
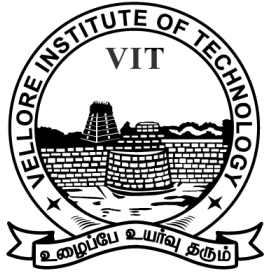


1. Drive Link for Video: [Video](#)
2. Website Link : [Smart Parking System Website](#)
3. TinkerCad Circuit Link: [Circuit Link](#)
4. ThingSpeak Channel Link: [Channel Stats](#)
5. Ppt Link:  IOT Presentation

Report Contd. → Next Page



# VIT<sup>®</sup>

## Vellore Institute of Technology

(Deemed to be University under section 3 of UGC Act, 1956)

School of Electronics Engineering VIT University, Vellore, Tamil Nadu

# Smart Parking System

## Project Report

### for

### ECE 3501

## IoT Fundamentals

Submitted by: -

**Sunanda Gupta-19BEC0593**

**Yash Agarwal-19BEC0599**

To

Prof. PRAKASH R

In G2 SLOT



**VIT**  
**UNIVERSITY**  
(Estd. u/s 3 of UGC Act 1956)

**SCHOOL OF ELECTRONICS ENGINEERING**

**Project IoT Fundamentals (ECE 3501)**

**It is certified that the project entitled “ Smart Parking System ” is the bonafide work for Project component of IoT Fundamentals by**

**YASH AGARWAL-19BEC0599**  
**SUNANDA GUPTA 19BEC0593**

**Of ECE branch under my supervision in G2 slot during the Fall Semester 2021 at V.I.T. University, Vellore- 632014**

Faculty

Signature:

Date:

## Introduction

In today's busy world, the concept of smart cities is much more widely accepted than it was just a decade ago. With the innovation in the Internet of Things and technology in general, people have a renewed vigour for a smarter future.

With this technological revolution also comes its challenges. The automobile industry is experiencing a boom and there are more vehicles on the road than ever before. This has led to traffic and congestion on the roads. Parking is also a problem that needs to be tackled.

In this project we aim to design and implement an IoT based parking system that would detect and alert the user when parking space is occupied and also indicate unoccupied spaces in a specified area.

This will help in alleviating the stress of looking for a parking spot when going out since it can be done in advance or even during transit.

## Objective

The problem of space and its availability is of much importance. The objective of the smart parking system is to streamline the painstaking and often time consuming process of finding and parking your vehicle. This not only saves us time, but also helps alleviate the congestion on the roads. This system cuts down on wait times and unnecessary loitering of vehicles that are in search of a parking spot.

The website enables us to access this information over the net and also register our vehicle to be parked at the location, which is then taken and used to update the number.

This system not only makes accessibility easy but also manages the congestion of vehicles avoiding long search and wait times. All bookings can be managed using the website.

## **Motivation**

With urban communities and metropolitan regions becoming busy constantly, discovering a parking spot is completely a test. It isn't just tedious yet in addition very disappointing. On account of IoT, there's an answer for taking care of the stopping issue emergency. This IoT based savvy stopping framework is intended to keep away from pointless voyaging and provocation in the quest for a suitable stopping region

## **Methodology**

### **Components and Software Used**

- Arduino Uno
- ESP8266
- Ultrasonic Sensors
- Resistors
- LEDs
- Breadboard
- 16\*2 LCD
- Tinkercad

- Thingspeak
- Hosting Server(Github Pages)

A smart parking system consists of infrared sensors that enable us to detect the presence of a vehicle in a particular parking spot. This information is processed by our microcontroller and sent to the cloud. The number of free spots is also reflected on the LCD screen, along with the dashboard of the website.

The website enables us to access this information over the net and also register our vehicle to be parked at the location, which is then taken and used to update the number.

This system not only makes accessibility easy but also manages the congestion of vehicles avoiding long search and wait times.

## **Website Working:**

When a user visits Smart Parking System's Website, on the website's dashboard all available parking slots are visible.

