IOT-Enabled Environmental Monitoring in Parks Project Documentation

objective:

Real-Time Data Collection:

IoT devices enable continuous and real-time monitoring of environmental parameters like air quality, temperature, and water levels.

Early Pollution Detection:

Swift detection of pollutants in air, water, and soil, allowing rapid response to mitigate environmental risks.

Optimized Resource Management:

IOT-driven insights aid in efficient use of resources, minimizing waste and promoting sustainability.

Enhanced Public Health:

Monitoring air pollutants helps in assessing public exposure, leading to strategies for healthier living environments.

Climate Change Understanding:

IOT sensors collect climate data, contributing vital information for climate change studies and adaptation strategies.

Smart Agriculture:

Environmental sensors in agriculture enhance crop management, optimizing irrigation and pesticide use for sustainable farming.

Biodiversity Preservation:

IoT supports wildlife tracking, protecting endangered species and preserving natural habitats.

Disaster Preparedness:

IoT sensors provide early warnings for natural disasters like floods or forest fires, facilitating timely evacuations and reducing risks.

Data-Driven Policies:

Collected IoT data empowers policymakers with evidence, aiding in the formulation of effective environmental regulations and policies.

Promoting Eco-conscious Behavior:

Environmental data accessible through IoT fosters awareness, encouraging individuals and communities to adopt eco-friendly practices Regenerate

by continue the output of phase 3 and 4,here I and my team member developing the webpage to show real time environmental data.

<u>Requirements:</u>

- Wokwi (platform for code)
- Firebase(for hosting)

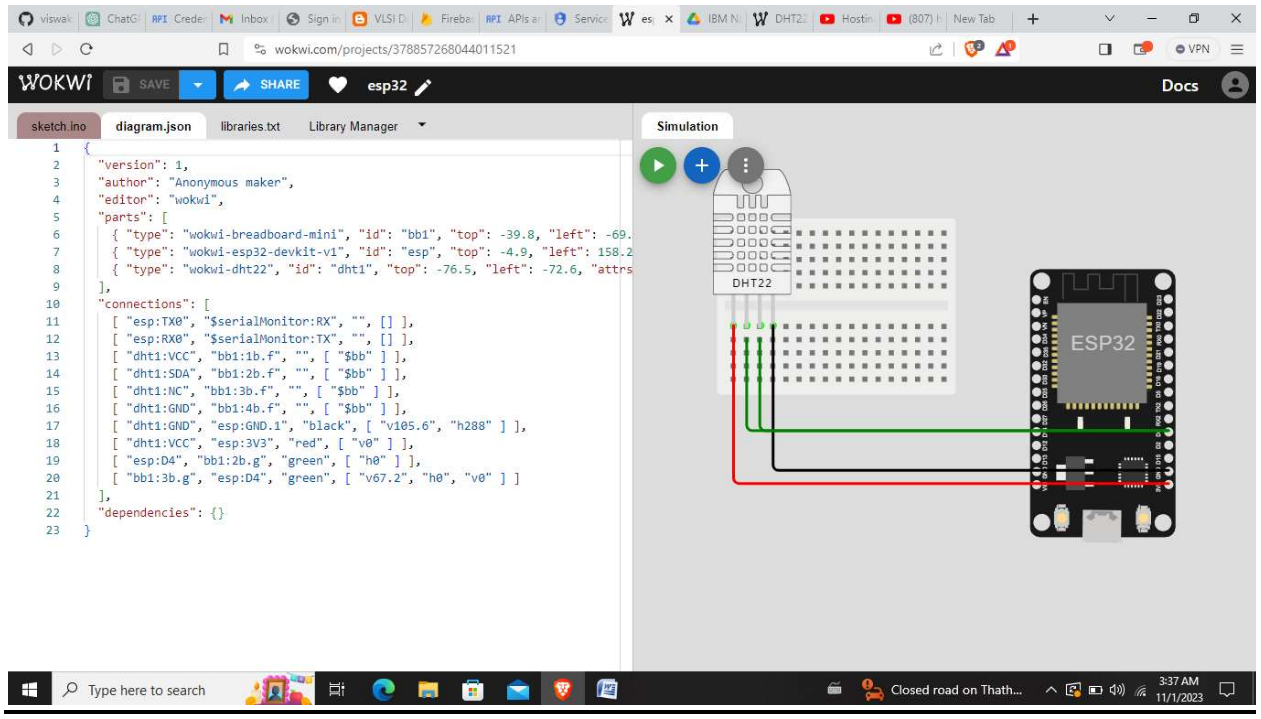
- Node.js
- Google cloud
- Html
- Css (for presentation layer)
- Javascript (function layer)

