WePARK

IOT BASED SMART PARKING SYSTEM

Project Report

SmartBridge - Internet of Things Externship

Group Members

Vishal Vythianathan K	20BEC1006
Shyam H	20BEC1055
Anirudh R	20BEC1276
Yukteshwar R	20BLC1029

TABLE OF CONTENTS

CHAPTER 1
INTRODUCTION

1.1 Overview

“WePark” is a Smart Parking System aimed at utilizing technology to optimize and streamline parking processes in urban areas. The system aims to improve parking management and enhance user experience by providing real-time information about available parking spaces and allowing users to reserve and navigate to them efficiently.

The system utilizes Wokwi, a virtual hardware simulation platform, to simulate and test the hardware components of the smart parking system. It allows developers to design and verify the functionality of sensors, actuators, and other electronic devices before physical implementation.

Node-RED, a visual programming tool, is employed to create the logic and workflows of the parking system. It enables easy integration and communication between different devices and services, ensuring seamless coordination and control.

IBM Cloud offers a robust and scalable cloud infrastructure to host the backend services of the smart parking system. It provides secure data storage, processing capabilities, and real-time analytics, allowing for efficient management of parking data and resources.

MIT App Inventor facilitates the development of a user-friendly mobile application that enables drivers to locate available parking spaces, make reservations, and receive real-time updates on parking availability.

Firebase, a mobile and web application development platform, serves as the backend for the mobile app, enabling seamless data synchronization, user authentication, and real-time communication between the app and the cloud-based services.

By integrating these technologies, the Smart Parking System aims to optimize parking space utilization, enhance user convenience, and improve overall parking management efficiency. It provides a scalable and adaptable solution that can be implemented in various parking environments, promoting smarter and more sustainable urban mobility.

1.2 Purpose

The project aims to address the common challenges faced in urban areas, such as limited parking availability and time wasted searching for vacant spots. By implementing a smart parking system, drivers will have access to real-time information about parking space availability, reducing congestion and improving overall traffic flow.

The project's objectives include creating a user-friendly interface for drivers to locate and reserve parking spots, integrating sensor technologies to detect and relay parking space occupancy, and developing an efficient algorithm for managing reservations and providing real-time updates.

CHAPTER 2
LITERATURE SURVEY

2.1 Existing Problem

AUTHOR	**TITLE**	**YEAR OF PUBLICATION**	**WORK DONE**	**INFERENCE**
ElakyaR,Juhi Seth, Pola Ashritha, R Namith	Smart Parking System using IoT	2019	smart way to automate the management of the parking system that allocates an efficient parking space	IR sensors are used to detect the availability of vehicles and send the data back to the cloud which further sends a message to the RFID tag owner about the availability of free/occupied spaces.
Ali, G., Ali, T., Irfan, M., Draz, U., Sohail, M., Glowacz, A., ... & Martis, C	IoT Based Smart Parking System Using Deep Long Short Memory Network	2020	A framework based on a deep long short term memory network to predict the availability of parking space with the integration of Internet of Things (IoT), cloud technology, and sensor networks.	A framework is created based on location, days of a week, and working hours and depending on this data the availability of parking spaces are determined.

Sant, A., Garg, L., Xuereb, P., & Chakraborty, C.	A Novel Green IoT-Based Pay-As-You-Go Smart Parking System	2021	IoT-based Pay- As-You-Go (PAYG) smart parking system by utilizing unused garage parking space	In this system the unused garage parking spaces are used as paid parking spaces and a system is built to store the user information in the cloud and to pay the garage owner automatically using online transfer of money.
Luque-Vega, L. F., Michel-Torres, D. A., Lopez-Neri, E., Carlos-Mancilla, M. A., González-Jiménez, L. E.	IoT Smart Parking System Based on the Visual-Aided Smart Vehicle Presence Sensor	2020	To improve mobility in cities with a larger and larger vehicle fleet, a novel sensing solution that is the cornerstone of a smart parking system, the smart vehicular presence sensor based Parking system.	The smart vehicular sensor is used to sense the presence of any vehicle in a allotted parking lot and stores the data in the cloud.

Atiqur, R.	Radio frequency identification based smart parking system using Internet of Things	2021	RFID based smart parking system using IoT is actualized. The ultrasonic sensors are set before the parking spots and speaker is utilized for sign. On the off chance that the parking space is vacant, at that point LED will squint and the parking opening is full LED will be OFF. At the point when vehicle is left the RFID will peruse the data and cut the particular sum and by utilizing IoT location will be sent through SMS to proprietor using GSM and GPS.	The vehicle user is notified as to whether there is space left for parking. The ultrasonic sensor senses and sends the data to cloud which determines whether there is any vacant space left.