To successfully book/reserve a slot virtually, the end user has to enter their vehicle information after which a slot is booked for the same and available parking spots counter reduces by 1.

## **Arduino**



The Arduino UNO is an open source microcontroller based on the microchip ATMEGA328P microcontroller developed by Arduino.cc. It is a board that is equipped with sets of digital and analogue input/output (I/O) pins that may be interfaced to various expansion boards (shields) and other circuits.

The board has 14 Digital pins, 6 Analog pins, and programmable with the Arduino IDE (Integrated

Development Environment) via a type B USB cable. It can be powered by a USB cable or by an external 9- volt battery, though it accepts voltages between 7 and 20 volts. It is also similar to the Arduino Nano and Leonardo.

## ESP8266



The ESP8266 is a system on a chip Wi-Fi microchip for Internet of Things applications produced by Espressif Systems. The ESP8266 module enables microcontrollers to connect to 2.4 GHz Wi-Fi, using IEEE 802.11 bgn.

It allows microcontrollers to connect to a Wi-Fi network and make simple TCP/IP connections using Hayes-style commands.

## Ultrasonic Sensors

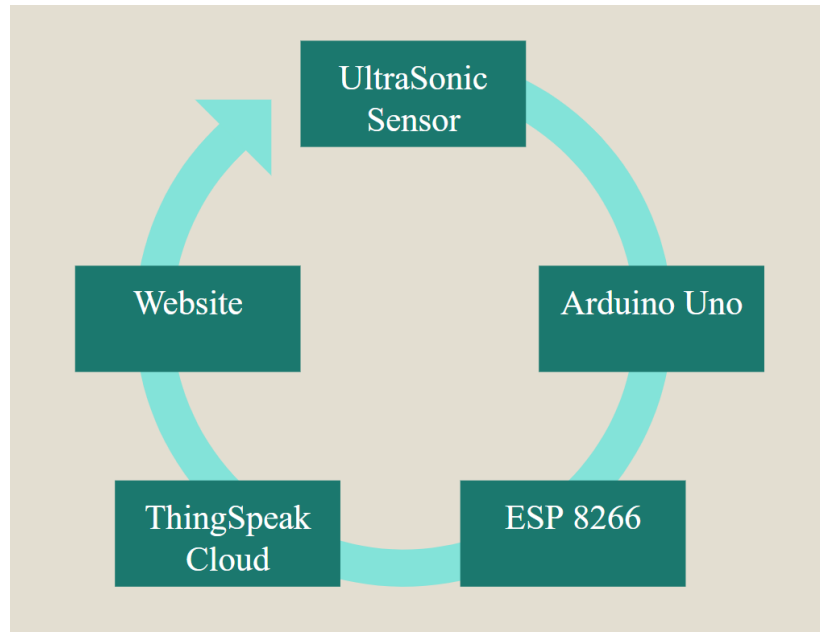


An ultrasonic sensor is a device that measures the distance between an object and itself using ultrasonic sound waves. It utilises a transducer in order to send and receive ultrasonic pulses that relay back information about an object's proximity to the sensor.

High frequency sound waves are reflected from boundaries and surfaces to produce distinct echo patterns that are used to determine distance.

