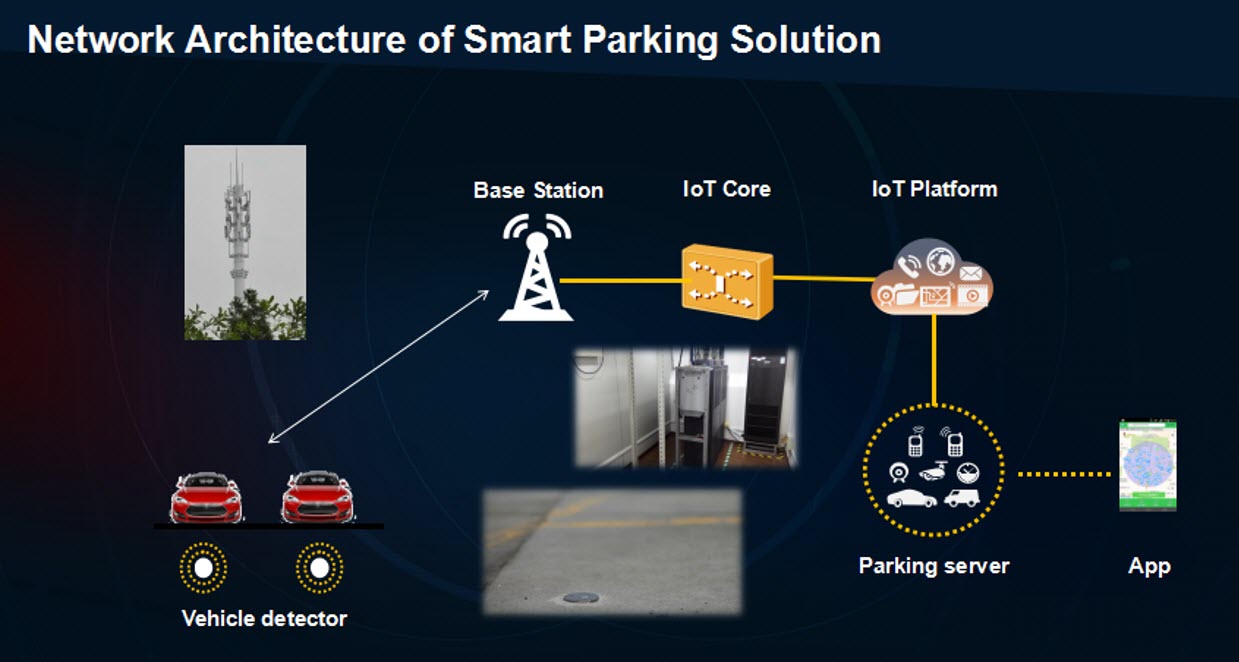
SMART PARKING

INTRODUCTION

The concept of Internet of Things (IoT) started with things with identity communication devices. The devices could be tracked, controlled or monitored using remote computers connected through Internet. IoT extends the use of Internet providing the communication, and thus inter-network of the devices and physical objects, or ‘Things’. The two prominent words in IoT are “internet” and “things”. Internet means a vast global network of connected servers, computers, tablets and mobiles using the internationally used protocols and connecting systems. Internet enables sending, receiving, or communicating of information. Thing in English has number of uses and meanings. Dictionary meaning of ‘Thing’ is a term used to reference to a physical object, an action or idea, situation or activity, in case when we do not wish to be precise. IoT, in general consists of inter-network of the devices and physical objects, number of objects can gather the data at remote locations and communicate to units managing, acquiring, organizing and analyzing the data in the processes and services. It provides a vision where things (wearable, watch, alarm clock, home devices, surrounding objects with) become smart and behave alive through sensing, computing and communicating by embedded small devices which interact with remote objects or persons through connectivity. The scalable and robust nature of Cloud computing is allowing developers to create and host their applications on it. Cloud acts as a partner for IoT as it acts as a platform where all the sensor data can be stored and accessed from remote locations. These factors gave rise to the amalgamation of both technologies thus leading to the formation of a new technology called Cloud of Things(CoT). In CoT the things(nodes) could be accessed, monitored and controlled from any remote location through the cloud. Due to high scalability in cloud any number of node could be added or removed from the IoT system on a real time basis. In simple terms IoT can be explained in form of an equation stating:

**Physical Object + Controller, Sensor and Actuators + Internet = Internet of Things**



STEPS

Below are the steps that a driver needs to follow in order to park its car using our parking system.