Wokwi:

Code:

```
#include <WiFi.h>

#include <FirebaseESP32.h>

#include <Adafruit_Sensor.h>

#include <DHT.h>

#define FIREBASE_HOST " https://esp32-environmental-monitoring-default-
rtdb.firebaseio.com"
#define FIREBASE_AUTH "AIzaSyCy2aZiFAz0PYkOEt2Wgwr1raIgn7QV97k"

#define DHTPIN 4 // Pin connected to the DHT22 sensor
#define DHTTYPE DHT22 // DHT sensor type

DHT dht(DHTPIN, DHTTYPE);

const char* ssid = "viswakumaran's xiaomi";
const char* password = NULL;

FirebaseData firebaseData;

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");
  dht.begin();
  Firebase.begin(FIREBASE_HOST);
}

void loop() {
  delay(2000); // Wait for 2 seconds to avoid sending data too frequently

  float humidity = dht.readHumidity();
  float temperature = dht.readTemperature();

  if (isnan(humidity) || isnan(temperature)) {
    Serial.println("Failed to read from DHT sensor");
    return;
  }

  Serial.print("Humidity: ");
  Serial.print(humidity);
  Serial.print("% - Temperature: ");
  Serial.print(temperature);
  Serial.println("°C");

  // Send data to Firebase
  Firebase.setFloat(firebaseData, "humidity", humidity);
  Firebase.setFloat(firebaseData, "temperature", temperature);

  if (Firebase.failed()) {
    Serial.println("Firebase data upload failed");
  } else {
    Serial.println("Firebase data sent successfully");
  }
}
```

Diagram:

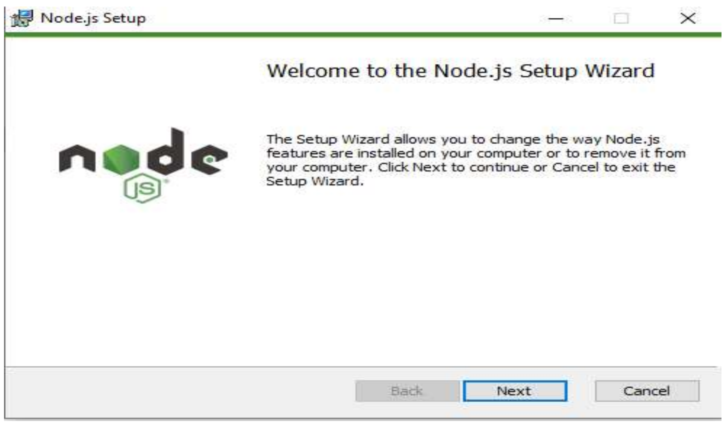


```
{
  "version": 1,
  "author": "Anonymous maker",
  "editor": "wokwi",
  "parts": [
    { "type": "wokwi-breadboard-mini", "id": "bb1", "top": -39.8, "left": -69.
    { "type": "wokwi-esp32-devkit-v1", "id": "esp", "top": -4.9, "left": 158.2
    { "type": "wokwi-dht22", "id": "dht1", "top": -76.5, "left": -72.6, "attrs
  ],
  "connections": [
    [ "esp:TX0", "$serialMonitor:RX", "", [] ],
    [ "esp:RX0", "$serialMonitor:TX", "", [] ],
    [ "dht1:VCC", "bb1:1b.f", "", [ "$bb" ] ],
    [ "dht1:SDA", "bb1:2b.f", "", [ "$bb" ] ],
    [ "dht1:NC", "bb1:3b.f", "", [ "$bb" ] ],
    [ "dht1:GND", "bb1:4b.f", "", [ "$bb" ] ],
    [ "dht1:GND", "esp:GND.1", "black", [ "v105.6", "h288" ] ],
    [ "dht1:VCC", "esp:3V3", "red", [ "v0" ] ],
    [ "esp:D4", "bb1:2b.g", "green", [ "h0" ] ],
    [ "bb1:3b.g", "esp:D4", "green", [ "v67.2", "h0", "v0" ] ]
  ],
  "dependencies": {}
}
```

Above screen shot describe the Json diagram file.