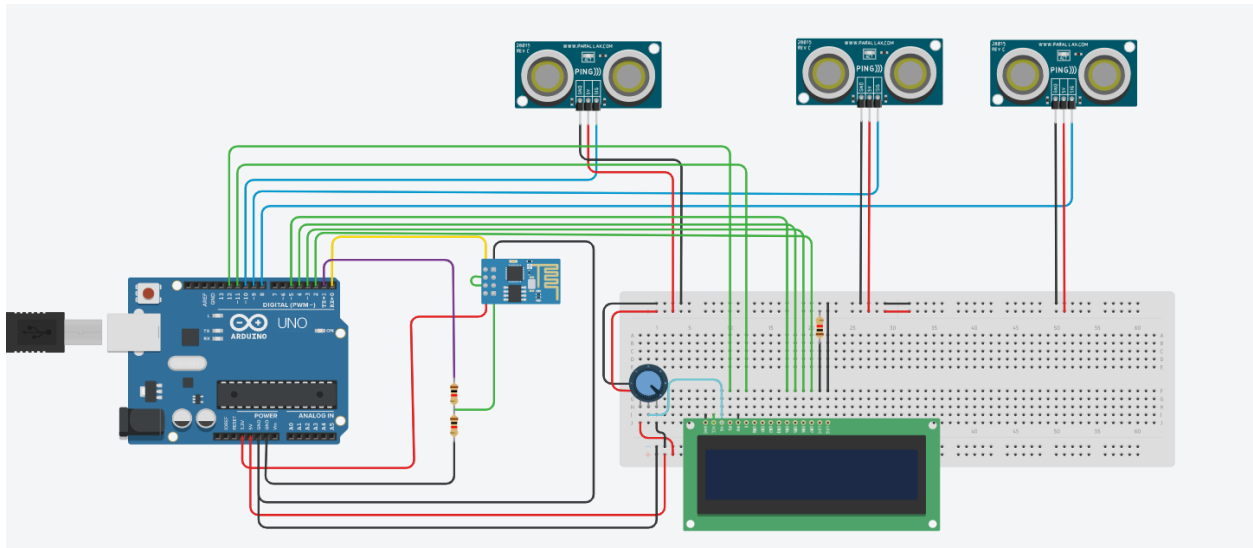


## Block Diagram



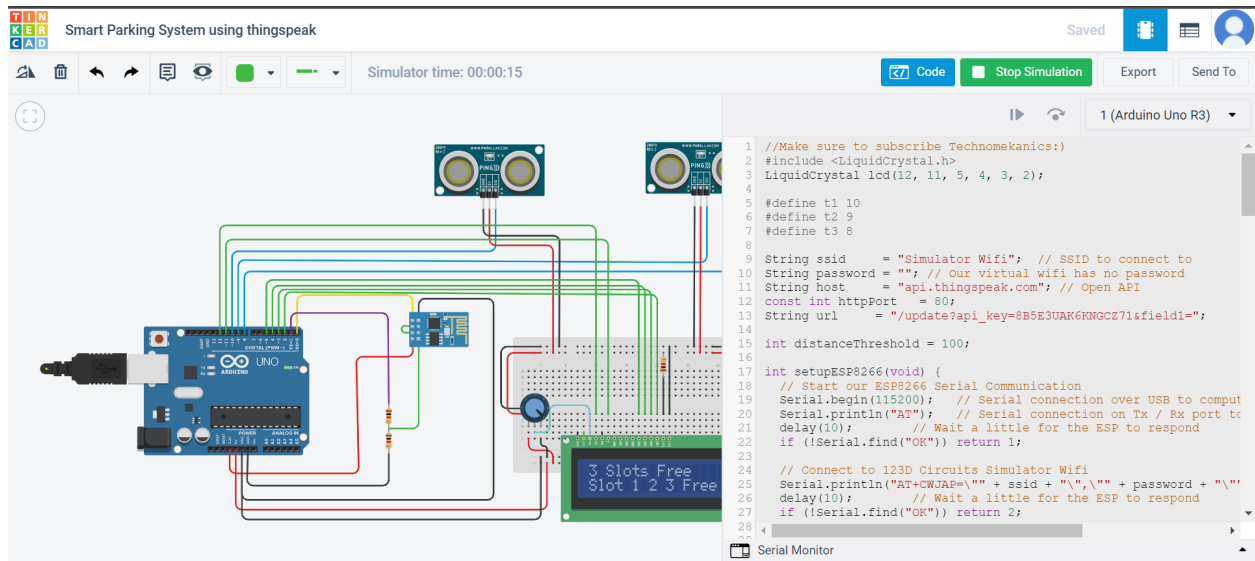
As indicated by the block diagram, analog information is taken from the ultrasonic sensor at the site and then sent to the Arduino Uno Microcontroller to be processed. This is also sent to the cloud using our ESP8266. The information is updated on the website.