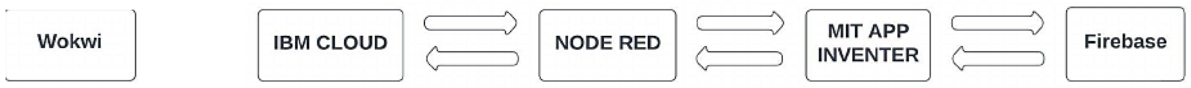
2.2 Proposed Solution

The proposed solution for the "Smart Parking System" project involves integrating Wokwi, Node-RED, IBM Cloud, MIT App Inventor, and Firebase to create a comprehensive system that enables real-time monitoring of parking spaces, provides user-friendly mobile applications, and utilizes cloud services for efficient data storage and management.

CHAPTER 3

THEORETICAL ANALYSIS

3.1 Block Diagram



3.2 Hardware and Software Requirements

1. Wokwi Simulator

   Components:

   1. ESP32
   2. Ultrasonic Sensors
   3. Leds
   4. PIR Sensors
   5. Servo Motors

2. IBM Cloud Platform
3. Node Red
4. MIT App Inventor
5. Firebase Cloud Platform
6. Android Mobile

CHAPTER 4
EXPERIMENTAL ANALYSIS

Market research: Examine the features, perks, characteristics of the market's current smart parking options. To differentiate your app, find any areas of weakness or improvement.

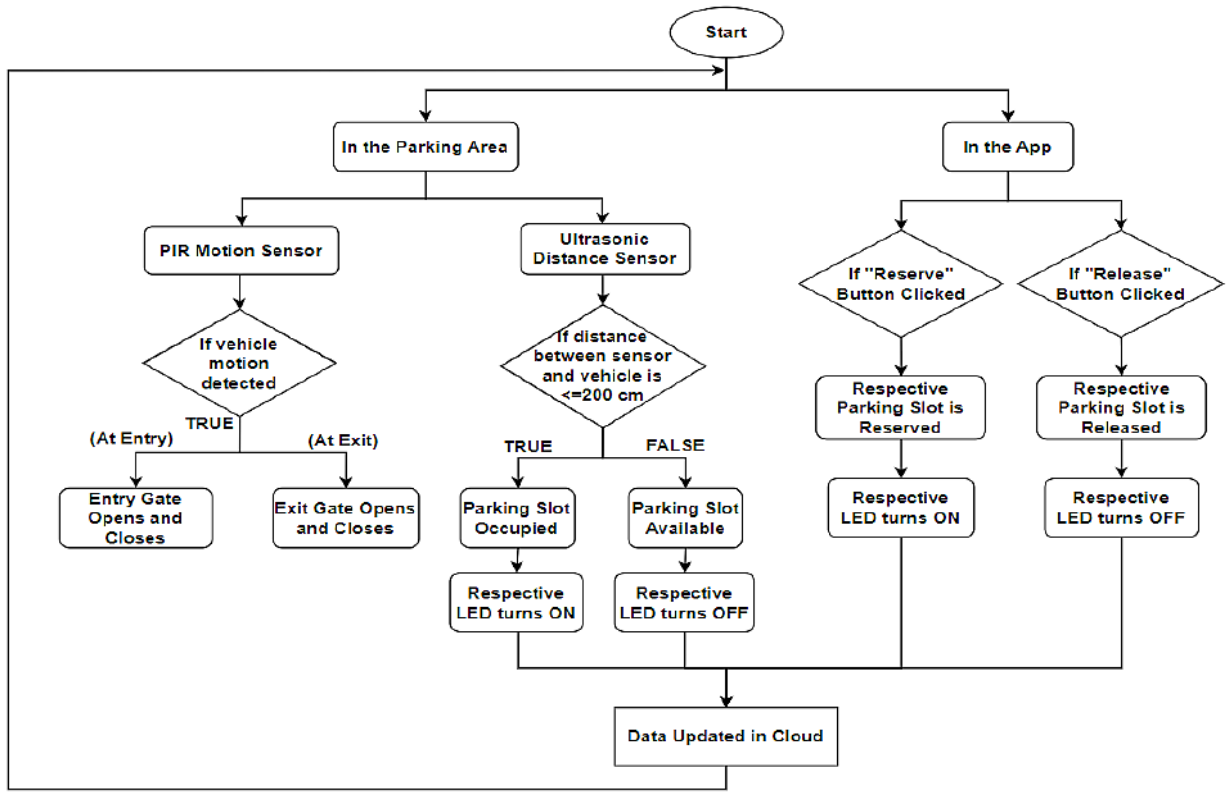
User Needs Analysis: It is essential to comprehend the requirements and expectations of the target audience. Gather information on the frequent parking-related pain points, preferences for app features, and desired user experience through surveys, interviews, or focus groups.