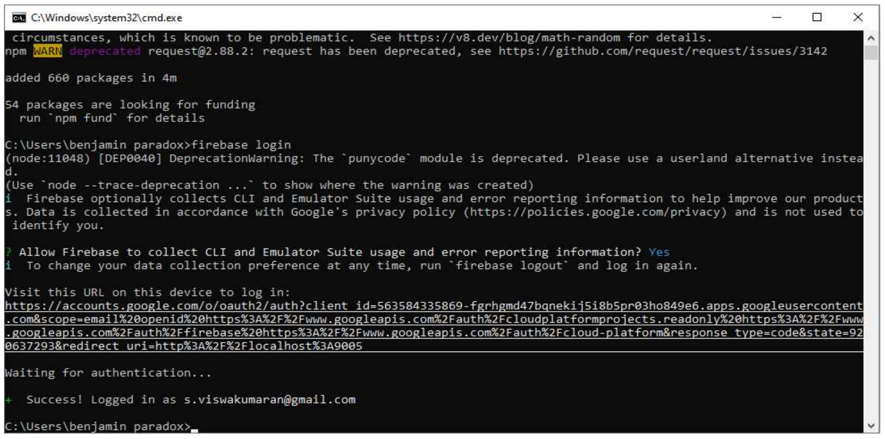
Node js:

Before using the firebase we need node js to install NPM command and install firebase init and connect desktop to firebase and github account

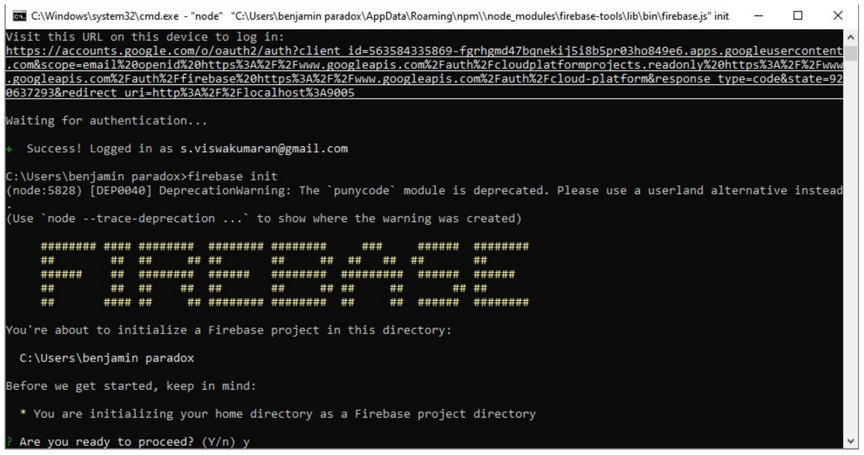


Welcome to the Node.js Setup Wizard

The Setup Wizard allows you to change the way Node.js features are installed on your computer or to remove it from your computer. Click Next to continue or Cancel to exit the Setup Wizard.



```
 circumstances, which is known to be problematic.  See https://v8.dev/blog/math-random for details.
npm WARN deprecated request@2.88.2: request has been deprecated, see https://github.com/request/request/issues/3142

added 660 packages in 4m

54 packages are looking for funding
  run `npm fund` for details

C:\Users\benjamin paradox>firebase login
(node:11048) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instea
d.
(Use `node --trace-deprecation ...` to show where the warning was created)
i  Firebase optionally collects CLI and Emulator Suite usage and error reporting information to help improve our product
s. Data is collected in accordance with Google's privacy policy (https://policies.google.com/privacy) and is not used to
 identify you.

? Allow Firebase to collect CLI and Emulator Suite usage and error reporting information? Yes
i  To change your data collection preference at any time, run `firebase logout` and log in again.

Visit this URL on this device to log in:
https://accounts.google.com/o/oauth2/auth?client_id=563584335869-fgrhgmd47bqnekij5i8b5pr03ho849e6.apps.googleusercontent
.com&scope=email%20openid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww
.googleapis.com%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform&response_type=code&state=92
0637293&redirect_uri=http%3A%2F%2Flocalhost%3A9005

Waiting for authentication...

+  Success! Logged in as s.viswakumaran@gmail.com

C:\Users\benjamin paradox>
```

