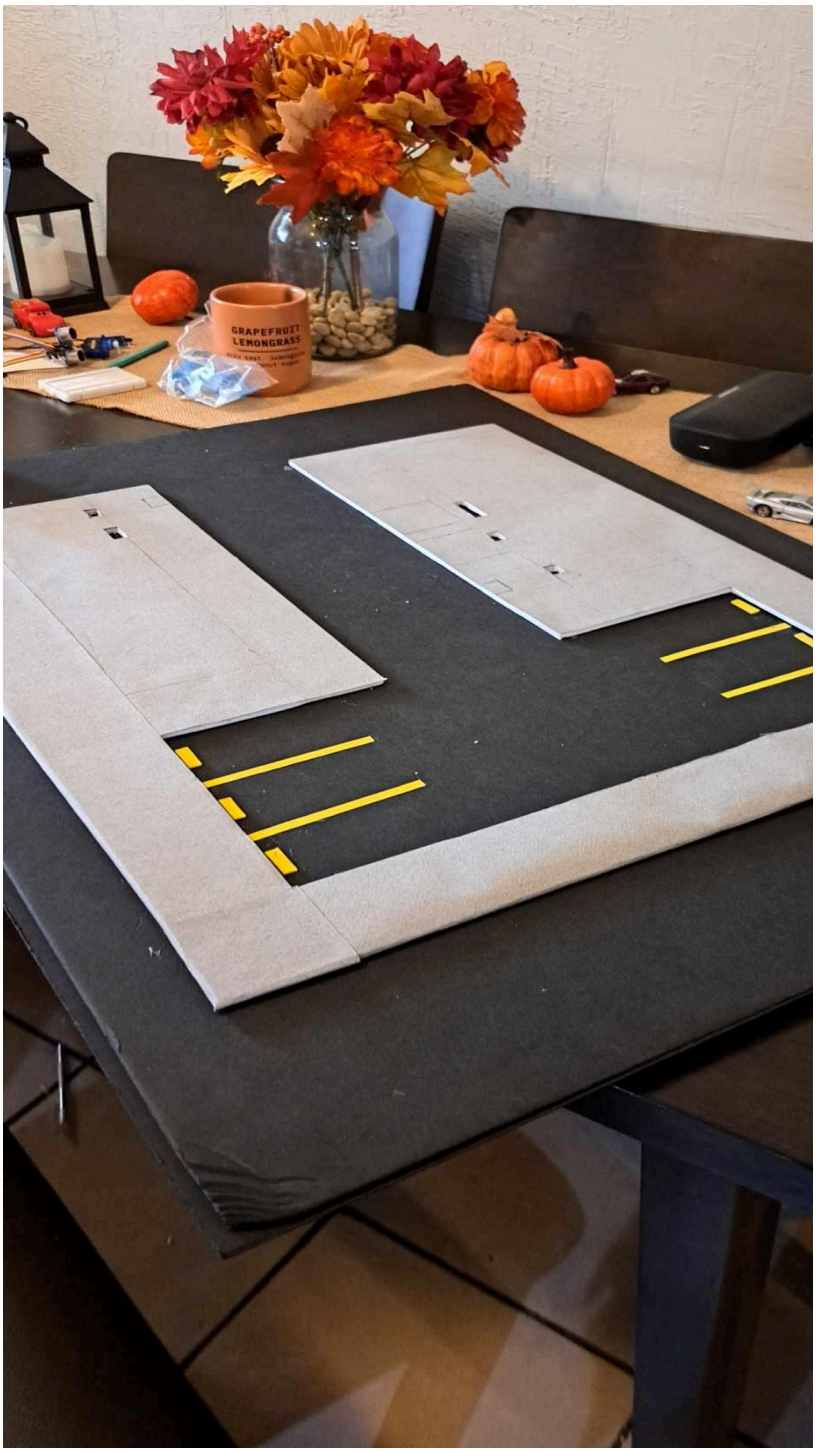


Image 19. Parking physical model (initial assembly)