Step 1: Install the smart parking application on your mobile device.

Step 2: With the help of the mobile app search for a parking area on and around your destination.

Step 3: Select a particular parking area.

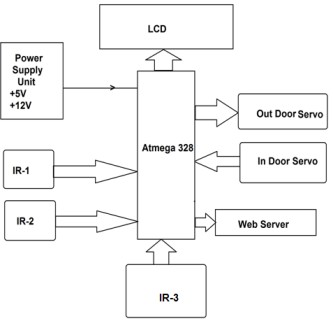
Step 4: Browse through the various parking slots available in that parking area.

Step 5: Select a particular parking slot.

Step 6: Select the amount of time (in hours) for which you would like to park your car for.

Step 7: Pay the parking charges either with your e-wallet or your credit card.

Step 8: Once you have successfully parked your car in the selected parking slot, confirm your occupancy using the mobile application.



Creating a Smart Parking project for ESP32 on the Wokwi platform involves using the ESP32 microcontroller to detect and manage parking spaces, and then visualizing the data on a virtual interface provided by Wokwi. Here's a step-by-step guide on how to create such a project:

\*\*Components Needed:\*\*

1. ESP32 development board

2. Ultrasonic distance sensors (HC-SR04) for each parking space

3. Breadboard and jumper wires

4. Wokwi virtual simulator (https://wokwi.com/)

\*\*Project Steps:\*\*

1. \*\*Hardware Setup:\*\*

a. Connect the HC-SR04 ultrasonic sensors to your ESP32 board. You will need one sensor per parking space.

b. Wire the HC-SR04 sensors as follows:

- VCC to 5V on ESP32

- GND to GND on ESP32

- Trig to a digital GPIO pin on ESP32 (e.g., GPIO2)

- Echo to another digital GPIO pin on ESP32 (e.g., GPIO4)

c. Connect all the sensors in the same way, one for each parking space you want to monitor.

2. \*\*Programming:\*\*

a. Write an Arduino sketch for the ESP32 that reads the distance data from the ultrasonic sensors.

```cpp

#include <Ultrasonic.h>

Ultrasonic sensor1(GPIO\_TRIGGER1, GPIO\_ECHO1);

Ultrasonic sensor2(GPIO\_TRIGGER2, GPIO\_ECHO2);

// Add more sensors if needed

void setup() {

Serial.begin(115200);

}

void loop() {

long distance1 = sensor1.read();

long distance2 = sensor2.read();

// Read distances from more sensors if needed

// Process distance data and manage parking spaces here

delay(1000); // Delay for better readability

}

```

b. In the loop function, process the distance data from each sensor to determine whether a parking space is occupied or vacant. You can set a threshold distance to decide when a space is occupied.

c. You may want to use a data structure to keep track of the parking space status, e.g., an array of boolean values.

3. \*\*Visualization:\*\*

a. Go to the Wokwi platform (https://wokwi.com/) and create an account if you haven't already.

b. Create a new project and select the ESP32 as your target board.

c. Import the Arduino sketch you created earlier into the Wokwi editor.

d. Use the virtual interface provided by Wokwi to display the parking space status. You can use LEDs or any other graphical elements to represent the parking spaces.

4. \*\*Testing:\*\*

a. Simulate the project on Wokwi and observe how the parking space status changes based on the simulated distance measurements.

b. Fine-tune your code and interface as needed to ensure it works correctly.