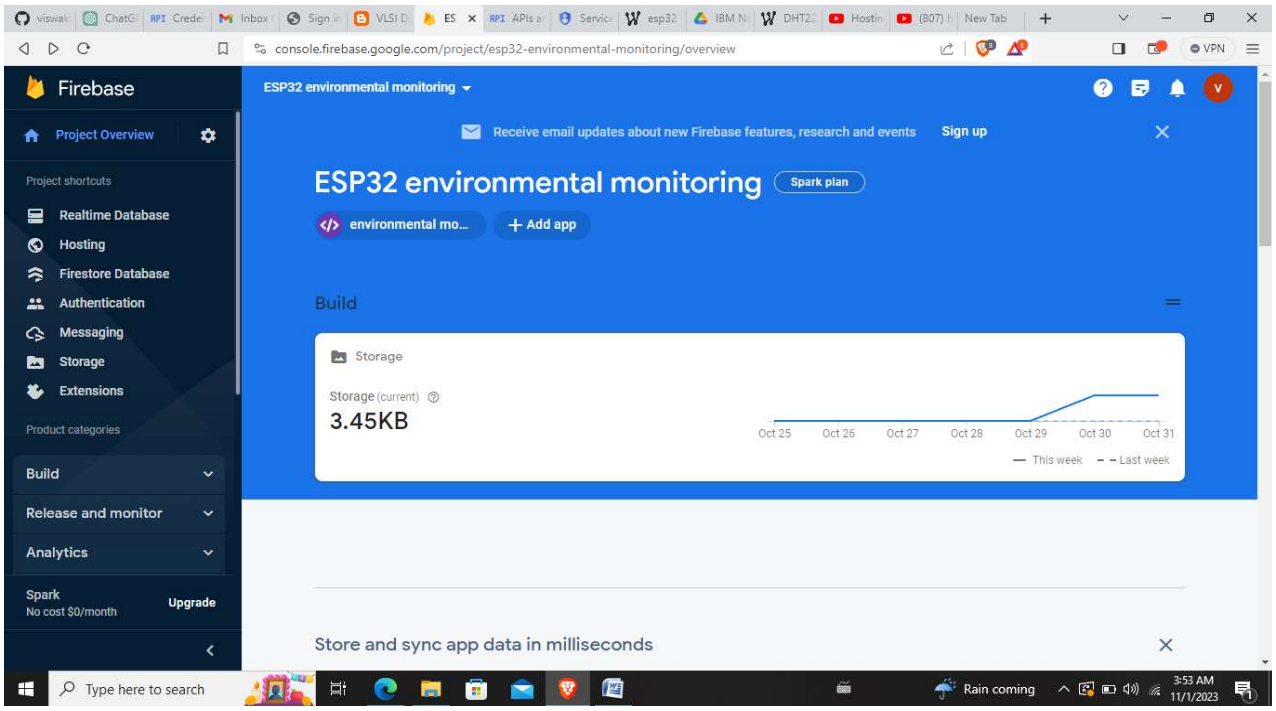

C:\Windows\system32\cmd.exe - "node" "C:\Users\benjamin paradox\AppData\Roaming\npm\\node_modules\firebase-tools\lib\bin\firebase.js" init

```
Visit this URL on this device to log in:
https://accounts.google.com/o/oauth2/auth?client_id=563584335869-fgrhgmd47bqnekij5i8b5pr03ho849e6.apps.googleusercontent
.com&scope=email%20openid%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloudplatformprojects.readonly%20https%3A%2F%2Fwww
.googleapis.com%2Fauth%2Ffirebase%20https%3A%2F%2Fwww.googleapis.com%2Fauth%2Fcloud-platform&response_type=code&state=92
0637293&redirect_uri=http%3A%2F%2Flocalhost%3A9005

Waiting for authentication...

+  Success! Logged in as s.viswakumaran@gmail.com

C:\Users\benjamin paradox>firebase init
(node:5828) [DEP0040] DeprecationWarning: The `punycode` module is deprecated. Please use a userland alternative instead
.
(Use `node --trace-deprecation ...` to show where the warning was created)

     ######## #### ########  ######## ########     ###     ######  ########
     ##        ##  ##     ## ##       ##     ##  ##   ##  ##       ##
     ######    ##  ########  ######   ########  #########  ######  ######
     ##        ##  ##    ##  ##       ##     ## ##     ##       ## ##
     ##       #### ##     ## ######## ########  ##     ##  ######  ########

You're about to initialize a Firebase project in this directory:

  C:\Users\benjamin paradox

Before we get started, keep in mind:

  * You are initializing your home directory as a Firebase project directory

? Are you ready to proceed? (Y/n) y
```

After installing node we can set configuration in local pc and able to work with firebase.