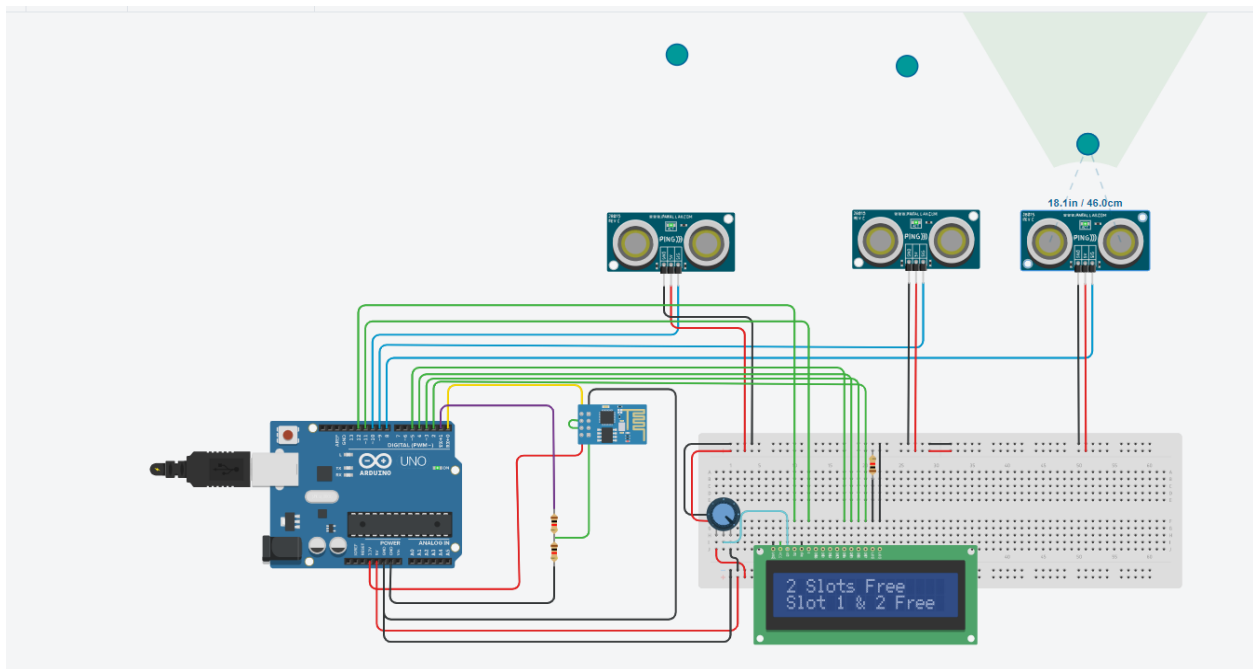
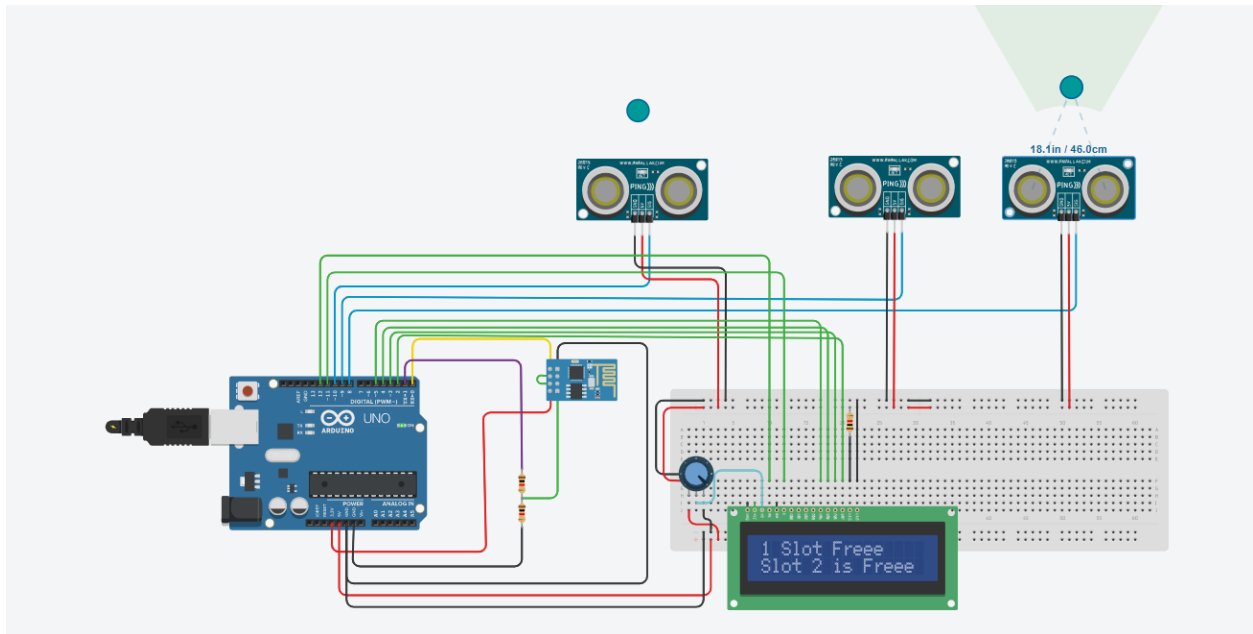
## Design