Technology Evaluation: Evaluate the different technologies and infrastructure needed to deploy the smart parking system. To find the best solutions for your app, weigh factors including sensors, cameras, mobile app platforms, cloud services.

Analyse historical parking data to identify patterns and trends, taking into account factors such as occupancy rates, peak times.

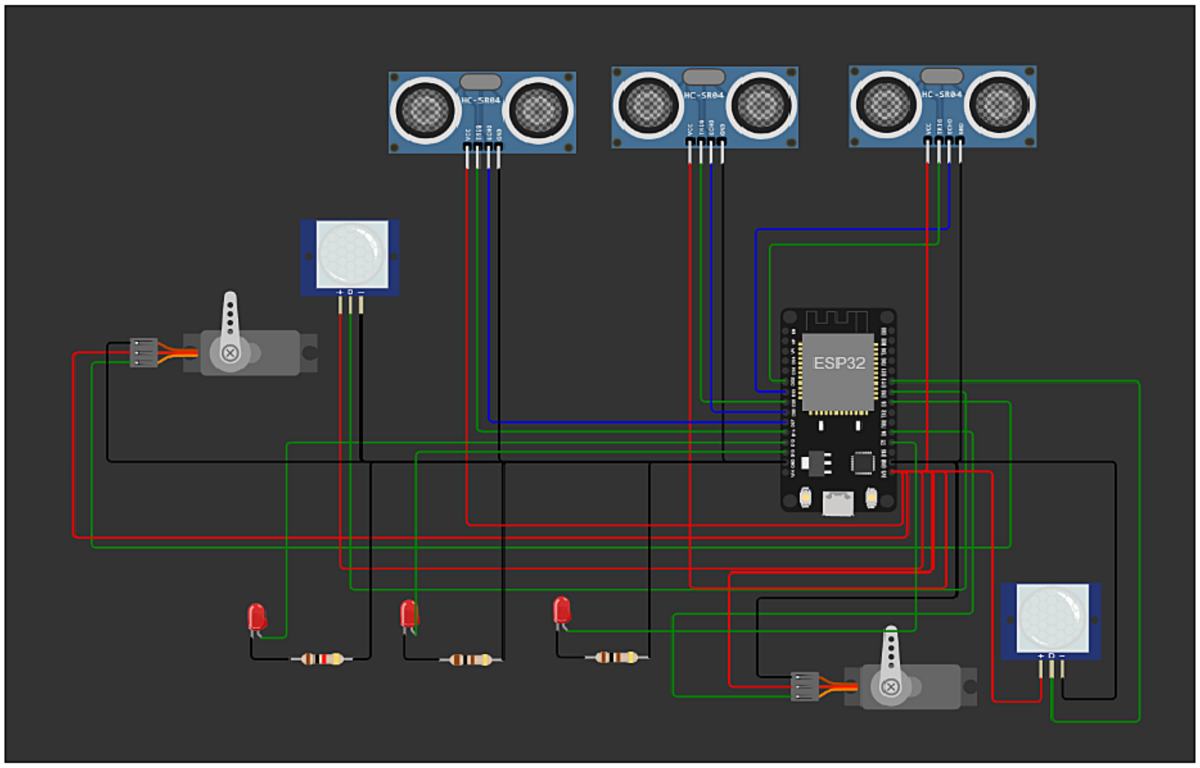
Performance testing: Create many virtual scenarios to evaluate how well the app performs under varied loads and busy parking times. To guarantee a smooth user experience, evaluate response times, scalability, and overall system stability.