Google firebase:

- We need firebase account for show real-time data base



After login into firebase account , create account with project name ESP32 Environmental monitoring and set web develop app in the given option, we can either use android and iso app development to show real time data on real time database. they generate some key and urls that are

- apikey
- authdomain
- databaseURL
- projected

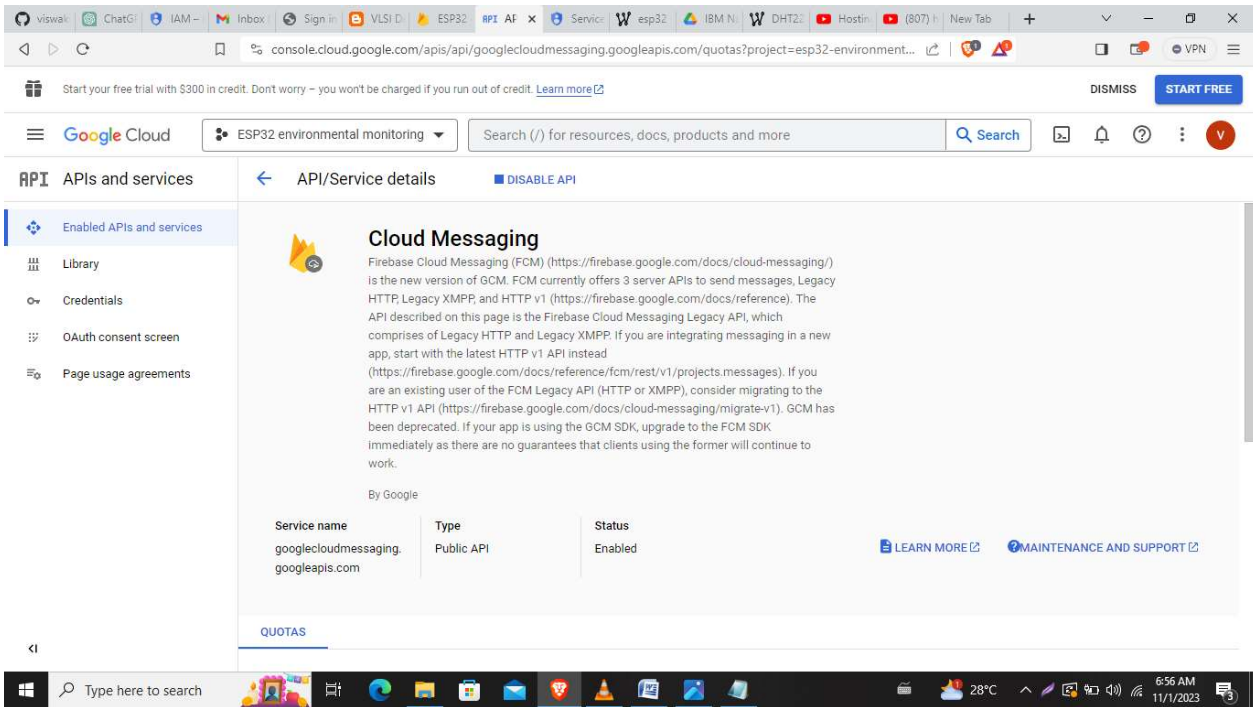
- storage bucket
- messaging sender id
- app id
- measurement id

<u>Google cloud:</u>

We need API to communicate between wokwi and firebase Google cloud provide API key and generate server id and sender id. Simple steps require to login with our Google account.

Open firebase account < project over view < project setting < cloud messaging < enable cloud messaging API.