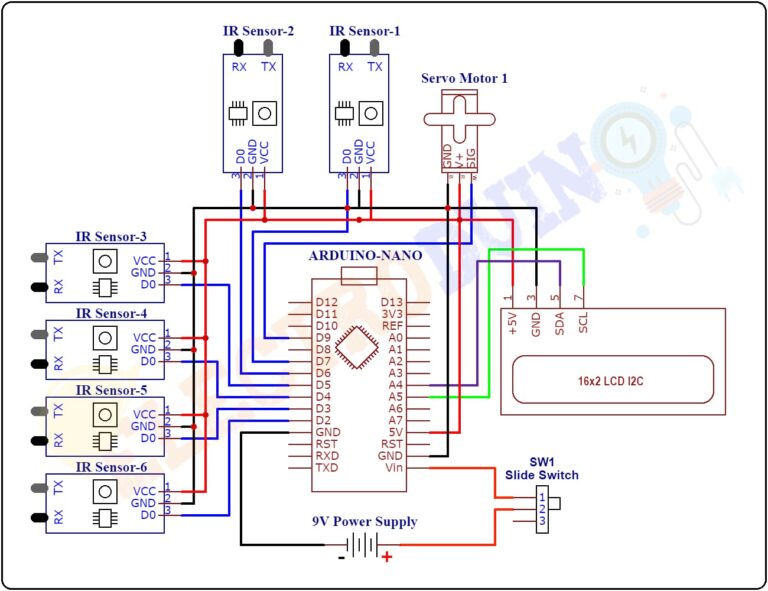
5. \*\*Deployment:\*\*

a. Once your Smart Parking project works as expected in the virtual simulator, you can deploy it to a physical ESP32 board and connect it to real sensors in a parking area.

6. \*\*Enhancements:\*\*

Depending on your project's requirements, you can add features such as mobile app integration for real-time parking updates, data logging, and alerts when parking spaces are full or vacant.

Remember to refer to the ESP32 and HC-SR04 datasheets and the Wokwi documentation for detailed information on programming and using these components in your project.



.

DEVELOPMENT PART-II

Connecting the Hardware in Wokwi

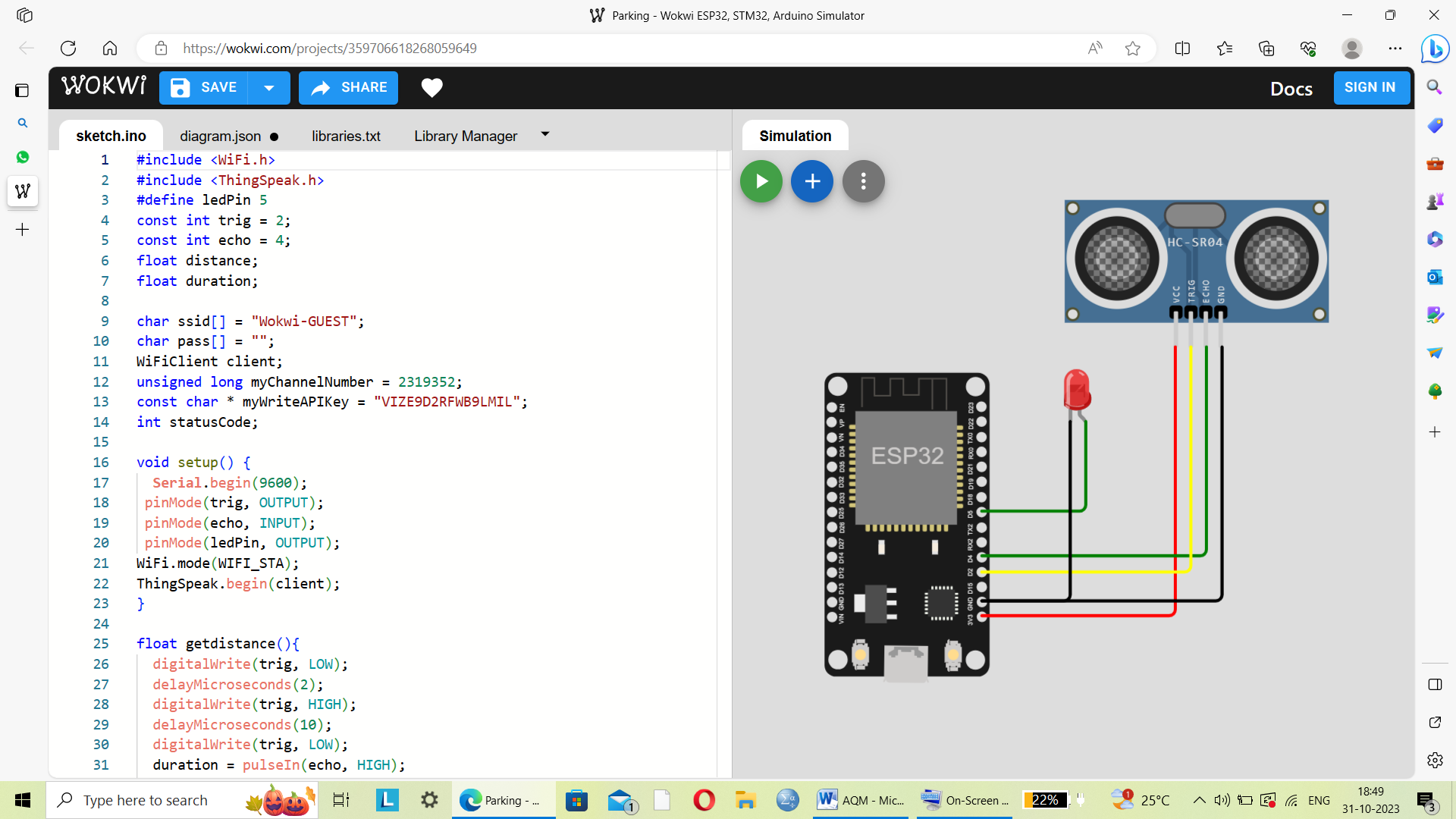
1. 1. In the Wokwi simulator, you can add components like the ESP32,