CHAPTER 5
FLOWCHART

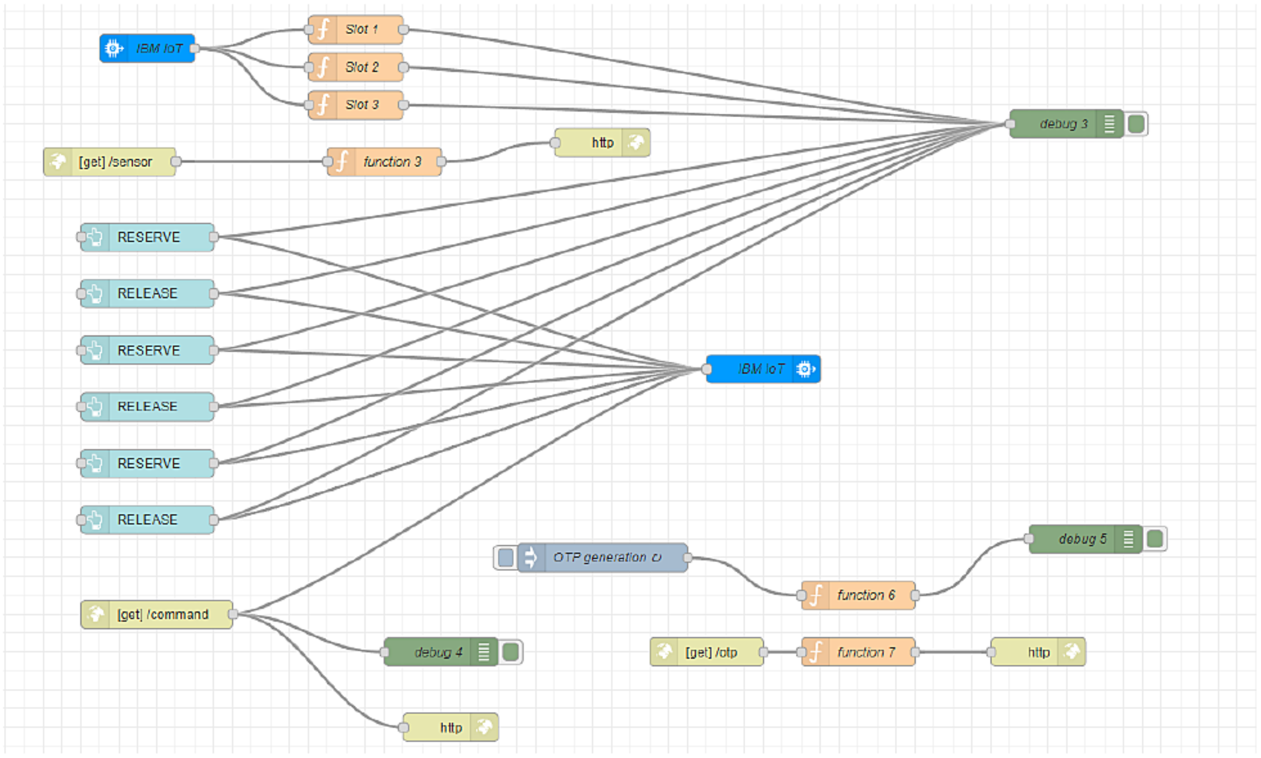

Start
In the Parking Area
In the App
PIR Motion Sensor
Ultrasonic Distance Sensor
If "Reserve" Button Clicked
If "Release" Button Clicked
If vehicle motion detected
TRUE
(At Entry)
(At Exit)
If distance between sensor and vehicle is <=200 cm
TRUE
FALSE
Respective Parking Slot is Reserved
Respective Parking Slot is Released
Entry Gate Opens and Closes
Exit Gate Opens and Closes
Parking Slot Occupied
Parking Slot Available
Respective LED turns ON
Respective LED turns OFF
Respective LED turns ON
Respective LED turns OFF
Data Updated in Cloud

CHAPTER 6
RESULT

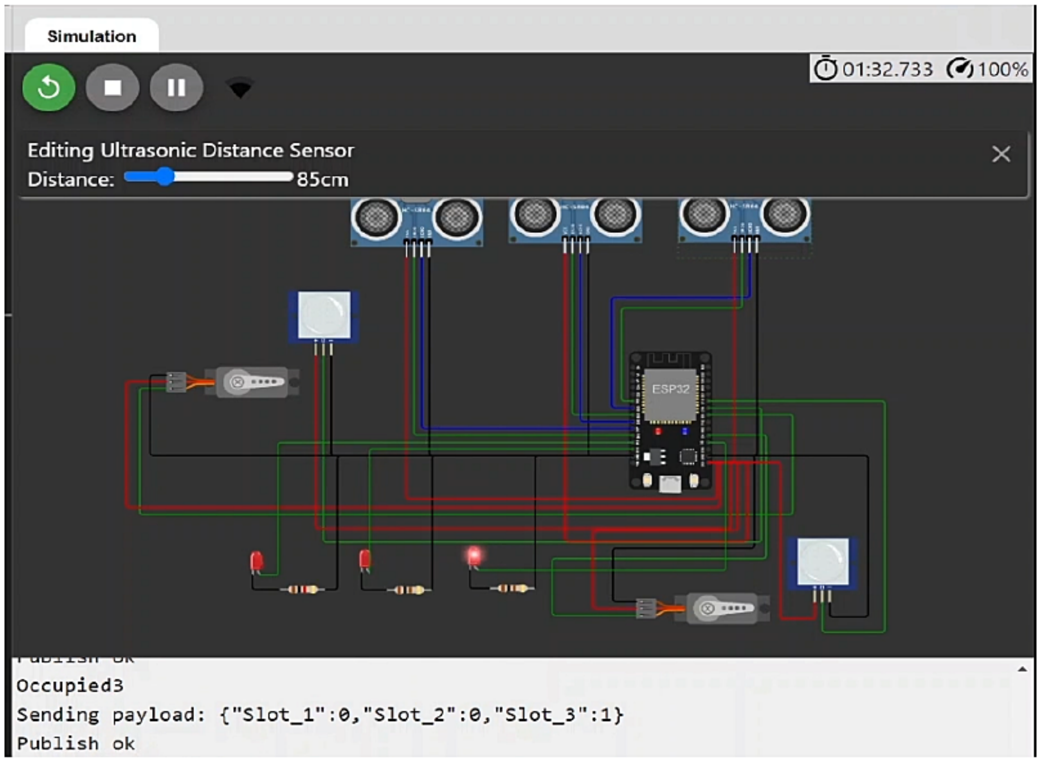
The smart parking system project successfully implemented advanced technologies to efficiently manage parking spaces. Utilizing sensors, real-time data, and a user-friendly mobile app, it streamlined parking operations, reduced congestion, and enhanced user experience. The project's outcomes demonstrated improved parking availability, reduced search time, and increased overall parking efficiency.



