Chapter 4: Cloud Data Monitoring

204335 : Microcontroller and IoT

Part: ESP8266

Suphakit Awiphan

Chiang Mai University

Chapter 4: Cloud Data Monitoring

204335 Microcontroller and IoT

Chapter 4 – Cloud Data Monitoring

Parts You'll Need for This Chapter

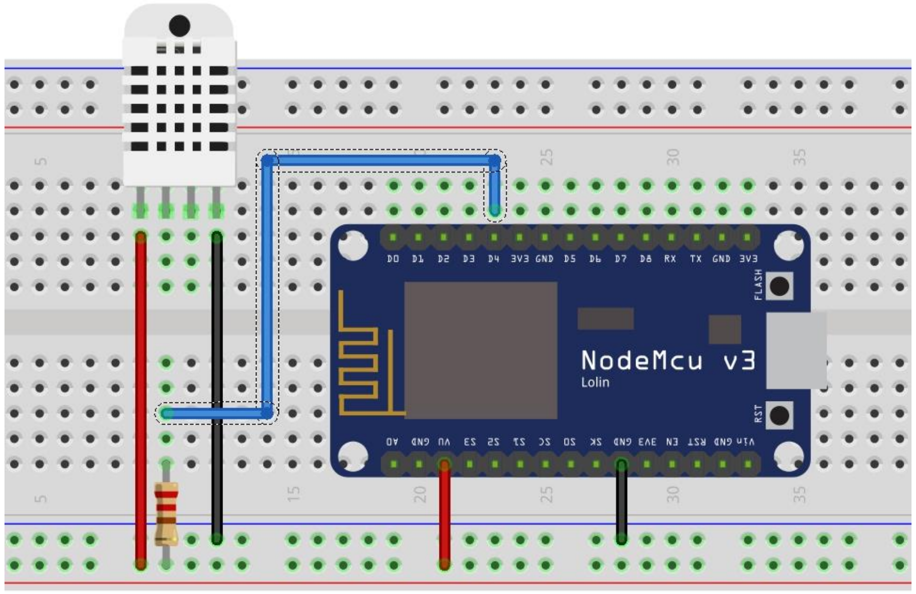
- ESP8266 board
- USB cable
- DHT22
- Soil moisture sensor
- 220 Ω resistor
- 100 Ω resistor
- 10 kΩ resistor
- Breadboard
- Jumper wires

Outline

- Connecting Sensors to ESP8266
- Posting Sensor Data Online
- Retrieving Online Data
- Securing Online Data
- Monitoring Sensor Data from Cloud Dashboard
- Creating Automated Alerts

4.1 Connecting Sensors to ESP8266

Getting Ready



NodeMcu v3
Lolin
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
FLASH
RST
A0 GND VU S3 S2 S1 SC SO SK GND 3V3 EN RST GND Vin
5
15
20
25
30
35

How to Do It

```
#include "DHT.h"

#define DHTPIN 2     // what digital pin we're connected to
#define DHTTYPE DHT11   // DHT 11

DHT dht(DHTPIN, DHTTYPE);

int sensorReading = 0; // holds value soil moisture sensor reading

void setup() {
  Serial.begin(9600);
  dht.begin();
}


void loop() {
  // Wait a few seconds between measurements.
  delay(2000);
  // get humidity reading
  float h = dht.readHumidity();
  // get temperature reading in Celsius
  float t = dht.readTemperature();
  // Check if any reads failed and exit early (to try again).
  if (isnan(h) || isnan(t)) {
    Serial.println("Failed to read from DHT sensor!");
    return;
  }
  //get soil moisture reading
  sensorReading = analogRead(A0);
```

```
  // display data on serial monitor
  Serial.print("Humidity: ");
  Serial.print(h);
  Serial.print(" %\t");
  Serial.print("Temperature: ");
  Serial.print(t);
  Serial.println(" *C ");
  Serial.print("Soil moisture: ");
  Serial.println(sensorReading);
}
```