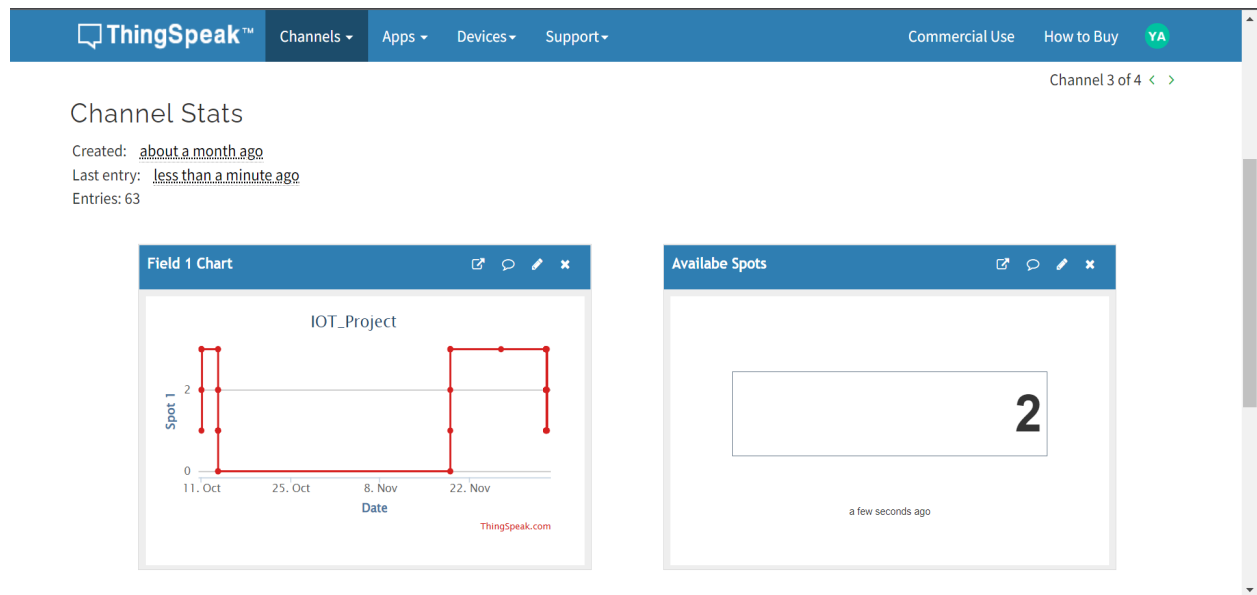


## Simulations and Results

Objective of the experiment and involved methodology has been portrayed through simulation