Wokwi Circuit

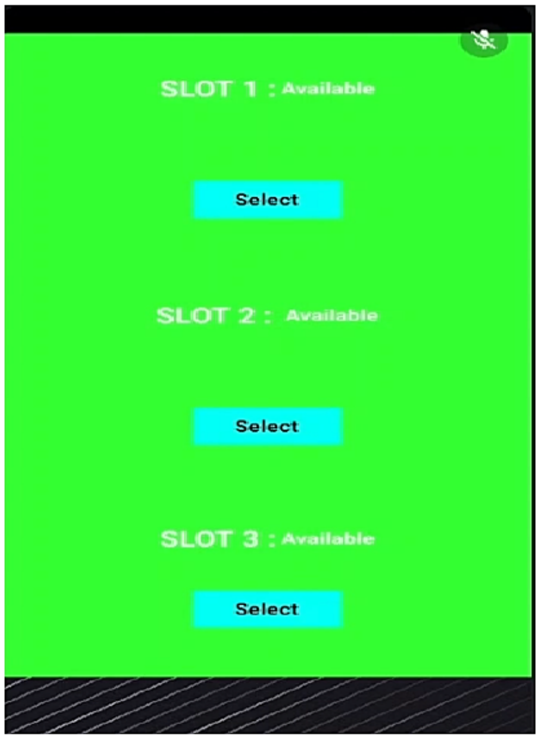


Node-red Flow

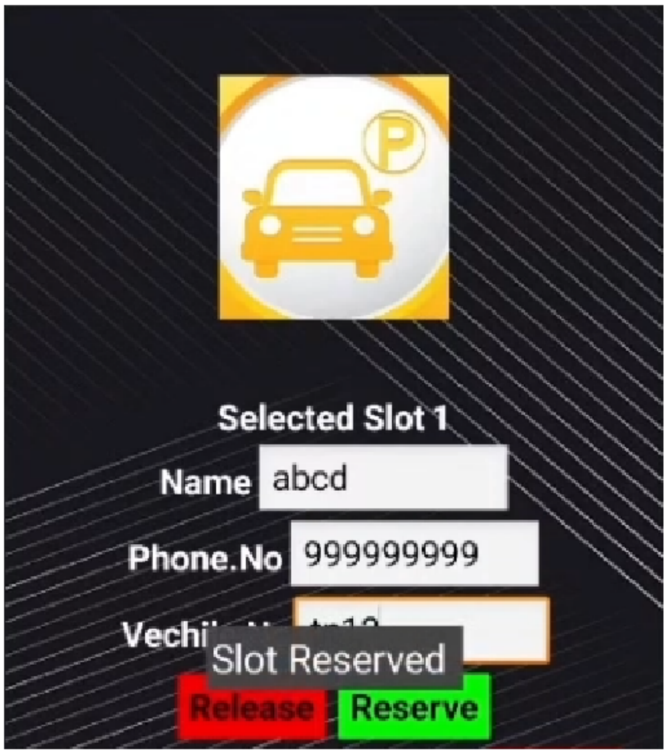


```
Publish ok
Occupied3
Sending payload: {"Slot_1":0,"Slot_2":0,"Slot_3":1}
Publish ok
```