It will automatically redirect through Google cloud remaining steps are handled by google.


console.cloud.google.com/apis/api/googlecloudmessaging.googleapis.com/quotas?project=esp32-environment...
Start your free trial with $300 in credit. Don't worry – you won't be charged if you run out of credit. Learn more
DISMISS
START FREE
Google Cloud
ESP32 environmental monitoring
Search (/) for resources, docs, products and more
Search
APIs and services
API/Service details
DISABLE API
Enabled APIs and services
Library
Credentials
OAuth consent screen
Page usage agreements
LEARN MORE
MAINTENANCE AND SUPPORT
QUOTAS
Type here to search
28°C
6:56 AM
11/1/2023

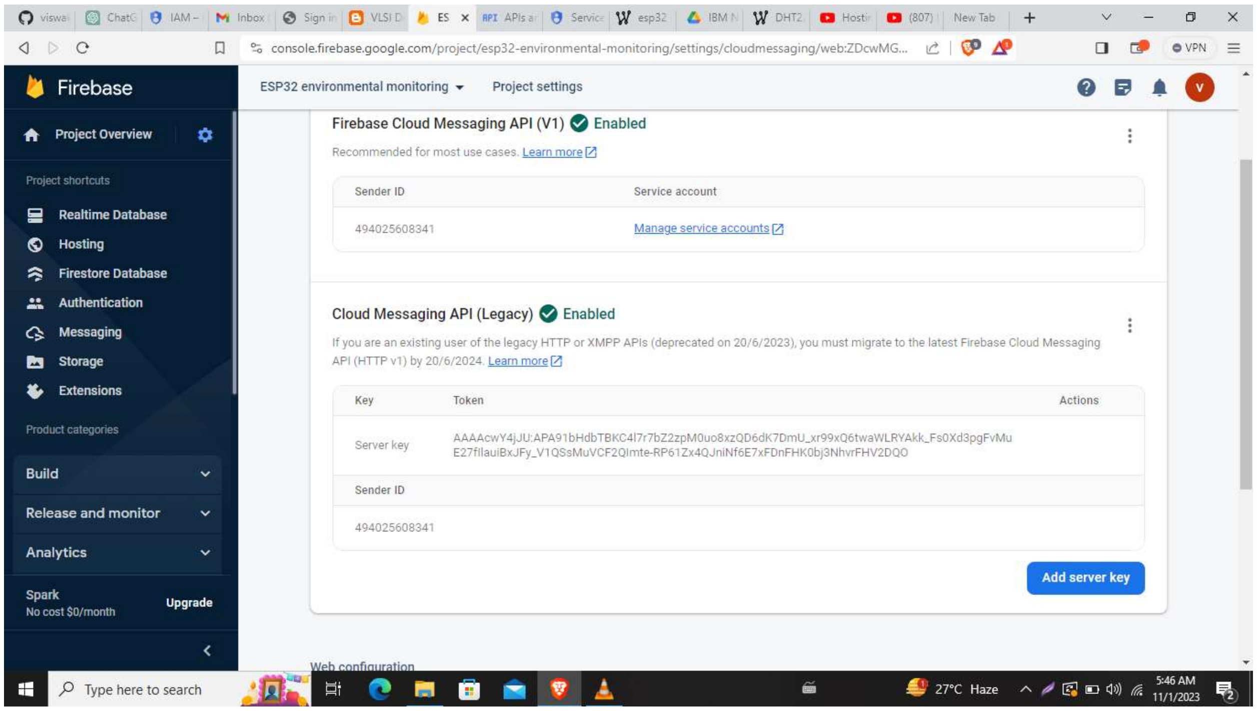
Cloud Messaging

Firebase Cloud Messaging (FCM) (https://firebase.google.com/docs/cloud-messaging/) is the new version of GCM. FCM currently offers 3 server APIs to send messages, Legacy HTTP, Legacy XMPP, and HTTP v1 (https://firebase.google.com/docs/reference). The API described on this page is the Firebase Cloud Messaging Legacy API, which comprises of Legacy HTTP and Legacy XMPP. If you are integrating messaging in a new app, start with the latest HTTP v1 API instead (https://firebase.google.com/docs/reference/fcm/rest/v1/projects.messages). If you are an existing user of the FCM Legacy API (HTTP or XMPP), consider migrating to the HTTP v1 API (https://firebase.google.com/docs/cloud-messaging/migrate-v1). GCM has been deprecated. If your app is using the GCM SDK, upgrade to the FCM SDK immediately as there are no guarantees that clients using the former will continue to work.

By Google

Service name	Type	Status
googlecloudmessaging.googleapis.com	Public API	Enabled

Firebase cloud messaging API shows the server and sender id, it generate credentials certificate in JSON format.