Vehicle Parking System

https://eradicate-2000.github.io/Parking-System.github.io/

Apps VIT Login to VTOP Microsoft Teams Mail Coding Ninjas Online Learning Carrer options WIFI DLD ALA Streaming Platforms Reading list

## Vehicle Parking System

DL 03 BN 0777

Car ▾

Add Vehicle

2

a few seconds ago

Vehicle No.	Vehicle Type	Vehicle Entry Time	Vehicle Exit Time	Departed	Amount
DL 03 BN 0777	Car	04 : 04	--	<div>Depart</div>	--

Vehicle Parking System

DL 04 Bb 1764

Truck

Add Vehicle

0

A few seconds ago

Vehicle No.	Vehicle Type	Vehicle Entry Time	Vehicle Exit Time	Departed	Amount
TN 04 Bb 1234	Car	23 : 26	--	Depart	--
TN 04 Bb 6764	Auto	23 : 26	--	Depart	--
DL 04 Bb 1764	Truck	23 : 27	--	Depart	--

## Conclusion

Hence we developed a smart parking system that enables us to detect and inform the user of the number of available parking slots.

This helps reduce the congestion on the already busy streets by giving people prior knowledge of space availability.

It also enables private and public parking spaces to monitor the vehicles parked and avoid overcrowding.

The website helps complete the user experience by providing a friendly and informative interface, helping to alleviate the day to day hustles.

# Code

## TinkerCad

```
#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

#define t1 10
#define t2 9
#define t3 8

String ssid  = "Simulator Wifi"; // SSID to connect to
String password = ""; // Our virtual wifi has no password
String host   = "api.thingspeak.com"; // Open API
const int httpPort = 80;
String url    = "/update?api_key=8B5E3UAK6KNGCZ71&field1=";

int distanceThreshold = 100;

int setupESP8266(void) {
    // Start our ESP8266 Serial Communication
    Serial.begin(115200); // Serial connection over USB to computer
    Serial.println("AT"); // Serial connection on Tx / Rx port to ESP8266
    delay(10);           // Wait a little for the ESP to respond
    if (!Serial.find("OK")) return 1;

    // Connect to 123D Circuits Simulator Wifi
    Serial.println("AT+CWLAP=\"" + ssid + "\",\"" + password + "\"");
    delay(10);           // Wait a little for the ESP to respond
    if (!Serial.find("OK")) return 2;

    // Open TCP connection to the host:
    Serial.println("AT+CIPSTART=\""TCP\"","" + host + "\", " + httpPort);
    delay(50);           // Wait a little for the ESP to respond
    if (!Serial.find("OK")) return 3;

    return 0;
}

long readDistance(int triggerPin, int echoPin)
{
    pinMode(triggerPin, OUTPUT);
    digitalWrite(triggerPin, LOW);
    delayMicroseconds(2);
    digitalWrite(triggerPin, HIGH);
```

```
delayMicroseconds(10);  
digitalWrite(triggerPin, LOW);  
pinMode(echoPin, INPUT);  
return pulseIn(echoPin, HIGH);  
}
```

```
void anydata(int temp) {
```

```
    // Construct our HTTP call  
    String httpPacket = "GET " + url + String(temp) + " HTTP/1.1\r\nHost: " + host + "\r\n\r\n";  
    int length = httpPacket.length();
```

```
    // Send our message length  
    Serial.print("AT+CIPSEND=");  
    Serial.println(length);  
    delay(10); // Wait a little for the ESP to respond if (!Serial.find(">")) return -1;
```

```
    // Send our http request  
    Serial.print(httpPacket);  
    delay(10); // Wait a little for the ESP to respond  
    if (!Serial.find("SEND OK\r\n")) return;
```

```
}
```

```
void setup() {
```

```
    lcd.begin(16,2);  
    lcd.setCursor(0,0);  
    Serial.begin (9600);  
    setupESP8266();
```

```
}
```

```
void loop() {
```

```
    float d1 = 0.01723 * readDistance(t1, t1);  
    float d2 = 0.01723 * readDistance(t2, t2);  
    float d3 = 0.01723 * readDistance(t3, t3);
```

```

Serial.println("d1 = " + String(d1) + "cm");
Serial.println("d2 = " + String(d2) + "cm");
Serial.println("d3 = " + String(d3) + "cm");

if (d1>100 & d2>100 & d3>100)
{
    lcd.setCursor(0,0);
    lcd.print("3 Slots Free");
    lcd.setCursor(0,1);
    lcd.print("Slot 1 2 3 Free");
    anydata(3);
    delay(500);
}
else if((d1>100 & d2>100)|(d2>100 & d3>100)|(d3>100 & d1>100))
{
    lcd.setCursor(0,0);
    lcd.print("2 Slots Free");
    lcd.setCursor(0,1);
    if(d1>100 & d2>100)
    {
        lcd.print("Slot 1 & 2 Free");
        anydata(2);
    }
    else if(d1>100 & d3>100)
    {
        lcd.print("Slot 1 & 3 Free");
        anydata(2);
    }
    else
    {
        lcd.print("Slot 2 & 3 Free");
        anydata(2);
    }
    delay(500);
}
else if(d1<100 & d2<100 & d3<100)
{
    lcd.setCursor(0,0);
    lcd.print("No Slot Free");
    lcd.setCursor(0,1);
    lcd.print("Parking Full");
    anydata(0);
    delay(500);
}

