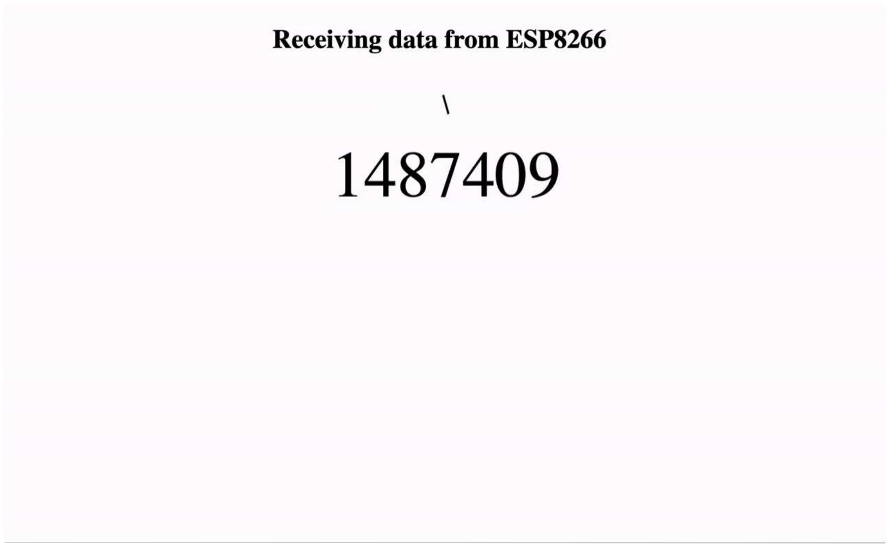
The Easiest Way to Send Data From ESP8266 to a Website

By instanceofMA (/member/instanceofMA/) in Circuits (/circuits/) > Wireless (/circuits/wireless/projects/)

6,960 2 3

Download Favorite


Receiving data from ESP8266

\

1487409

(/member/instanceofMA/)
By **instanceofMA (/member/instanceofMA/)**
Check out Grandeur! (https://grandeur.dev/) Follow

About: Simplifying IoT dev; abridging that huge gap between our softwares & hardwares! | Habits over rules. Exploration. More About instanceofMA » (/member/instanceofMA/)

More by the author:

There are hundreds of real-world applications that involve sending data from a microcontroller to a website, webpage, or web dashboard. And almost all of such applications prefer to be wireless. In such cases, the ESPxxxx family of WiFi modules is the most commonly used one due to their tiny form factor as compared to Arduinos coupled with WiFi shield.

But communication of any kind of data wirelessly between two things requires the knowledge of the languages of the internet, like HTTP GET and POST and XML or JSON. This is usually out of scope of embedded engineers like us, and their mere mention is sometimes enough to send shivers down our spines. So I'm going to try to kick this fear's ass today, **with a much simpler and more powerful way to do the same thing, and more.** Using a tool I call Grandeur (https://grandeur.dev/).

It lets you send data to and fro among your devices and webpages without a learning curve or a requirement to know the languages of the web, e.g., HTTP, JSON, XML, WebSockets, MQTT, etc. Let me show you how. Follow me to the end!

Add Tip Ask Question Comment Download

Supplies

1. ESP8266 board, e.g., NodeMCU
2. Arduino IDE installed in Laptop

Add Tip Ask Question Comment Download

Step 1: Program ESP8266 to Send Data

Here's the code I used for my ESP8266:

```
#include <Grandeur.h>
#include "WiFi.h"


// WiFi credentials
const char* ssid = YourWiFiSsid;
const char* passphrase = YourWiFiPassphrase;


// Grandeur credentials
const char * apiKey = YourApiKey;
const char* token = YourDeviceToken;
const char* deviceId = YourDeviceId;


Grandeur::Project project;


void setup() {
  Serial.begin(9600);
  // This connects to the device to WiFi.
  connectToWiFi(ssid, passphrase);
  // You can initialize device configurations like this.
  project = grandeur.init(apiKey, token);
}
```

<Grandeur.h> gives us access to the functions to send and receive data from the internet.

In **setup()**, I connect to WiFi using the **connectToWiFi()** function and then connect to internet using the **grandeur.init()** function.

In **loop()**, I send the millis() timer value to the internet using **project.device(deviceId).data().set("millis", millis())**. The if-statement

```
if(millis() - current > X) {
      /** CODE
      */
      current = millis();
}
```

makes sure the **CODE** inside the if-statement runs only after every X milliseconds. This is a way of introducing a nonblocking **delay(X)** into the code. This keep my ESP8266 from crashing by sending data too fast.