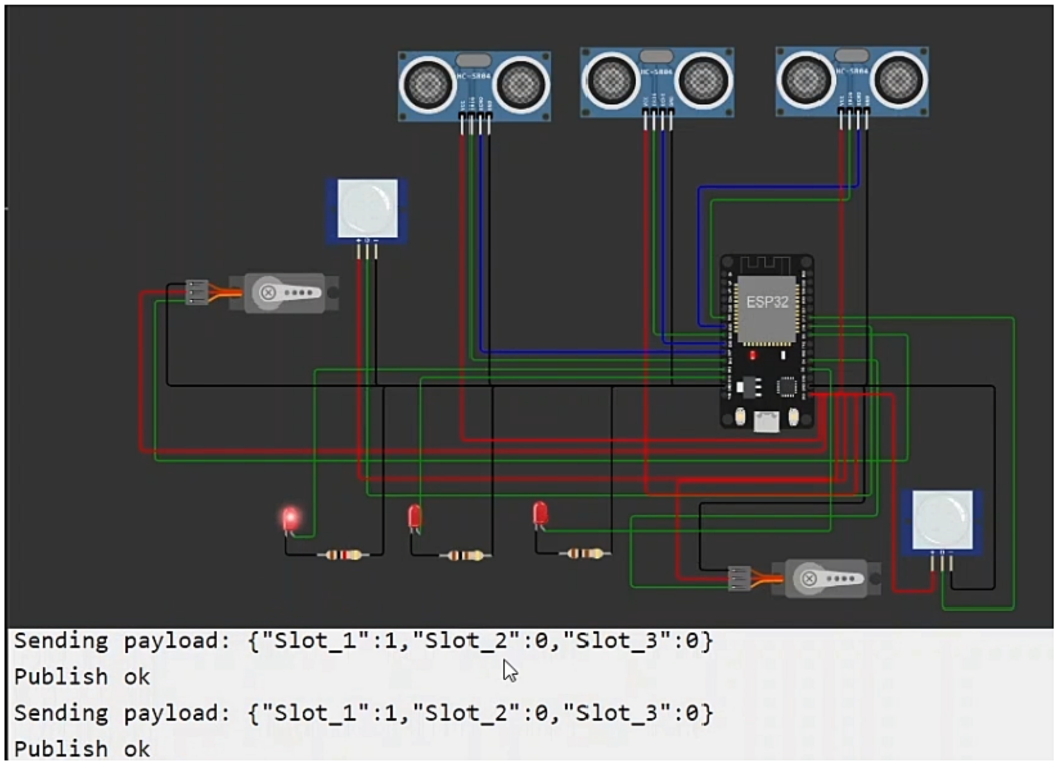
When Slot 3 is occupied



App display when all 3 slots are available



App display when slot is reserved



```
Sending payload: {"Slot_1":1,"Slot_2":0,"Slot_3":0}
Publish ok
Sending payload: {"Slot_1":1,"Slot_2":0,"Slot_3":0}
Publish ok
```

When Slot is reserved LED glows and status becomes occupied

CHAPTER 7
ADVANTAGES AND DISADVANTAGES

Advantages:

- Ease of Use
- Real Time monitoring
- Scalability
- Data Analytics
- Integration Capabilities

Disadvantages:

- Technical Complexity
- Cost
- Reliance on Internet Connectivity
- Privacy and Security concerns
- Limited Hardware support

CHAPTER 8
APPLICATIONS

Here are some few applications:

- Parking Guidance system
- Mobile Application for drivers
- Revenue Management
- Integration with smart city infrastructure
- Data Analytics

CHAPTER 9
CONCLUSION

The project successfully implemented a Smart Parking System by integrating Wokwi, NodeRed, IBM Cloud, MIT App Inventor, and Firebase. It demonstrated efficient parking management through real-time data monitoring, automated notifications, and user-friendly mobile application. The solution aimed to optimize parking space utilization and enhance user experience for a smarter and more convenient parking system.

CHAPTER 10
FUTURE SCOPE

The future scope of this project is promising. The integration of these advanced technologies allows for a comprehensive and efficient parking management solution. With the increasing challenges of urbanization and limited parking spaces, such a system has immense potential for implementation in smart cities. By leveraging Wokwi for simulation, NodeRed for data processing, IBM Cloud for storage and analysis, MIT App Inventor for user interface development, and Firebase for real-time updates, the project can provide seamless parking guidance, occupancy tracking, and payment integration. Additionally, the system's scalability enables future expansion and integration with emerging technologies, making it a viable solution for the evolving parking needs of urban areas.

CHAPTER 11

BIBLIOGRAPHY AND APPENDIX

REFERENCES:

1. https://www.ijeat.org/wp-content/uploads/papers/v9i1/A1963109119.pdf

2. Ali, G., Ali, T., Irfan, M., Draz, U., Sohail, M., Glowacz, A., ... & Martis, C. (2020). IoT based smart parking system using deep long short memory network. *Electronics*, *9*(10), 1696.

3. Sant, A., Garg, L., Xuereb, P., & Chakraborty, C. (2021). A novel green IoT-based pay-as-you-go smart parking system. *CMC Comput Mater Cont*, *67*(3), 3523-3544.