2. Connect the components using virtual jumper wires.

3. Ultrasonic distance senson connect to the pin on the ESP32.

4.let light connect to the esp32 .

HARDWARE CONNECTION



SOURCE CODE:

#include <WiFi.h>

#include <ThingSpeak.h>

#define ledPin 5

const int trig = 2;

const int echo = 4;

float distance;

float duration;

char ssid[] = "Wokwi-GUEST";

char pass[] = "";

WiFiClient client;

unsigned long myChannelNumber = 2319352;

const char \* myWriteAPIKey = "VIZE9D2RFWB9LMIL";

int statusCode;

void setup() {

Serial.begin(9600);

pinMode(trig, OUTPUT);

pinMode(echo, INPUT);

pinMode(ledPin, OUTPUT);

WiFi.mode(WIFI\_STA);

ThingSpeak.begin(client);

}

float getdistance(){

digitalWrite(trig, LOW);

delayMicroseconds(2);

digitalWrite(trig, HIGH);

delayMicroseconds(10);

digitalWrite(trig, LOW);

duration = pulseIn(echo, HIGH);

return {duration\*0.034/2};

}

void loop() {

connectToCloud();

writeData();

distance = getdistance();

if (distance<200){

digitalWrite(ledPin, HIGH);

Serial.println("Parking Space Occupied");

}

else{

Serial.println("Parking Space Available");

digitalWrite(ledPin, LOW);

}

delay(5000);

}

void connectToCloud(){

if(WiFi.status() != WL\_CONNECTED) {

Serial.print("Attempting to connect");

while(WiFi.status() != WL\_CONNECTED) {

WiFi.begin(ssid, pass);

for(int i=0;i<5;i++) {

Serial.print(".");

delay(5000);

}

}

Serial.println("\nConnected.");

}

}

void writeData(){

distance = getdistance();

ThingSpeak.setField(1, distance);

statusCode = ThingSpeak.writeFields(myChannelNumber,myWriteAPIKey);