Firebase Cloud Messaging API (V1) Enabled

Recommended for most use cases. Learn more

Sender ID	Service account
494025608341	Manage service accounts

Cloud Messaging API (Legacy) Enabled

If you are an existing user of the legacy HTTP or XMPP APIs (deprecated on 20/6/2023), you must migrate to the latest Firebase Cloud Messaging API (HTTP v1) by 20/6/2024. Learn more

Key	Token	Actions
Server key	AAAAcwY4jJU:APA91bHdbTBKC4l7r7bZ2zpM0uo8xzQD6dK7DmU_xr99xQ6twaWLRYAkk_Fs0Xd3pgFvMu E27fIlauiBxJFy_V1QSsMuVCF2QImte-RP61Zx4QJniNf6E7xFDnFHK0bj3NhvrFHV2DQO	

Sender ID
494025608341

Web development:

With the help of the firebase web development we can host our website in internet. The data sent from iot devices are embedded in website with the help java script tag in html and for user definition style tag are used in html (css) as presentation layer . integrating the API key in java script tag are used to display data over the internet.

Code:

```
<html lang="en">

<head>

  <meta charset="UTF-8">

  <meta name="viewport" content="width=device-width, initial-scale=1.0">

  <title>Real-Time IoT Data</title>
```

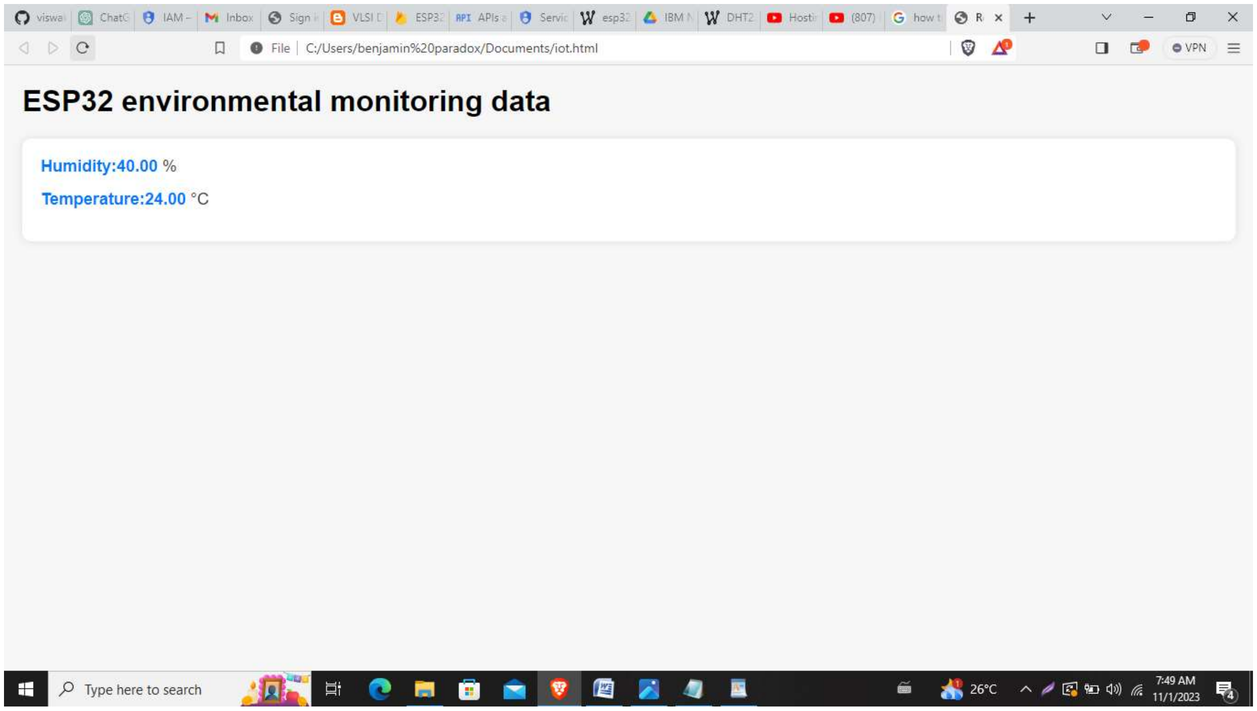
```
    <h1> ESP32 environmental monitoring data </h1>
   <style>
      body {
         font-family: Arial, sans-serif;
         margin: 20px;
      }
      .data-container {
         margin-top: 20px;
      }
      .data-item {
         margin-bottom: 10px;
      }
body {
   font-family: Arial, sans-serif;
   margin: 20px;
   background-color: #f7f7f7;
}
.data-container {
  margin-top: 20px;
   padding: 20px;
   border-radius: 10px;
   box-shadow: 0px 0px 10px 0px rgba(0, 0, 0, 0.1);
   background-color: #ffffff;
}
```

```
.data-item {
   margin-bottom: 15px;
   font-size: 18px;
   color: #333333;
}

.data-item strong {
   color: #007bff; /* Change this color to your preferred color */
}

#humidity, #temperature {
   font-weight: bold;
   color: #28a745; /* Change this color to your preferred color */
}

/* Responsive styles for smaller screens */
@media screen and (max-width: 768px) {
  .data-container {
      padding: 15px;
   }

   .data-item {
      font-size: 16px;
```

```
    }
}
    </style>
</head>
<body>
    <div class="data-container">
        <div class="data-item">
            <strong>Humidity:</strong> <span id="humidity"></span> %
        </div>
        <div class="data-item">
            <strong>Temperature:</strong> <span id="temperature"></span> Â°C
        </div>
    </div>

    <script src="https://www.gstatic.com/firebasejs/9.5.0/firebase-app-compat.js"></script>
    <script src="https://www.gstatic.com/firebasejs/9.5.0/firebase-database-
compat.js"></script>
    <script>
        // Firebase configuration
        const firebaseConfig = {
  apiKey: "AIzaSyCy2aZiFAz0PYkOEt2Wgwr1raIgn7QV97k",
  authDomain: "esp32-environmental-monitoring.firebaseapp.com",
  databaseURL: "https://esp32-environmental-monitoring-default-rtdb.firebaseio.com",
  projectId: "esp32-environmental-monitoring",
storageBucket: "esp32-environmental-monitoring.appspot.com",
messagingSenderId: "494025608341",
appId: "1:494025608341:web:bec4872274bb5844db5f9c",
measurementId: "G-M08YFMBQNC"
   };

   firebase.initializeApp(firebaseConfig);

   const database = firebase.database();

   // Reference to the 'data' node in your Firebase database
   const dataRef = database.ref('data');

   // Update UI with real-time data from Firebase
   dataRef.on('value', (snapshot) => {
     const data = snapshot.val();
     if (data) {
       document.getElementById('humidity').textContent = data.humidity || 'N/A';
       document.getElementById('temperature').textContent = data.temperature || 'N/A';
     } else {
       document.getElementById('humidity').textContent = 'N/A';
       document.getElementById('temperature').textContent = 'N/A';
     }
   });
```

```
  </script>
</body>
</html>
```