4.2 Posting Sensor Data Online

Posting Sensor Data Online

- Using the ESP8266, you can log sensor data to online servers for monitoring and controlling
- We will use an ESP8266 board to post and store sensor data on dweet.io
- Posting data to dweet.io is simple
- It is done by calling a URL such as

  https://dweet.io/dweet/for/my-thing-name?hello=world

- You are required to change
  - *my-thingname* to the name of your thing,
  - *hello* to the name of the parameter that you are posting,
  - *world* to the value of the parameter

Posting Sensor Data Online

1) Connect the ESP8266 board to a Wi-Fi network that has Internet connection and then read the humidity and temperature from the sensors

2) Replace the <humidity value> and <temperature value> in the URL with the readings obtained from the sensors

The URL will be used to send a http request to the dweet.io server

3) When the ESP8266 successfully sends the http request, the sensor data will be published on the dweet.io platform. To view the data, visit this URL from any web browser: https://dweet.io/follow/garden-monitor-11447:

```
// Libraries
#include <ESP8266WiFi.h>
#include "DHT.h"
```

Posting Sensor Data Online

4) ssid and password is set in this section:

```
// Wi-Fi network SSID and password
const char* ssid = "your-ssid";
const char* password = "your-password";
```

5) Store the host name of the cloud server:

```
// Host
const char* host = "dweet.io";
```

6) Define the pin that is connected to the DHT22 signal pin and the type of DHT sensor that we are using:

```
#define DHTPIN 2 // what digital pin DHT22 is connected to
#define DHTTYPE DHT22 // DHT 11 sensor
```

7) Create a dht object:

```
DHT dht(DHTPIN, DHTTYPE);
```

Posting Sensor Data Online

8) Initialize the serial interface and the DHT object.

9) Configure the ssid and password of the Wi-Fi network and connect the ESP8266 to it:

```
void setup() {
  Serial.begin(115200); // initialize serial interface
  dht.begin(); // initialize DHT22 sensor
  delay(10);
  // We start by connecting to a WiFi network
  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
```

Posting Sensor Data Online

```
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Posting Sensor Data Online

10) Delay for five seconds, then print the name of the host we are connecting to on the serial monitor:

```
void loop() {
  delay(5000);
  Serial.print("connecting to ");
  Serial.println(host);
```

11) Connect to the host server. Retry if not successful:

```
  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
    Serial.println("connection failed");
    return;
  }
```

Posting Sensor Data Online

12) Get readings from the DHT22 sensor and soil moisture sensor:

```
// Read sensor inputs
// get humidity reading
float h = dht.readHumidity();
// get temperature reading in Celsius
float t = dht.readTemperature();
// Check if any reads failed and exit early (to try again).
while (isnan(h) || isnan(t)) {
   Serial.println("Failed to read from DHT sensor!");
   delay(2000); // delay before next measurements
   //get the measurements once more
   h = dht.readHumidity();
   t = dht.readTemperature();
}
```

Posting Sensor Data Online

Generate a URL for the GET request we will send to the host server. The URL will include the sensor readings:

```
// We now create a URI for the request
String url = "/dweet/for/garden-monitor-11447?humidity=";
url += String(h);
url += "&temperature=";
url += String(t);
```

Posting Sensor Data Online

13. Send the GET request to the server and check whether the request has been received, or if it has been timed out:

```
    // Send request
    Serial.print("Requesting URL: ");
    Serial.println(url);
    client.print(String("GET ") + url + " HTTP/1.1\r\n" +
       "Host: " + host + "\r\n" +
       "Connection: close\r\n\r\n");
    unsigned long timeout = millis();
    while (client.available() == 0) {
       if (millis() - timeout > 5000) {
          Serial.println(">>> Client Timeout !");
          client.stop();
          return;
    }
}
```

Posting Sensor Data Online

Read incoming data from the host server line by line and display the data on the serial monitor. Close the connection after all the data has been received from the server:

```
    // Read all the lines from the answer
    while(client.available()){
       String line = client.readStringUntil('\r’);
       Serial.print(line);
    }
    // Close connecting
    Serial.println();
    Serial.println("closing connection");
 }
```

4.3 Retrieving Online Data

Retrieving Online Data

Reading the most recent data from dweet.io is just as simple as posting it. All you've got to do is send an http request using the following URL:

https://dweet.io/get/latest/dweet/for/my-thing-name

1. Replace my-thing-name with the name of your thing and you are good to go.

Therefore, in our case, the URL we will use is https://dweet.io/get/latest/dweet/for/garden-monitor-11447.

2. In addition to getting the most recently posted data, you can retrieve the last five

data entries in a 24-hour period. You can do that using the following URL:

https://dweet.io/get/dweets/for/garden-monitor-11447.

However, we will not be using this URL in our code.

Retrieving Online Data

3. Now that you have your URL figured out, connect your ESP8266 board to a Wi-Fi network with an Internet connection and send an http request to dweet.io using your URL. dweet.io will send a response that contains the most recently logged sensor data.

4. You can capture and display the data on the serial monitor or use it to perform a certain function on your board. The code we are using will display the data on the serial monitor:

```
// Libraries

#include <ESP8266WiFi.h>
```

Retrieving Online Data

5. ssid and password:

```
    const char* ssid = "your-ssid";
    const char* password = "your-password";
```

6. Store the host name of the cloud server:

```
    const char* host = "dweet.io";
```

7. Configure the ssid and password of the Wi-Fi network and connect the ESP8266 to the Wi-Fi network:

```
  void setup() {
    // Serial
    Serial.begin(115200);
    delay(10);
    // We start by connecting to a WiFi network
    Serial.println();
    Serial.println();
    Serial.print("Connecting to ");
```

Retrieving Online Data

```
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
     delay(500);
     Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}
```

Retrieving Online Data

8. Delay for five seconds then print the name of the host we are connecting to on the serial monitor:

```
void loop() {
  delay(5000);
  Serial.print("connecting to ");
  Serial.println(host);
```

9. Connect to the host server:

```
  // Use WiFiClient class to create TCP connections
  WiFiClient client;
  const int httpPort = 80;
  if (!client.connect(host, httpPort)) {
     Serial.println("connection failed");
     return;
  }
```

Retrieving Online Data

10. Generate the URI for the GET request that we will send to the host server. We will

use it to retrieve data from the dweet.io servers:

```
// We now create a URI for the request
String url = "/get/latest/dweet/for/garden-monitor-11447";
```

11. Send the GET request to the server and check whether the request has been received, or if it has been timed out:

```
        // Send request
        Serial.print("Requesting URL: ");
        Serial.println(url);
        client.print(String("GET ") + url + " HTTP/1.1\r\n" + "Host: " + host + "\r\n" +
        "Connection: close\r\n\r\n");
        unsigned long timeout = millis();
        while (client.available() == 0) {
           if (millis() - timeout > 5000) {
           Serial.println(">>> Client Timeout !");
           client.stop();
          return;
        }
      }
```

Retrieving Online Data

Read incoming data from the host server line by line and display the data on the serial monitor. Close the connection after all the data has been received from the server:

```
    // Read all the lines from the answer
    while(client.available()){
       String line = client.readStringUntil('\r’);
       Serial.print(line);
    }
    // Close connecting
    Serial.println();
    Serial.println("closing connection");
  }
```

4.4 Securing Online Data

Locking Your Things

- When you lock your thing, its name cannot be used by another person.
- Using the lock and key, you can secure your things and prevent other people/things from accessing your data on dweet.io
- You can then lock your things through the web API by calling this URL:

```
https://dweet.io/lock/{thing_name}?lock={your_lock}&key={your_key}
```

- If your thing is locked successfully, you will receive this reply:

```
{"this":"succeeded","by":"locking","the":"garden-monitor-11447","with":"lock"}
```

Locking Your Things

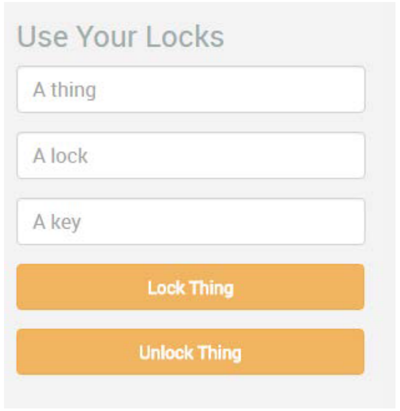
- You can also use the https://dweet.io/locks webpage to lock your things
- Once your thing is locked, you will only be able to access it using the unique key.
- To do that, just add a parameter called key to the API call you are making. For instance:

```
https://dweet.io/dweet/for/{my_locked_thing}?key={my_key}&hello=world&foo=bar

https://dweet.io/get/dweets/for/{my_locked_thing}?key={my_key}

https://dweet.io/get/latest/dweet/for/{my_locked_thing}?key={my_key}

https://dweet.io/listen/for/dweets/from/{my_locked_thing}?key={my_key}
```



Locking Your Things

- You can unlock your thing using this URL:

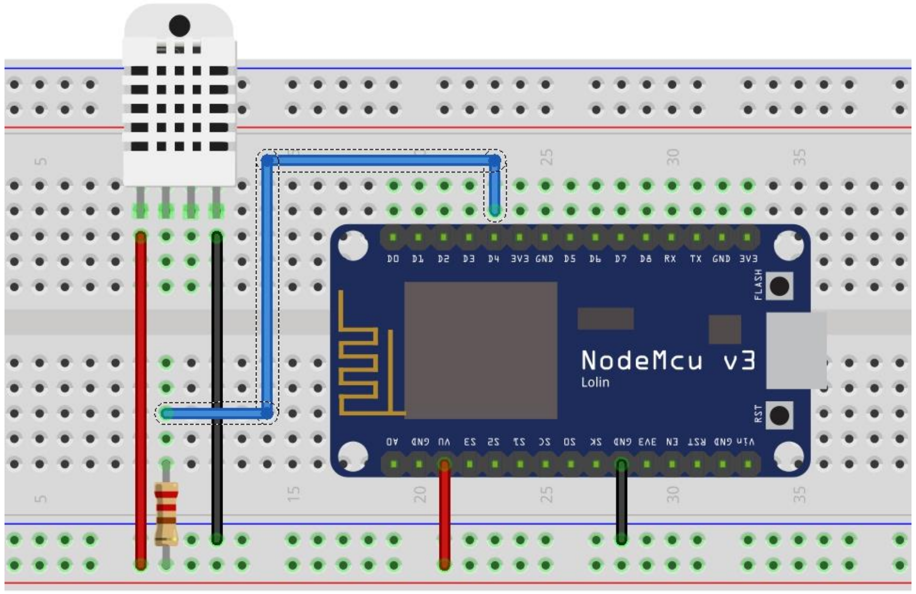
  https://dweet.io/unlock/{thing_name}?key={your_key}

- remove the lock completely regardless of what things it is connected to using this URL:

  https://dweet.io/remove/lock/{your_lock}?key={your_key}

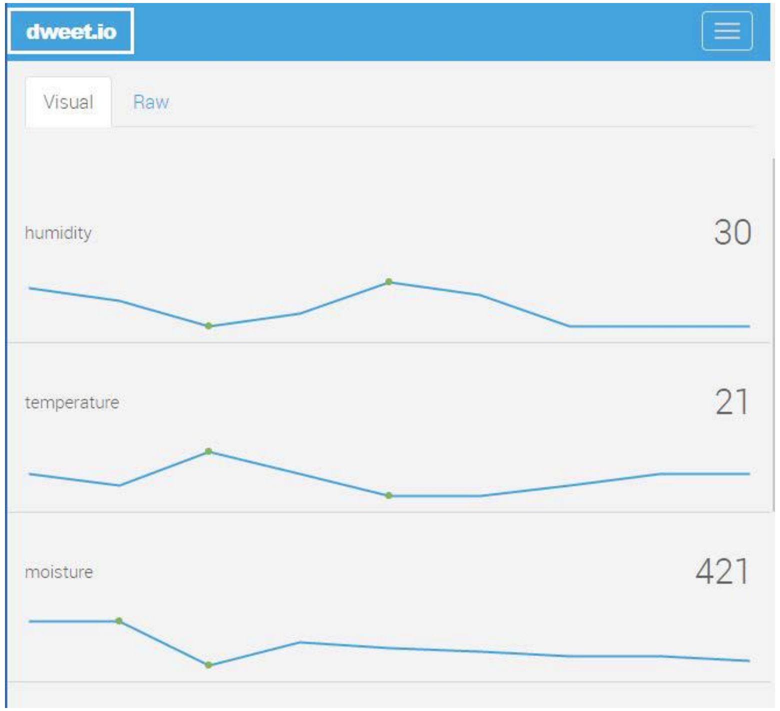
4.5 Monitoring Sensor Data from Cloud Dashboard

Getting Ready


NodeMcu v3
Lolin
D0 D1 D2 D3 D4 3V3 GND D5 D6 D7 D8 RX TX GND 3V3
A0 GND VU S3 S2 S1 SC SO SK GND 3V3 EN RST GND Vin