4. Luque-Vega, L. F., Michel-Torres, D. A., Lopez-Neri, E., Carlos-Mancilla, M. A., & González-Jiménez, L. E. (2020). Iot smart parking system based on the visual-aided smart vehicle presence sensor: SPIN-V. *Sensors*, *20*(5), 1476.

5. Atiqur, R. (2021). Radio frequency identification based smart parking system using internet of things. *IAES International Journal of Robotics and Automation*, *10*(1), 41

Wokwi Code:

```
#include <ESP32Servo.h>
#include <WiFi.h>//library for wifi
#include <PubSubClient.h>//library for MQtt

Servo myservo; //servo as gate
Servo myservos; //servo as gate

int carEnter = 19; // entry sensors
int carExited = 18;
int pir_stat_1 = 0;
int pir_stat_2 = 0;
int count = 0;
int pos = 0;
int status1, status2, status3;
int c=0;
int d=0;
```

```
boolean entrysensor, exitsensor,s1,s2,s3;

int e1=27;
int t1=14;
int e2=26;
int t2=25;
int e3=33;
int t3=32;
int led1=12;
int led2=13;
int led3=2;
boolean s1_occupied = false;
boolean s2_occupied = false;
boolean s3_occupied = false;


void callback(char* subscribetopic, byte* payload, unsigned int payloadLength);

//-------credentials of IBM Accounts------

#define ORG "axcdx6"//IBM ORGANITION ID
#define DEVICE_TYPE "abcd"//Device type mentioned in ibm watson IOT Platform
#define DEVICE_ID "1234"//Device ID mentioned in ibm watson IOT Platform
#define TOKEN "12345678" //Token
String data3;


//-------- Customise the above values --------
char server[] = ORG ".messaging.internetofthings.ibmcloud.com";// Server Name
char publishTopic[] = "iot-2/evt/Data/fmt/json";// topic name and type of event perform and format in which data to be send
char subscribetopic[] = "iot-2/cmd/command/fmt/String";// cmd REPRESENT command type AND COMMAND IS TEST OF
FORMAT STRING
char authMethod[] = "use-token-auth";// authentication method
char token[] = TOKEN;
char clientId[] = "d:" ORG ":" DEVICE_TYPE ":" DEVICE_ID;//client id



WiFiClient wifiClient; // creating the instance for wificlient
PubSubClient client(server, 1883, callback ,wifiClient); //calling the predefined client id by passing parameter like server
id,portand wificredential
```

```
void   setup()  {
delay(1000);
  Serial.begin (9600);
  wificonnect();
  mqttconnect();
  myservo.attach(4); // entry
  myservos.attach(5); // exit
  pinMode(carExited, INPUT); // ir as input
  pinMode(carEnter, INPUT); // ir as input

  pinMode(e1,INPUT);      //echo      pin
  pinMode(t1,OUTPUT); //trigger pin

  pinMode(e2,INPUT);      //echo      pin
  pinMode(t2,OUTPUT); //trigger pin

  pinMode(e3,INPUT);      //echo      pin
  pinMode(t3,OUTPUT); //trigger pin

  pinMode(led1,OUTPUT);    //LED
  pinMode(led2,OUTPUT);    //LED
  pinMode(led3,OUTPUT); //LED



void loop() {

digitalWrite(t1,LOW);
digitalWrite(t1,HIGH);
delayMicroseconds(10);
digitalWrite(t1,LOW);
float dur1 = pulseIn(e1,HIGH);
float dis1 = (dur1*0.0343)/2;

digitalWrite(t2,LOW);
digitalWrite(t2,HIGH);
delayMicroseconds(10);
digitalWrite(t2,LOW);
float dur2 = pulseIn(e2,HIGH);
float dis2 = (dur2*0.0343)/2;
```

```
digitalWrite(t3,LOW);
digitalWrite(t3,HIGH);
delayMicroseconds(10);
digitalWrite(t3,LOW);
float dur3 = pulseIn(e3,HIGH);
float dis3 = (dur3*0.0343)/2;

 pir_stat_1=      digitalRead(carEnter);
 pir_stat_2 = digitalRead(carExited);

 //s1 = digitalRead();
 //s2 = digitalRead();
 //s3 = digitalRead();

   if (pir_stat_1 == HIGH) { // if high then count and send data
   for(pos=90;pos<=180;pos++)

    myservo.write(pos);
    delay(15);

  for(pos=180;pos>=90;pos--)

    myservo.write(pos);
    delay(15);



  if   (pir_stat_2   ==   HIGH)   {   //if   high   then   count   and   send
  for(pos=0;pos<=180;pos++)

    myservos.write(pos);
    delay(15);

   for(pos=180;pos>=0;pos--)

    myservos.write(pos);
    delay(15);


 if(c==0)
 if (dis1<=200 && s1_occupied == false) {
 Serial.println("Occupied1 ");
 s1_occupied = true;
 digitalWrite(led1, HIGH);
 }
