ENVIRONMENTAL MONITORING USING PARKING SYSTEM USNG IOT

Phase 5 project submission

Team Members :

Vinisha M - 921021104055

Punitha S - 921021104035

Swathi S.J - 921021104050

Ruthranayaki D - 921021104043

INTRODUCTION :

Environmental monitoring using parking systems is a new and innovative approach to collecting data on environmental conditions in urban areas. By installing sensors on parking systems, researchers and city planners can gather information on a variety of environmental factors, including air quality, temperature, humidity, and noise levels.

Environmental monitoring data from parking systems can be used to:

- Track air quality trends and identify areas with high levels of pollution
- Monitor the impact of urban heat islands and develop strategies to cool cities
- Identify areas with high levels of noise pollution and develop mitigation strategies
- Track the impact of climate change on urban areas

In addition to its environmental benefits, environmental monitoring using parking systems can also improve the efficiency and effectiveness of parking management. For example, sensors can be used to track parking occupancy in real time, which can help drivers find available parking spaces more quickly and easily. Sensors can also be used to detect illegal parking and enforce parking regulations.

Overall, environmental monitoring using parking systems is a promising new approach to improving the urban environment and making cities more livable. As the technology continues to develop and become more affordable, environmental monitoring using parking systems is expected to become more widespread in cities around the world.

Project Description:

- Develop a system for environmental monitoring using parking systems
- Deploy the system in a pilot city
- Collect data on a variety of environmental factors
- Analyze the data and identify trends and areas of concern
- Develop strategies to improve the urban environment based on the data

The "Environmental Monitoring in Parking System" project aims to enhance the sustainability and user experience of parking facilities by implementing a comprehensive environmental monitoring system. This system will utilize a network of sensors and advanced technologies to collect, analyze, and manage various environmental parameters within and around parking areas.

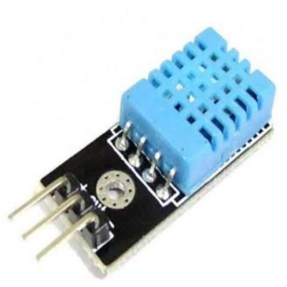
Components :

Hardware :

Sensors:

Sensors are used to measure environmental conditions, such as temperature, humidity, air quality, and noise levels.

Digital Humidity and Temperature sensor



Actuators: Actuators are used to control physical devices, such as fans, pumps, and lights. They are used to implement the actions dictated by the microcontroller. For example, if the temperature in the parking lot gets too high, the microcontroller can send a command to the actuator to turn on the fans.

Communication modules: Communication modules are used to transmit data between the microcontroller and other devices, such as a central server or a mobile app. This allows users to remotely monitor the environmental conditions in the parking lot and to receive alerts if there are any problems.

Cameras: Cameras can be used to monitor the security of the parking lot and to identify vehicles that are illegally parked.

Displays: Displays can be used to show users the environmental conditions in the parking lot, as well as other information, such as the availability of parking spaces.

ESP32 Boards:

Data Loggers : Data loggers are an essential component of environmental monitoring systems in parking systems. They are used to collect data from sensors and store it for later analysis. Data loggers are typically used to measure environmental conditions such as temperature, humidity, air quality, and noise levels.

User Interface :The user interface (UI) allows users to interact with the system. The UI should be easy to use and navigate, and it should provide users with all of the information they need to effectively monitor the environmental conditions in the parking lot.

Software :

Data acquisition software: This software is responsible for collecting data from the sensors and storing it in a database.

Data analysis software: This software is used to analyze the collected data and identify trends and patterns.

Reporting software: This software is used to generate reports on the environmental conditions in the parking lot.

Visualization software: This software is used to create visualizations of the data, such as charts and graphs.

Alarm software: This software is used to send alerts to users if there are any problems with the environmental conditions in the parking lot.

In addition to these basic components, some parking-based environmental monitoring systems may also include additional software components, such as:

Prediction software: This software is used to predict future environmental conditions in the parking lot. This information can be used to take preventive measures, such as increasing ventilation or reducing traffic flow.

Optimization software: This software is used to optimize the operation of the parking system to reduce environmental impact.