FLASH
RST
5
15
20
25
30
35

How to Do It

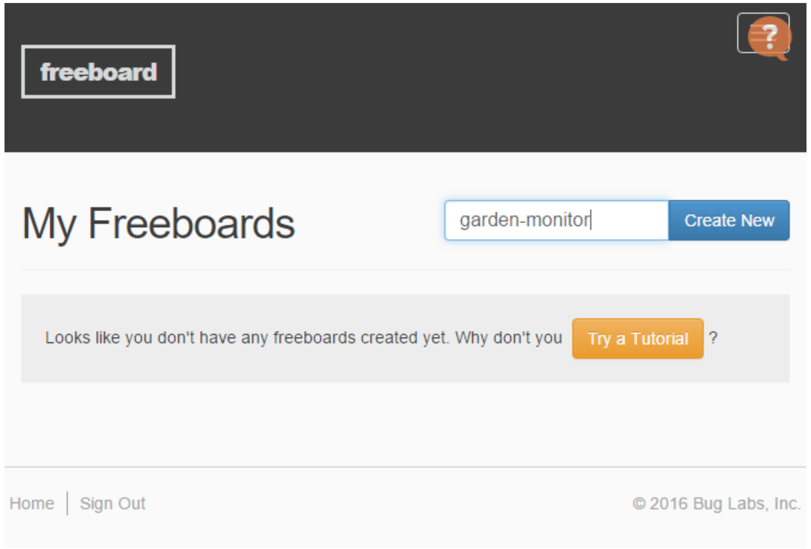
- call http://dweet.io/follow/{thing-name}. Replace {thingname} with the name of your thing



The disadvantage of using the dweet.io graphical representation of posted data is that you do not have any other options for how to present your data other than the line graph

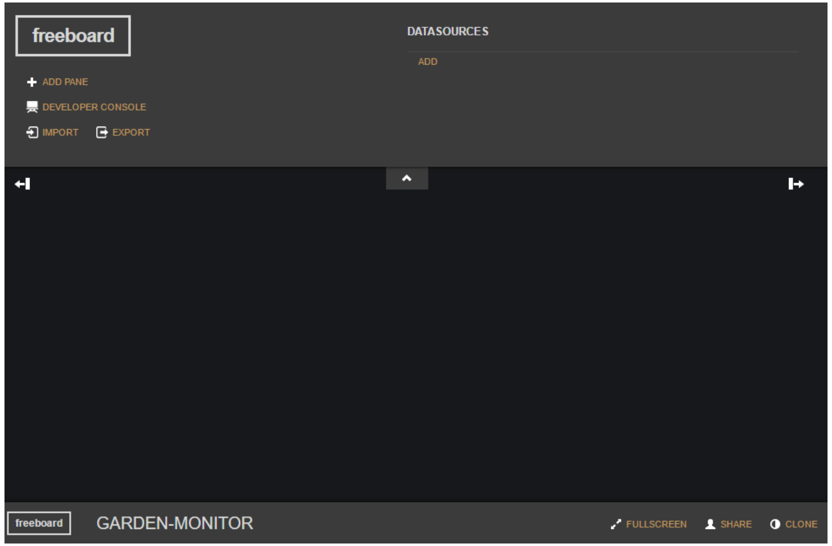
How to Do It – Freeboard.io

- Freeboard.io facilitates cloud dashboards that take the posted data and convert it to meaningful graphical presentations

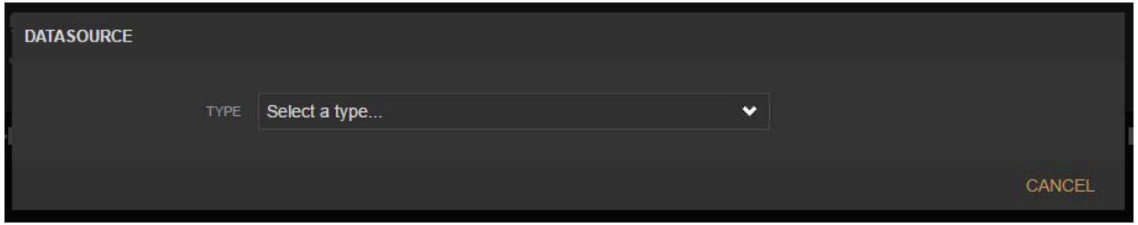