 if(dis1>200 && s1_occupied == true) {
 Serial.println("Available1 ");
 s1_occupied = false;
 digitalWrite(led1, LOW);

if(d==0)

 if (dis2<=200 && s2_occupied == false) {
 Serial.println("Occupied2 ");
 s2_occupied = true;
 digitalWrite(led2, HIGH);
 }
 if(dis2>200 && s2_occupied == true) {
 Serial.println("Available2 ");
 s2_occupied = false;
 digitalWrite(led2, LOW);

if(e==0)

 if (dis3<=200 && s3_occupied == false) {
 Serial.println("Occupied3 ");
 s3_occupied = true;
 digitalWrite(led3, HIGH);
 }
 if(dis3>200 && s3_occupied == true) {
 Serial.println("Available3 ");
 s3_occupied = false;
 digitalWrite(led3, LOW);

 if(s1_occupied==true)

   status1=1;

 else if(s1_occupied==false)
```

```
    status1=0;

  if(s2_occupied==true)

    status2=1;

  else if(s2_occupied==false)

    status2=0;

  if(s3_occupied==true)

    status3=1;

  else if(s3_occupied==false)

    status3=0;

  PublishData(status1,status2,status3);
  delay(5000);
  if (!client.loop()) {
  mqttconnect();



/*.....................................retrieving to Cloud...............................*/

void PublishData(int status1, int status2, int status3) {
mqttconnect();//function call for connecting to ibm


  creating the String in in form JSon to update the data to ibm cloud
  */
  String payload = "{\"Slot_1\":";
  payload += status1;
  payload += "," "\"Slot_2\":";
  payload += status2;
  payload += "," "\"Slot_3\":";
  payload += status3;
  payload += "}";
  Serial.print("Sending payload: ");
  Serial.println(payload);
```

```
if (client.publish(publishTopic, (char*) payload.c_str())) {
Serial.println("Publish ok");// if it sucessfully upload data on the cloud then it will print publish ok in Serial monitor or else it
will print publish failed
} else {
Serial.println("Publish failed");
}
}
void mqttconnect() {
if (!client.connected()) {
Serial.print("Reconnecting client to ");
Serial.println(server);
while (!!!client.connect(clientId, authMethod, token)) {
Serial.print(".");
delay(500);
}
initManagedDevice();

}
void wificonnect() //function defination for wificonnect

Serial.println();
Serial.print("Connecting to ");
WiFi.begin("Wokwi-GUEST", "", 6);//passing the wifi credentials to establish the connection
while (WiFi.status() != WL_CONNECTED) {
delay(500);
Serial.print(".");
}
Serial.println("");
Serial.println("WiFi connected");
Serial.println("IP address: ");
Serial.println(WiFi.localIP());
}
void initManagedDevice() {
if (client.subscribe(subscribetopic)) {
Serial.println((subscribetopic));
Serial.println("subscribe to cmd OK");
} else {
```

```
void callback(char* subscribetopic, byte* payload, unsigned int payloadLength)

 Serial.print("callback invoked for topic: ");
 Serial.println(subscribetopic);
 for (int i = 0; i < payloadLength; i++) {
 //Serial.print((char)payload[i]);
 data3 += (char)payload[i]; }
 Serial.println("data: "+ data3);
 if(data3=="{\"command\":\"Reserve slot-1\"}") {
 Serial.println(data3);
 digitalWrite(led1,HIGH);
 s1_occupied=true;
 status1=1;
 c=1;
 PublishData(status1,status2,status3);
 }
 else if(data3=="{\"command\":\"Release slot-1\"}") {
 Serial.println(data3);
 digitalWrite(led1,LOW);
 s1_occupied=false;
 status1=0;
 c=0;
 PublishData(status1,status2,status3);
 }
 else if(data3=="{\"command\":\"Reserve slot-2\"}") {
 Serial.println(data3);
 digitalWrite(led2,HIGH);
 s2_occupied=true;
 status2=1;
 d=1;
 PublishData(status1,status2,status3);
 }
 else if(data3=="{\"command\":\"Release slot-2\"}") {
 Serial.println(data3);
 digitalWrite(led2,LOW);
 s2_occupied=false;
 status2=0;
 d=0;
 PublishData(status1,status2,status3);
 }
```

```
digitalWrite(led3,HIGH);
s3_occupied=true;
status3=1;
e=1;
PublishData(status1,status2,status3);
}
else if(data3=="{\"command\":\"Release slot-3\"}") {
Serial.println(data3);
digitalWrite(led3,LOW);
s3_occupied=false;
status3=0;
e=0;
PublishData(status1,status2,status3);
}
```