Output:


ESP32 environmental monitoring data

Humidity:40.00 %

Temperature:24.00 °C

Real-Time Environmental Monitoring Benefits for Park Visitors and Outdoor Activities:

Health and Safety:

Real-time data on air quality and weather conditions ensure visitors' health and safety, allowing them to make informed decisions about outdoor activities.

Comfortable Experience:

Monitoring temperature and humidity levels enables park management to create a comfortable environment, encouraging longer stays and more enjoyable experiences for visitors.

Weather Alerts:

IoT devices provide real-time weather updates, allowing visitors to plan their activities effectively and stay prepared for sudden weather changes.

Outdoor Event Optimization:

Parks can host outdoor events and activities based on real-time data, ensuring optimal conditions for sports, concerts, and other gatherings.

Energy Efficiency:

Monitoring energy usage through IoT devices helps parks implement energy-efficient solutions, reducing costs and promoting sustainability.

Waste Management:

Smart waste bins equipped with IoT sensors optimize waste collection schedules, keeping the park clean and environmentally friendly.

Wildlife Conservation: Environmental sensors aid in wildlife tracking and protection, enhancing the natural habitat and preserving biodiversity within the park.

Interactive Experiences:

IoT-powered interactive exhibits and informational kiosks engage visitors with real-time environmental data, enhancing their educational experience.

Community Engagement:

Real-time data availability fosters a sense of community. Park visitors can participate in environmental initiatives, promoting a shared responsibility for the park's well-being.

Promotion of Outdoor Activities:

By ensuring a pleasant environment, real-time monitoring encourages outdoor activities such as jogging, picnics, and sports, promoting a healthier lifestyle among park visitors.

In summary, real-time environmental monitoring using IOT devices not only enhances the overall park experience for visitors but also promotes outdoor activities while fostering environmental awareness and community engagement.