Program:

```
int value_sensor = 0;
void setup ()
{
  pinMode(A1, INPUT);
  Serial.begin(9600);
  pinMode(6, OUTPUT);
}
void loop()
{
  // Gas senor with buzzer
```

```
 value_sensor = analogRead(A1);
 Serial.println(value_sensor);
 if (value_sensor > 200) {
   tone(6, 523, 1000); // play tone 60 (C5 = 523 Hz)
 }
 delay(10);
// Delay a little bit to improve simulation performance
```

Main program :

```
# include <WiFi.h>
#include <DHT.h>
#include <FirebaseArduino.h>
#include <PubSubClient.h>
// Replace with your Wi-Fi credentials
const char* ssid = "Wokwi-Guest";
const char* password = "YourWiFiPassword";
// Your Firebase project credentials
const char* firebaseHost = "https://ibmgr1-default-rtdb.firebaseio.com/ibmgr1";
const char* firebaseAuth = "AIzaSyBy9qLB8-kIxS1Kftv3I6pY3aVWmOI83w8";
const int DHTPin = 4; // Pin to which the DHT sensor is connected
DHT dht(DHTPin, DHT22);
// MQTT Broker
const char* mqttServer = "ed342ccf55c1484eb534c8c92861048b.s2.eu.hivemq.cloud"; //
Replace with your MQTT broker address
const int mqttPort = 8883;
const char* mqttUser = "aiyengar";
const char* mqttPassword = "Mh12hn4226!!!";
const char* mqttTopic = "/ESP32";
WiFiClient wifiClient;
PubSubClient mqttClient(wifiClient);
void setup() {
  Serial.begin(115200);
```

```
 WiFi.begin(ssid, password);
 while (WiFi.status() != WL_CONNECTED) {
   delay(1000);
   Serial.println("Connecting to WiFi...");
 }

 Serial.println("Connected to WiFi");
// Initialize Firebase
 Firebase.begin(firebaseHost, firebaseAuth);

 // MQTT setup
 mqttClient.setServer(mqttServer, mqttPort);
 mqttClient.setCallback(mqttCallback);
 // Connect to MQTT broker
 if (mqttClient.connect("ESP32Client", mqttUser, mqttPassword)) {
   Serial.println("Connected to MQTT broker");
 }
}
void loop() {
 float humidity = dht.readHumidity();
 float temperature = dht.readTemperature();
 if (!isnan(humidity) && !isnan(temperature)) {
   Serial.print("Temperature: ");
   Serial.print(temperature);
   Serial.print(" °C, Humidity: ");
   Serial.print(humidity);
   Serial.println(" %");
   // Firebase
   Firebase.pushFloat("/environment/temperature", temperature);
   Firebase.pushFloat("/environment/humidity", humidity);
```

```
    // MQTT publish
    String tempPayload = String(temperature);
    String humidityPayload = String(humidity);
    mqttClient.publish((String(mqttTopic) + "/temperature").c_str(), tempPayload.c_str());
    mqttClient.publish((String(mqttTopic) + "/humidity").c_str(), humidityPayload.c_str());
  } else {
    Serial.println("Failed to read from DHT sensor");
  }
  mqttClient.loop();
  delay(60000); // Send data every 60 seconds
}
void mqttCallback(char* topic, byte* payload, unsigned int length) {
  // Handle MQTT subscription messages if needed
}
```

JSON :

```
{
  "version": 1,
  "author": "Ashok Iyengar",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-breadboard-mini", "id": "bb1", "top": 56.2, "left": -136.8, "attrs": {} },
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -43.3, "left": 110.2, "attrs": {} },
    { "type": "wokwi-dht22", "id": "dht1", "top": 29.1, "left": -101.4, "attrs": {} }
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "dht1:VCC", "bb1:5b.g", "", [ "$bb" ] ],
    [ "dht1:SDA", "bb1:6b.g", "", [ "$bb" ] ],
    [ "dht1:NC", "bb1:7b.g", "", [ "$bb" ] ],
    [ "dht1:GND", "bb1:8b.g", "", [ "$bb" ] ],
    [ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ],
    [ "dht1:GND", "esp:GND.1", "black", [ "v-38.4", "h268.8" ] ],
    [ "dht1:SDA", "esp:D4", "green", [ "v-67.2", "h288.1" ] ],
    [ "dht1:NC", "esp:D4", "green", [ "v-67.2", "h278.5" ] ]
  ],
  "dependencies": {}
```

<u>Libraries :</u>

```
# Wok Wi Library List
# See https://docs.wokwi.com/guides/libraries
# Automatically added based on includes:
DHT sensor library
WiFi
DHT22
PubSubClient
MQTT
Firebase Arduino based on WiFi101
Firebase Arduino Client Library for ESP8266 and ESP32
ESP8266 Firebase
```

<u>To continue building the project by developing the environmental monitoring platform, we need to:</u>

Identify the specific environmental parameters that we want to monitor. This could include air quality, water quality, soil quality, noise levels, vibration levels, and/or other factors.