Here's the **WiFi.h** file that I included:

```
#ifndef WIFI_H
#define WIFI_H


#include <ESP8266WiFi.h>


void connectToWiFi(const char* ssid, const char* passphrase) {
  Serial.begin(9600);
  // Disconnecting WiFi if it"s already connected.
  WiFi.disconnect();
  // Setting it to Station mode which basically scans for nearby WiFi routers.
  WiFi.mode(WIFI_STA);
  // Begin connecting to WiFi
  WiFi.begin(ssid, passphrase);
  Serial.printf("\nDevice is connecting to WiFi using SSID %s and Passphrase %s.\n", ssid,
}


#endif /* WIFI_H */
```

It gives us the **connectToWiFi()** function that helps us connect with WiFi.

Add Tip   Ask Question   Comment   Download

Step 2: Build a Simple Webpage

Here's a simple webpage that I designed. It displays any data that's coming from my ESP8266.

```
<html>
    <head>
        <title>Receiving data from ESP8266</title>
        <!-- Link to Grandeur -->
        <script src="https://unpkg.com/grandeur-js"></script>
        <link rel="stylesheet" href="./main.css">
    </head>
    <body>
        <h1>Receiving data from ESP8266 &nbsp</h1>
        <h1 class="loader">/</h1>
        <p id="data">–</p>


        <script src="./index.js"></script>
    </body>
</html>
```

The line 5 (script tag) does the same thing in this webpage as **<Grandeur.h>** does in the ESP8266 program – gives us access to the functions to send and receive data from the internet.

Here's the stylesheet I used, it makes the loader rotate and brings everything to the center:

```
body {
    display: flex;
    flex-direction: column;
    align-items: center;
}


/* Loader rotation */
.loader {
    position: relative;
    -webkit-animation: spin 1=0.5s linear infinite;
    -moz-animation: spin 0.5s linear infinite;
    animation: spin 0.5s linear infinite;
}


@-moz-keyframes spin {
    100% { -moz-transform: rotate(360deg); }
}
@-webkit-keyframes spin {
    100% { -webkit-transform: rotate(360deg); }
}
@keyframes spin {
    100% {
        -webkit-transform: rotate(360deg);
```

Add Tip   Ask Question   Comment   Download

Step 3: Program the Webpage to Receive Data

Here is the Javascript file of my page:

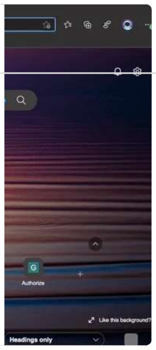
```
const apiKey = YourApiKey;
const accessKey = YourAccessKey;
const accessToken = YourAccessToken;


const project = grandeur.init(apiKey, accessKey, accessToken);
project.auth().login(YourUserEmail, YourUserPassword);

// This subscribes to the "millis" variable.
project.devices().device(YourDeviceId).data().on("millis", (path, value) => document.getEle
```

Subscribing to a variable from my webpage means that my webpage is asking to receive the variable's value when my ESP8266 sends it to the internet. Here I'm asking to receive whatever my ESP8266 sends inside the "millis" variable, which in my case – according to the Step 1 – is the ESP8266's **millis()** timer value.

Add Tip   Ask Question   Comment   Download

Step 4: Fill in the Gaps

0:00 / 0:24



0:00 0:09

0:00 0:10

6 More Images

Now that the ESP8266 is ready to send data and the webpage is ready to receive it, we need to fill in some gaps.

In ESP8266 program, we need three things:

1. YourApiKey
2. YourDeviceToken
3. YourDeviceId

In our webpage's Javascript, we need six things:

1. YourApiKey
2. YourAccessKey
3. YourAccessToken
4. YourUserEmail
5. YourUserPassword
6. YourDeviceId

To get these, we need to create a project on the Grandeur Dashboard at https://cloud.grandeur.tech/ (https://cloud.grandeur.tech/). Here are the steps you can follow:

1. Go to https://cloud.grandeur.tech/ (https://cloud.grandeur.tech/) and click on **get access** to sign up.
2. Create a **new project.**
3. Add a **new model**.
4. Add a **new device**. Copy and save the device ID and token it returns.
5. Add a **new user**. Copy and save the user email and password you used here.
6. **Pair the device** we created in step 4 with the user we created in step 5.
7. Go to settings, **copy and save the API key** from here.
8. Generate **new access key**. Copy and save the access key and token it returns.
9. **Allow the origin** http://localhost:8080 or whatever URL your webpage is served on. We'll see where to find this URL in the next step.