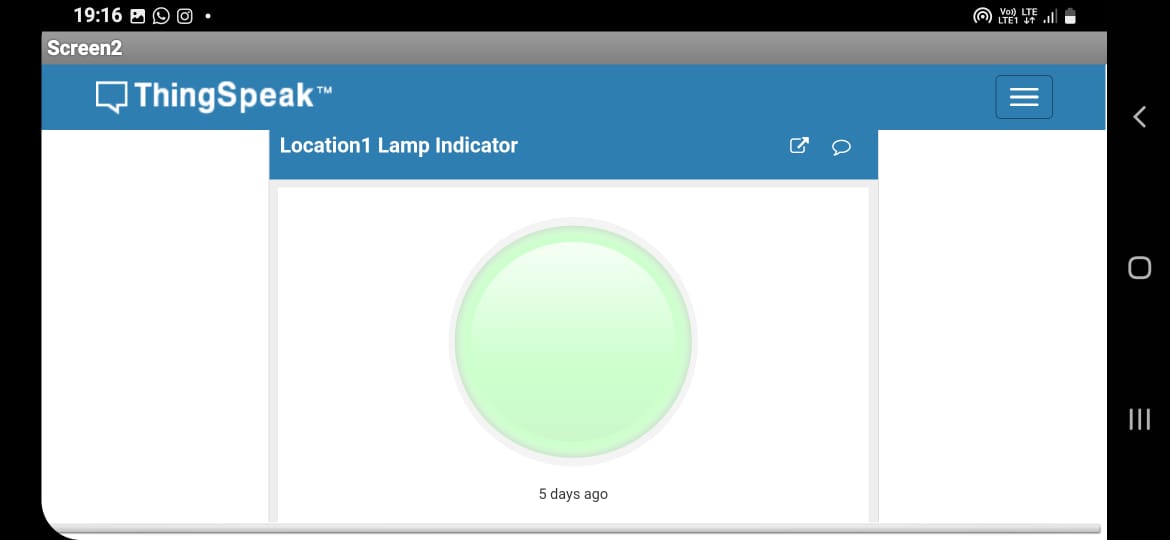
if(statusCode == 200) //successful writing code

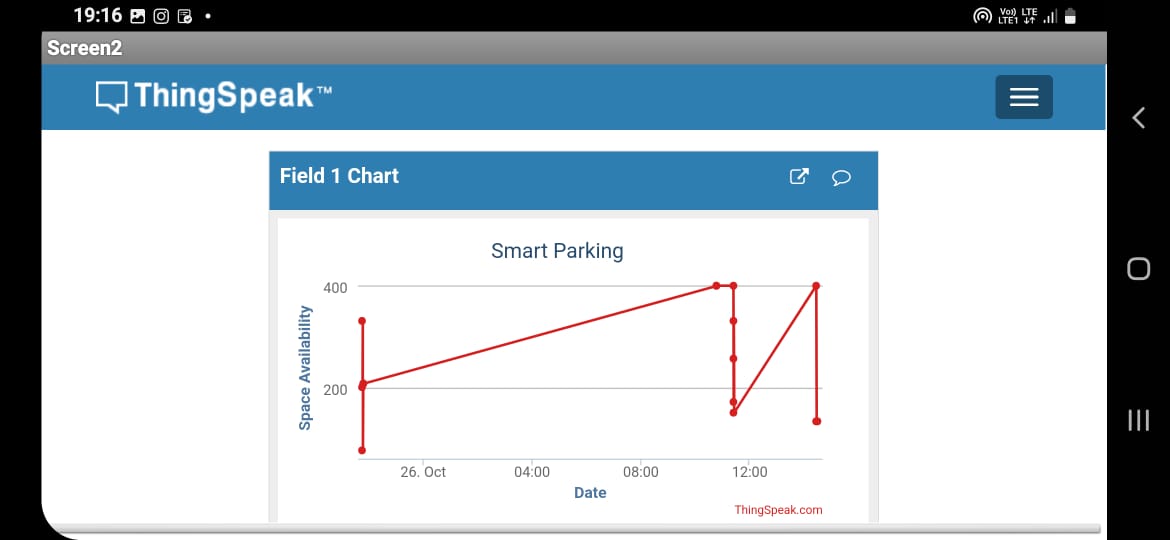
Serial.println("Channel update successful.");

else

Serial.println("Problem Writing data. HTTP error code :" +

String(statusCode));

}



DEVELOP APK USING MIT APP INVENTOR

For every simulation in the wokwi platform the data can be update into personal channel created in the Thingspeak. We can use the data to know the difference level daily update and also live stream the data into the SM Interface using MIT APP INVENTOR