Select the appropriate sensors and devices to collect data on these parameters. There are a wide variety of environmental sensors available, so it is important to choose the ones that are most relevant to the specific parameters that we want to monitor and the environment where we will be deploying them.

Design and implement a system for collecting, transmitting, and storing the sensor data. This system may include a variety of hardware and software components, such as data acquisition systems, edge computing devices, cloud computing services, and data management software.

Develop data visualization and analytics tools to analyse the sensor data and identify trends and patterns. This will allow us to gain insights into the environmental conditions and identify any potential problems.

Develop a user interface for the environmental monitoring platform. This interface should allow users to easily view the sensor data, historical data, and analytics results.

1. Identify the specific environmental parameters that we want to monitor:

2. Select the appropriate sensors and devices to collect data on these parameters:

Some factors:

Accuracy, Precision, Range, Sensitivity, Durability, Cost

3. Design and implement a system for collecting, transmitting, and storing the sensor data:

4. Develop data visualization and analytics tools to analyse the sensor data and identify trends and patterns:

5. Develop a user interface for the environmental monitoring platform.

This interface should allow users to easily view the sensor data, historical data, and analytics results

Detailed overview:

IoT devices:

The IoT devices can be any type of device that can collect temperature and humidity data and send it over the internet. Examples of IoT devices that can be used for this purpose include:

Microcontrollers (e.g., Arduino, Raspberry Pi)

Single-board computers (e.g., Beagle Bone Black, NVIDIA Jetson Nano)

Dedicated environmental sensors (e.g., DHT11, DHT22, SHT31)

Weather stations

communication protocols:

MQTT: MQTT is a lightweight messaging protocol that is well-suited for IoT devices because it is efficient and scalable.

CoAP: CoAP is a constrained protocol that is designed for devices with limited resources, such as battery-powered sensors.

XMPP: XMPP is an extensible messaging protocol that can be used for a variety of applications, including IoT.

Data storage:

The platform will need to store the temperature and humidity data so that it can be displayed to users in real time. A database can be used to store the data. Some popular databases for IoT applications include:

Influx DB: Influx DB is a time series database that is designed to store and query large amounts of time-stamped data.

Timescale DB: Timescale DB is a time series database that is built on top of PostgreSQL.

MongoDB: MongoDB is a NoSQL database that is well-suited for storing and querying data from a variety of sources, including IoT devices.

Data processing:

The platform will need to process the temperature and humidity data to calculate averages, trends, and other insights

Data visualization:

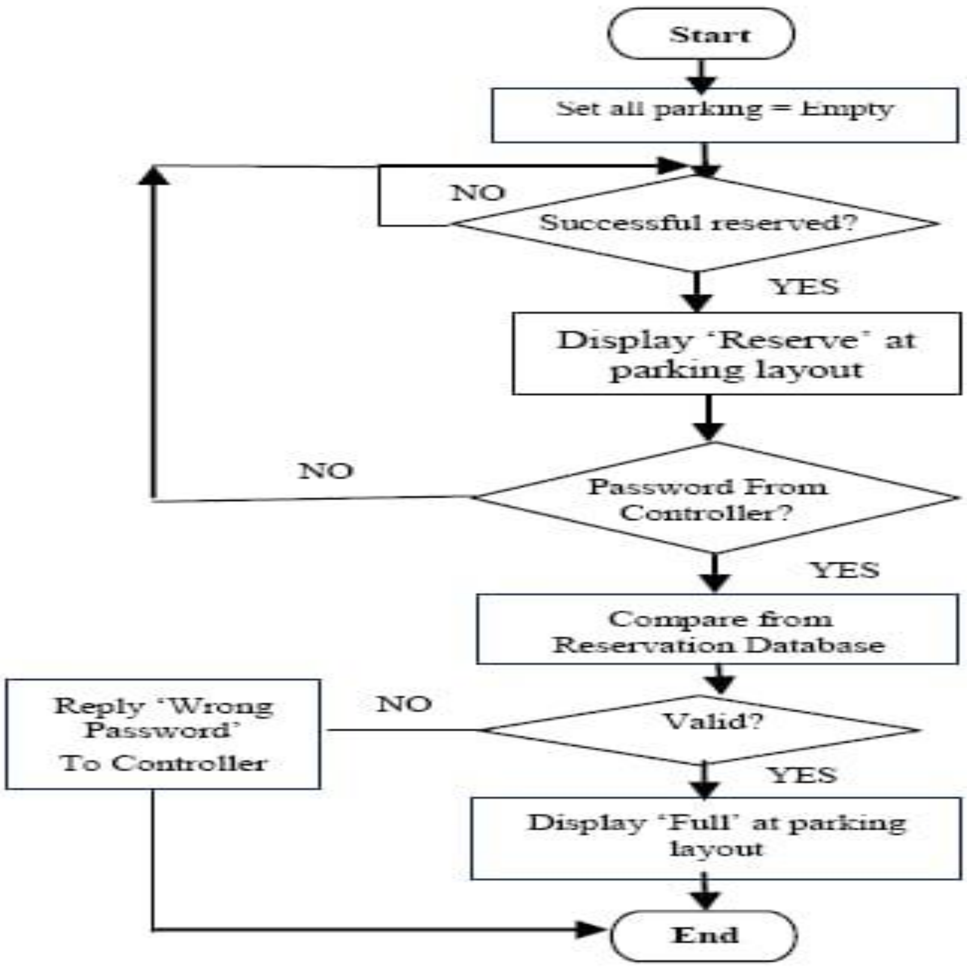
The platform will need to visualize the temperature and humidity data in a way that is easy for users to understand. This can be done using a variety of data visualization tools, such as:

Grafana: Grafana is a popular open-source data visualization platform.

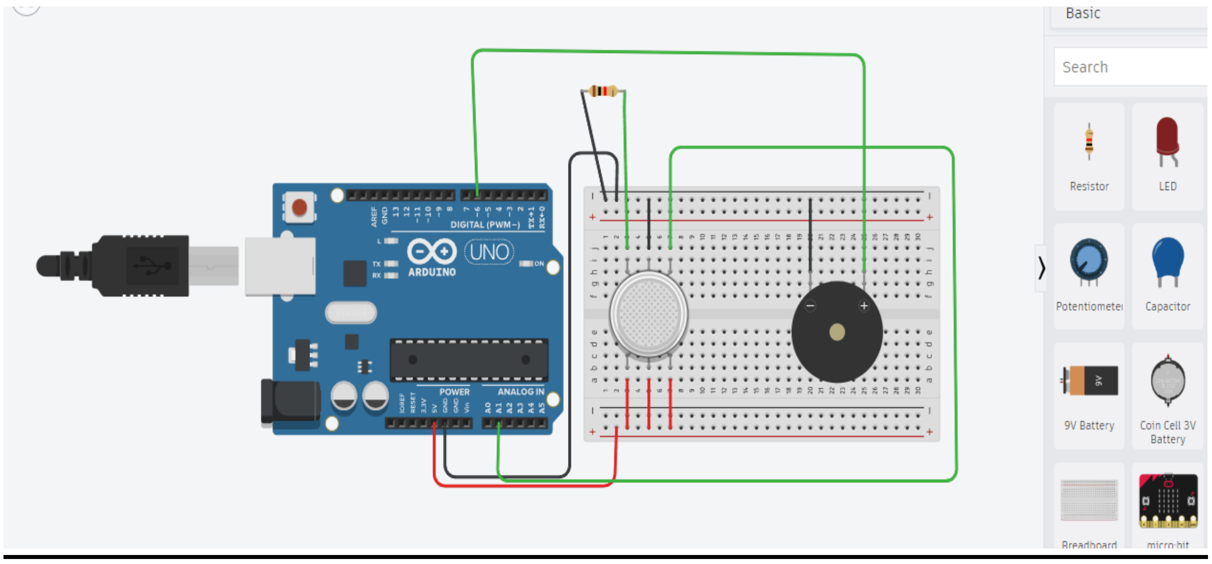
Kibana: Kibana is a data visualization tool that is part of the Elasticsearch stack.