I have also attached all the steps in order in GIFs.

By doing this, **you get access to use Grandeur as your communication broker.**

Now,

1. In ESP8266 program, paste your WiFi SSID and WiFi passphrase, and the API key, device ID, and device token, you copied in step 7 and 4, respectively.
2. In the webpage's Javascript file, paste the API key, access key, access token, user email, user password, and device ID, you copied in step 7, 8, 5, and 4, respectively.

Now both the ESP8266 and the webpage are ready to go live.

Step 5: Go Live

Compile and upload the ESP8266 program in your ESP8266 using Arduino IDE or CLI.

Start a local server to serve your webpage. If you have python 3, you can run this command in a terminal in your webpage's folder:

```
python3 -m http.server
```

If you have node.js, you can run this:

```
npx http-server
```

Open your browser and go to the URL that is displayed in your terminal after running these commands. **This is the URL that you should add while allowing origin in the last step.**

If everything works perfectly, your ESP8266 will connect to WiFi, and then to the internet, and start sending **millis()** value twice in a second. In your browser, you should see the wheel spinning and below it, the value of ESP8266's millis updating twice every second.

Step 6: Conclusion

In essence, it's just two methods that are making this work:

1. In ESP8266, **project.device(deviceId).data().set("millis", millis())** sends data to the internet.
2. In the webpage's Javascript file, **project.devices().device(deviceId).data().on("millis", FunctionToPrintData)** receives data from the internet.

You can also send data in reverse, from the webpage to the ESP8266, by switching the methods:

1. In the webpage's Javascript file, **project.devices().device(deviceId).data().set("hello", "Hello from webpage.").**
2. In ESP8266, **project.device(deviceId).data().on("hello", FunctionToPrintData).**

You can also receive and send data to more than one device from the webpage. Just add a new device and pair it with your user in your project from the Grandeur Dashboard at https://cloud.grandeur.tech/devices/ (https://cloud.grandeur.tech/devices/) and copy and paste the line **project.devices().device(deviceId).data().on("millis", FunctionToPrintData)** another time in the webpage's Javascript but replace its device Id with the new device, and now your webpage will receive data from both the ESP8266. This is how the webpage's Javascript looks for receiving data from multiple devices:

```
const apiKey = YourApiKey;
const accessKey = YourAccessKey;
const accessToken = YourAccessToken;


const project = grandeur.init(apiKey, accessKey, accessToken);
project.auth().login(YourUserEmail, YourUserPassword);

// Receiving data from device 1.
project.devices().device(YourDeviceIdOne).data().on("millis", (path, value) => document.get
// Receiving data from device 2.
project.devices().device(YourDeviceIdTwo).data().on("millis", (path, value) => document.get
```

Here's the GitHub repo (https://github.com/abdullahmahboob/Send-data-from-ESP8266-to-a-Website.git) containing the source code for this project. **esp8266** folder contains the Arduino sketch that you can compile and upload to ESP8266 using Arduino IDE or CLI. **website** folder contains the code for the webpage which you can serve using a local server.

I hope this simplifies the data exchange among your devices and webpages. If you have any questions, ask them in comments, or you can always reach me out at ma@grandeur.tech. Thanks for bearing with me!

Add Tip Ask Question Comment Download

Be the First to Share

Did you make this project? Share it with us!

I Made It!

Recommendations

(/Print-Paint-and-Program-a-Guardian-to-Track-Humans/)

Print, Paint, and Program a Guardian to Track Humans and Dogs Using a Pi, Camera, and Servo (/Print-Paint-and-

6 414

(/Battery-Board-for-PALPi-Game-Console/)

Battery Board for PALPi Game Console (/Battery-Board-for-PALPi-Game-Console/) by Arnov Sharma

21 3.5K

(/contest/yard23/) (/contest/reuse23/)

Add Tip

Ask Question

Post Comment

We have a **be nice** policy.
Please be positive and constructive.

Add Images Post

3 Comments

(/member/pongzhim/) pongzhim (/member/pongzhim/) 3 months ago

Reply Upvote

I can't generate access keys

(https://content.instructables.com/FI8/24WF/LDHEAGWD/FI824WFLDHEAGWD.jpg?auto=webp&fit=bounds&frame=1&height=1024&width=1024)

1 (/member/StumpChunkman/) StumpChunkman (/member/StumpChunkman/) 1 year ago

Reply Upvote

Thanks for sharing!

1 reply

Post Comment