Using MIT app inventor we have to create app that can be named as SMART PARKING

It can be used to monitor and regular update from the cloud system

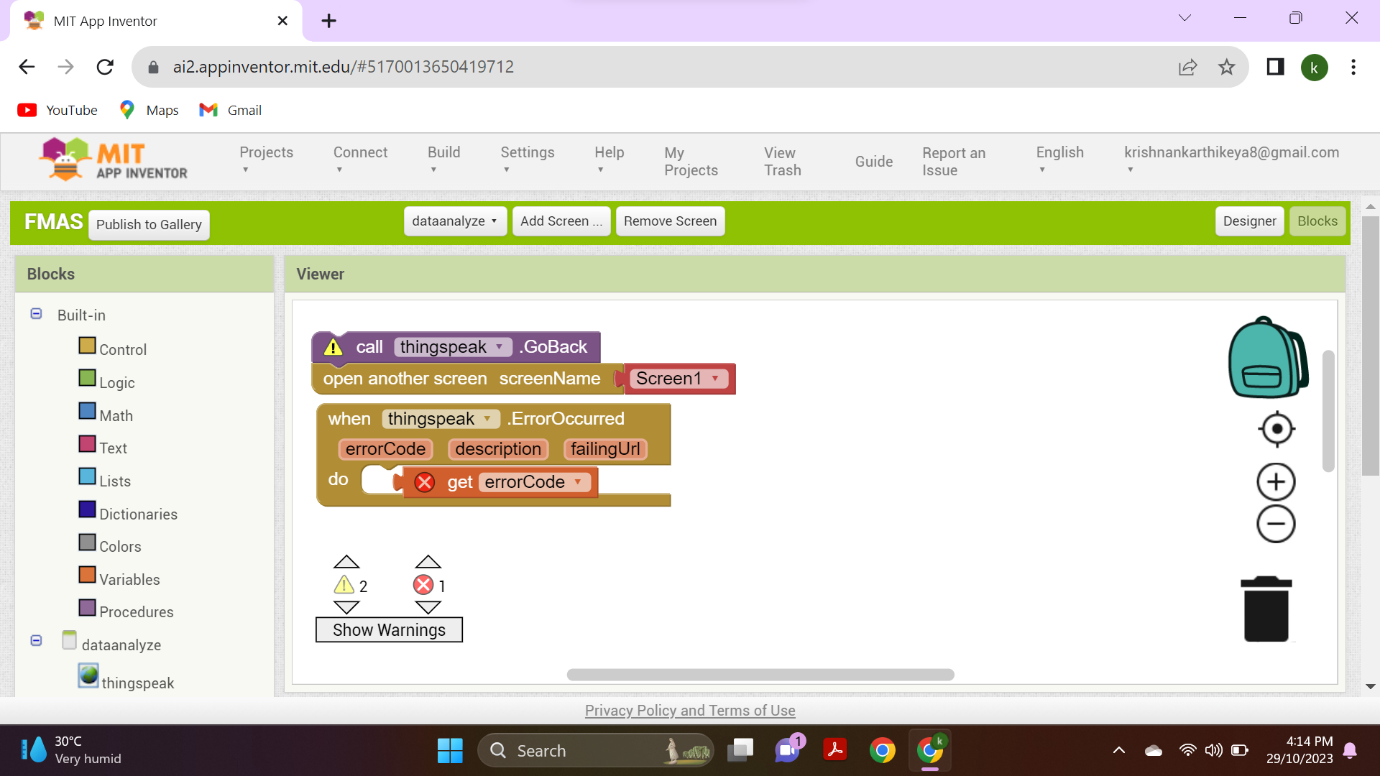
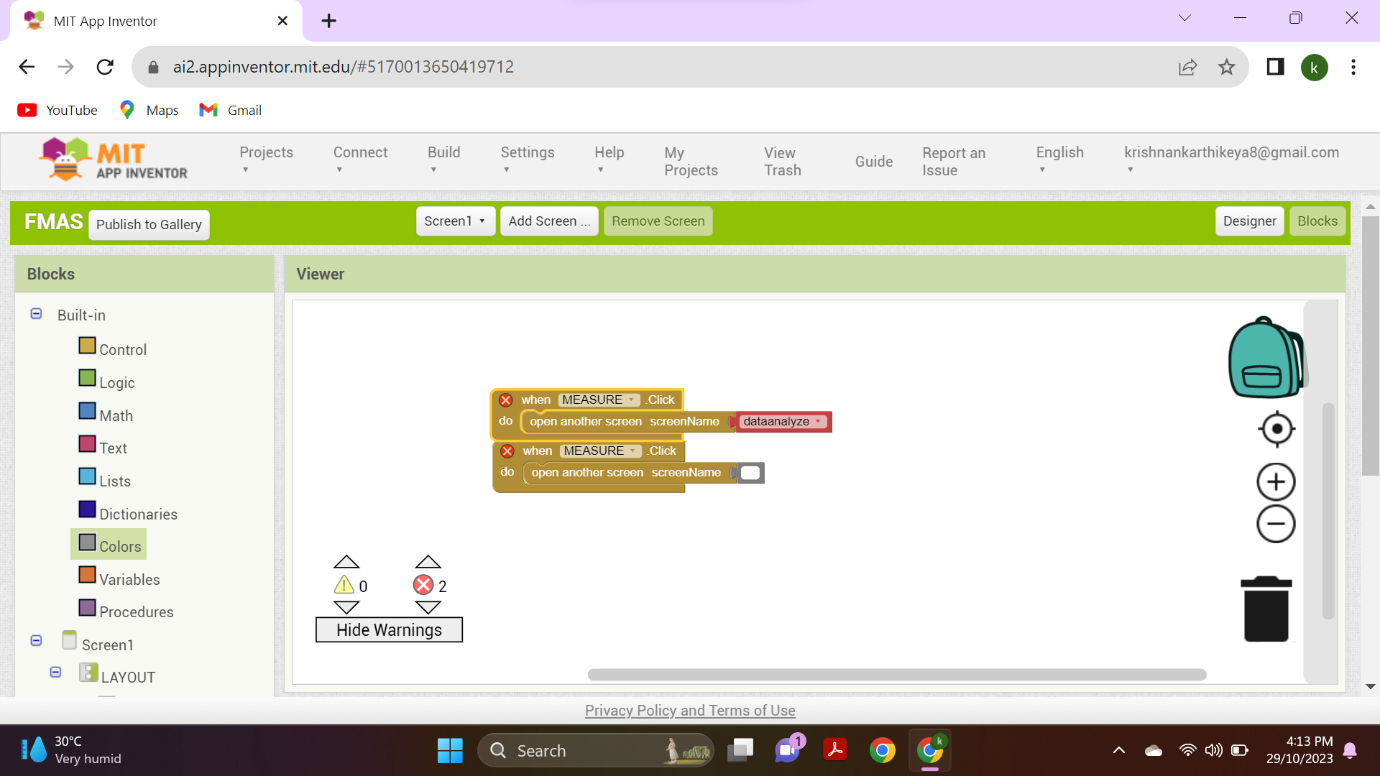
AQM contains two screen