Tableau: Tableau is a commercial data visualization platform.

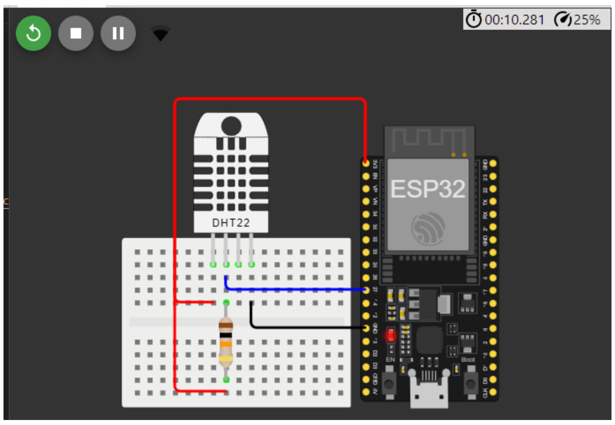
Flow Chart:

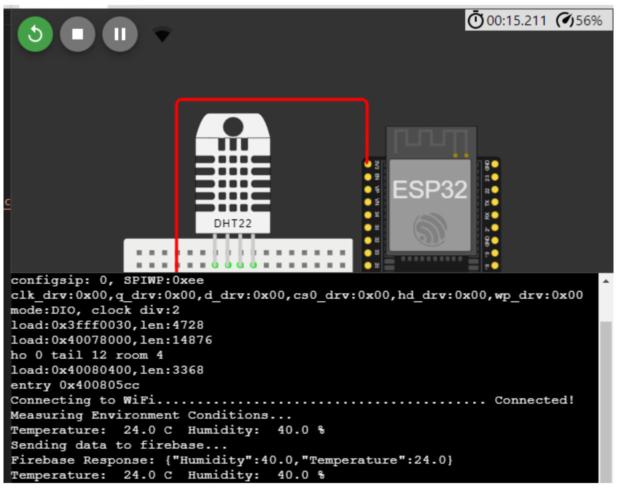

Start
Set all parking = Empty
NO
Successful reserved?
YES
Display 'Reserve' at parking layout
NO
Password From Controller?
YES
Compare from Reservation Database
Reply 'Wrong Password' To Controller
NO
Valid?
YES
Display 'Full' at parking layout
End

Trinket Result:



Wok Wi Stimulation results:





```
configsip: 0, SPIWP:0xee
clk_drv:0x00,q_drv:0x00,d_drv:0x00,cs0_drv:0x00,hd_drv:0x00,wp_drv:0x00
mode:DIO, clock div:2
load:0x3fff0030,len:4728
load:0x40078000,len:14876
ho 0 tail 12 room 4
load:0x40080400,len:3368
entry 0x400805cc
Connecting to WiFi........................................ Connected!
Measuring Environment Conditions...
Temperature:  24.0 C  Humidity:  40.0 %
Sending data to firebase...
Firebase Response: {"Humidity":40.0,"Temperature":24.0}
Temperature:  24.0 C  Humidity:  40.0 %
```

Conclusion:

Environmental monitoring using parking systems is a promising new technology that has the potential to improve the comfort, safety, and environmental sustainability of parking lots. By collecting and analysing data on environmental conditions in parking lots, system owners can identify and address potential problems, such as high temperature, poor air quality, and excessive noise. This information can also be used to optimize the operation of the parking system to reduce energy consumption and environmental impact.

In the future, environmental monitoring using parking systems is expected to become more widely adopted as the technology continues to develop and become more affordable. As more parking lots are equipped with environmental monitoring systems, the benefits of this technology will be realized by a wider range of stakeholders, including parking lot owners, users, and the environment as a whole.

This code snippet simulates a temperature sensor and checks the temperature every 5 minutes. If the temperature goes above a certain threshold (25°C in this case), it prints a message indicating that the temperature is too high. In a real-world implementation, you would replace the temperature simulation code with actual sensor readings and replace the print statements with actions or alerts that are relevant to your parking system's needs.