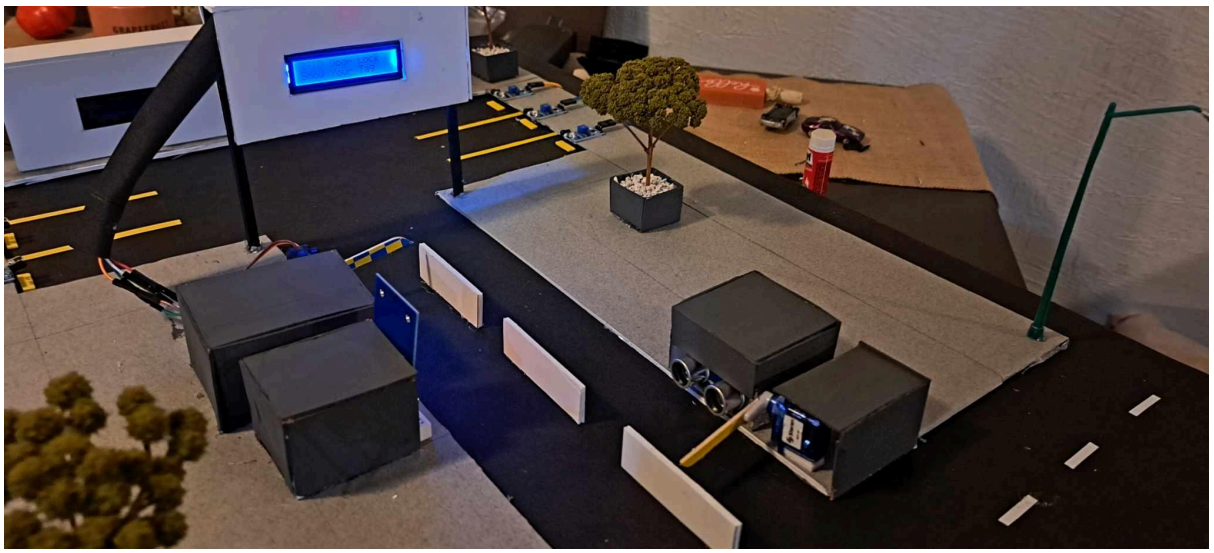


Image 20. Parking physical model (assembly in progress)

Image 21. Parking physical model (final design)

Conclusions

This project has basically addressed the problems of efficient streamlined management of parking spaces. This system displays a welcome message and provides information about the availability of parking space. By using this system, there would be a significant reduction of the cost incurred to hire personnel in order to control the traffic in the parking lot and traffic congestion problem will be solved by faster check in and check out.

A successful implementation of this project would result in less traffic and chaos in crowded parking spaces like in malls and business buildings where many people share a parking space. As the Smart Car Parking System requires minimal manpower, there are minimum chances for human error, increased security in addition to a swift and friendly car parking experience for drivers. Developing a smart parking solution in various buildings within a city would also solve the problem of air pollution by vehicles.

Recommendations

This project can be upgraded in several ways, for example, for a better infrastructure for sharing the IR sensors data that represents the availability of the parking slots, the data needs to be shared with a more potent hardware device, like a Raspberry Pi 4 or newer this due the need that this system of parking administration needs to be functional 24/7, and the web server needs to be in a server, that also operates 24/7 and handle all petitions from users that want to manage the parking status.

Bibliographic References

1. Feng, Y., & Liu, Y. (2017). Mechanical Parking System. Mechanical Engineering Blekinge Institute Of Technology.

2. Robin G., Danda B. & Rios F. (2017). Smart Parking: Parking Occupancy Monitoring and Visualization System for SmartCities. IEEE.

3. D. Kanteti, D. V. S. Srikar & T. K. Ramesh. (2017) Smart parking system for commercial stretches in cities,' 2017 International Conference on Communication and Signal Processing.