SCREEN 1

It is the open Desktop for the SMART PARKING

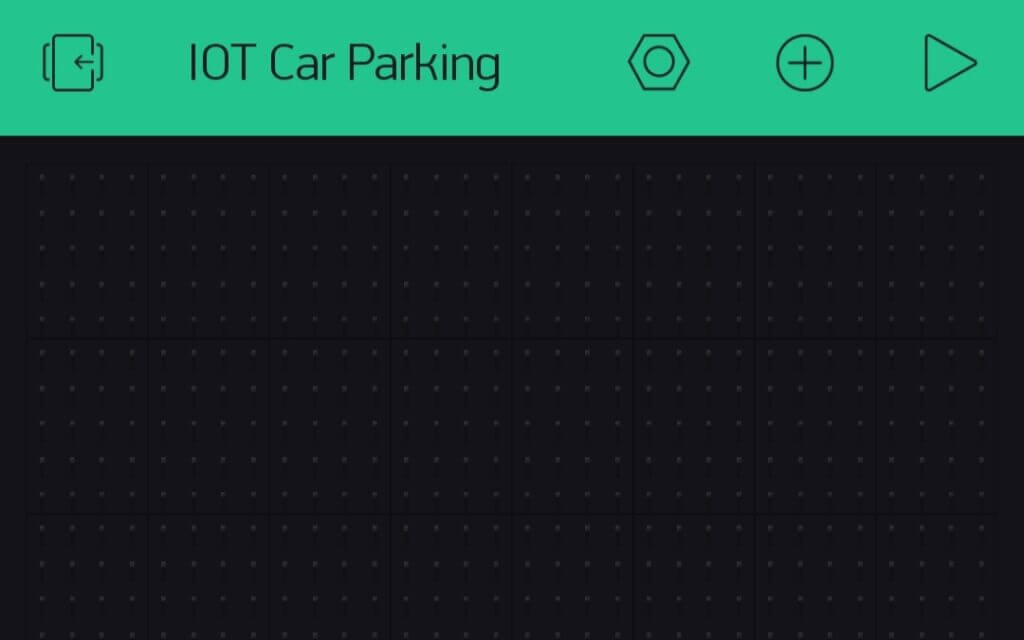
DATA ANALYST

It is used to collect the data from the cloud and provide alert message to the device

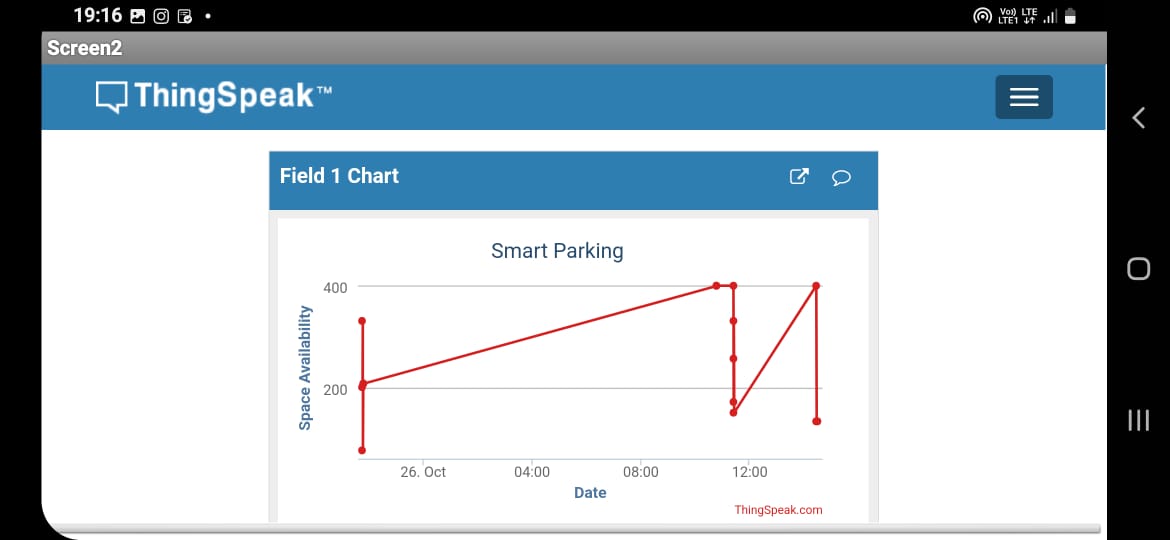


INSTALL THE APK MODE INTO THE MOBILE

1. USER INTERFACE



2.DATA ARRIVED FROM THINGSPEAK CONNECTED THROUGH THE MIT APP INVENTOR .



CONCLUSION

The concept of Smart Cities have always been a dream for humanity. Since the past couple of years large advancements have been made in making smart cities a reality. The growth of Internet of Things and Cloud technologies have give rise to new possibilities in terms of smart cities. Smart parking facilities and traffic management systems have always been at the core of constructing smart cities. In this paper, we address the issue of parking and present an IoT based Cloud integrated smart parking system. The system that we propose provides real time information regarding availability of parking slots in a parking area. Users from remote locations could book a parking slot for them by the use of our mobile application. The efforts made in this paper are indented to improve the parking facilities of a city and thereby aiming to enhance the quality of life of its people