```



```

    }
else if((d1<100 & d2<100)|(d2<100 & d3<100)|(d3<100 & d1<100))
{
    lcd.setCursor(0,0);
    lcd.print("1 Slot Free");
    lcd.setCursor(0,1);
    if(d1>100)
    {
        lcd.print("Slot 1 is Free");
        anydata(1);
    }
    else if (d2>100)
    {
        lcd.print("Slot 2 is Free");
        anydata(1);
    }
    else
    {
        lcd.print("Slot 3 is Free");
        anydata(1);
    }
    delay(500);
}

delay(100);
}

```

### HTML Code

```

<!DOCTYPE html>
<html lang="en">

<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <link rel="stylesheet" href="css/style.css">
    <title>Vehicle Parking System</title>
</head>

<body>
    <div class="wrapper">
        <h1 class="heading">Vehicle Parking System</h1>

```

```

<div class="formInput">
  <input type="text" id="vehicleNo" placeholder="Enter The Vehicle Number">
  <select id="vehicleType">
    <option value="Select" selected>Select</option>
    <option value="Car">Car</option>
    <option value="Scooty">Scooty</option>
    <option value="Bike">Bike</option>
    <option value="Auto">Auto</option>
    <option value="Truck">Truck</option>
  </select>
  <button type="button" onclick="addVehicle()">Add Vehicle</button>
</div>
<div margin='5px' align='center'>
  <iframe width="450" height="260" style="border: 1px solid #cccccc;"
src="https://thingspeak.com/channels/1531868/widgets/365804"></iframe> </div>
<table>
  <thead>
    <tr>
      <th>Vehicle No.</th>
      <th>Vehicle Type</th>
      <th>Vehicle Entry Time</th>
      <th>Vehicle Exit Time</th>
      <th>Departed</th>
      <th>Amount</th>
    </tr>
  </thead>
  <tbody id="tbody">

  </tbody>
</table>
</div>
<script src="js/script.js"></script>
</body>

</html>

```

### Css File

```

@import
url('https://fonts.googleapis.com/css2?family=Poppins:wght@100;200;300;400;500;600;700;800;900&di
splay=swap');
*{
  margin: 0;

```

```

padding: 0;
}
body{
height: 100%;
width: 100%;
display: flex;
justify-content: center;
align-items: center;
background-color: rgb(97, 97, 97);
/* background: linear-gradient(40deg, #804bbd, #2b1645); */
flex-direction: column;
font-family: 'Poppins', sans-serif;
overflow-y: auto;
}
body::-webkit-scrollbar{
width: 0;
}
.heading{
width: 100%;
text-align: center;
font-size: 40px;
margin: 20px 0;
color: white;
}
.formInput{
margin-top: 50px;
width: 100%;
display: flex;
justify-content: center;
align-items: center;
flex-direction: column;
}
.tableParent{
overflow-y: auto;
}
.formInput input,
.formInput select{
height: 40px;
border: none;
outline: none;
width: 250px;
border-radius: 7px;
font-size: 17px;

```

```
    text-align: center;
}
.formInput button{
    padding: 0 10px;
    height: 40px;
    border-radius: 7px;
    margin-top: 30px;
    border: none;
    font-size: 17px;
    transition: 0.3s;
    outline: none;
}
.formInput button:hover{
    background-color: rgb(172, 172, 172);
}
.formInput select{
    margin-top: 10px;
}
table{
    margin: 40px 0;
}
th{
    border-top-left-radius: 5px;
    border-top-right-radius: 5px;
    font-size: 18px;
}
table th,
table td{
    width: 150px;
    text-align: center;
    height: 30px;
    background-color: white;
    padding: 5px 10px;
}
table button{
    border: none;
    outline: none;
    padding: 5px 10px;
    border-radius: 5px;
    transition: 0.3s;
}
table button:hover{
    background-color: rgb(172, 172, 172);
```

```
}
```

### JavaScript Code

```
let tbody = document.getElementById('tbody');
function addVehicle() {
    let vehicleType = document.getElementById('vehicleType').value;
    let vehicleNo = document.getElementById('vehicleNo').value;
    if (vehicleNo.length === 0 || vehicleType === "Select") {
        alert("Cannot Add Vehicle!");
    } else {
        let newTime = new Date();
        let hours = newTime.getHours();
        let minutes = newTime.getMinutes();
        if (hours < 10) {
            hours = `0${hours}`
        }
        if (minutes < 10) {
            minutes = `0${minutes}`
        }
        tbody.innerHTML += `
        <tr>
            <td>${vehicleNo}</td>
            <td>${vehicleType}</td>
            <td>${hours} : ${minutes}</td>
            <td>--</td>
            <td>
                <button type="button" onclick="departVehicle(this)">Depart</button>
            </td>
            <td>--</td>
        </tr>
        `;
    }
}

function departVehicle(element) {
    let newTime = new Date();
    let hours = newTime.getHours();
    let minutes = newTime.getMinutes();
    if (hours < 10) {
        hours = `0${hours}`
    }
    if (minutes < 10) {
        minutes = `0${minutes}`
    }
}
```

```
}
element.parentNode.parentNode.children[3].innerText = `${hours} : ${minutes}`;

let type = element.parentNode.parentNode.children[1].innerText;
let price;
switch (type) {
  case "Car":
    price = 100;
    break;
  case "Scooty":
    price = 10;
    break;
  case "Bike":
    price = 20;
    break;
  case "Auto":
    price = 50;
    break;
  case "Truck":
    price = 150;
    break;
}
element.parentNode.parentNode.children[5].innerText = `${price}Rs`;
element.disabled = true;
}
```

## References

- “The Internet of Things” by Samuel Greengard
- “Getting started with Internet of Things” by Cuno Pfister
- “Learning Internet of Things” by Peter Waher, Packt Publishing, 2015