Looks like you don't have any freeboards created yet. Why don't you Try a Tutorial ?

© 2016 Bug Labs, Inc.



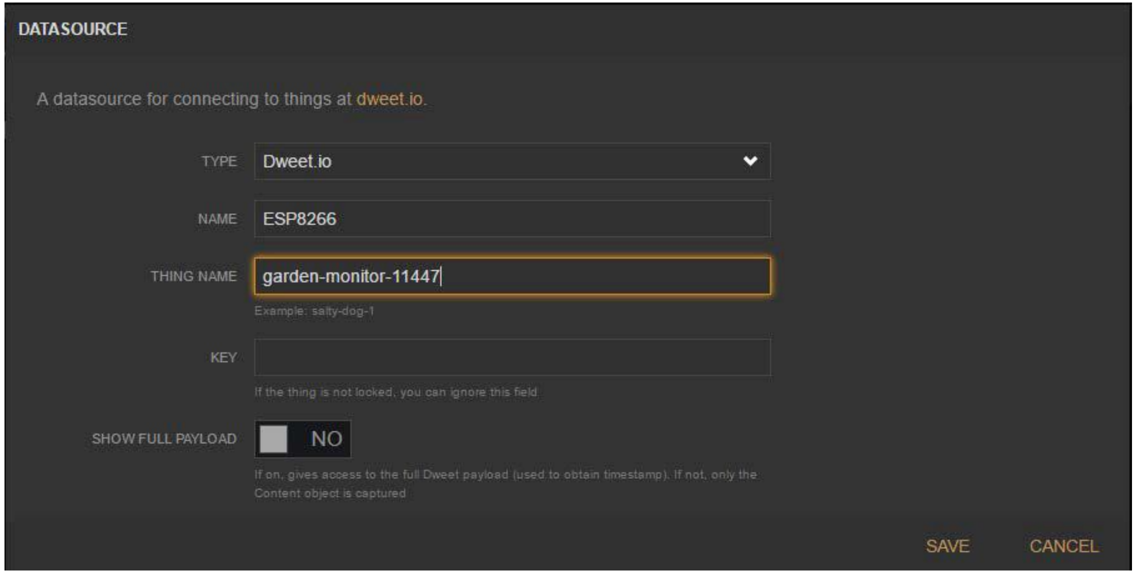
How to Do It – Freeboard.io



DATASOURCE

TYPE Select a type...

CANCEL



DATASOURCE

A datasource for connecting to things at dweet.io.

TYPE Dweet.io

NAME ESP8266

THING NAME garden-monitor-11447

Example: salty-dog-1

KEY

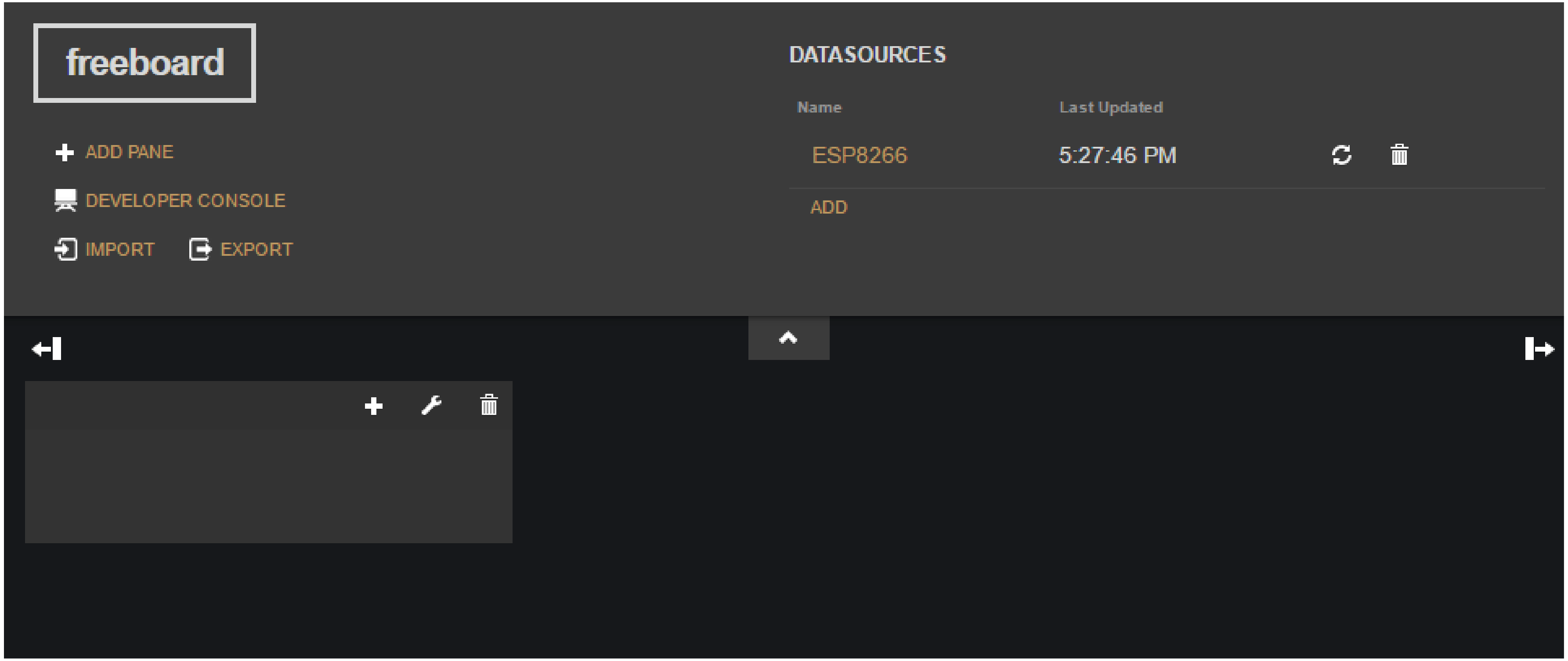
If the thing is not locked, you can ignore this field

SHOW FULL PAYLOAD NO

If on, gives access to the full Dweet payload (used to obtain timestamp). If not, only the Content object is captured

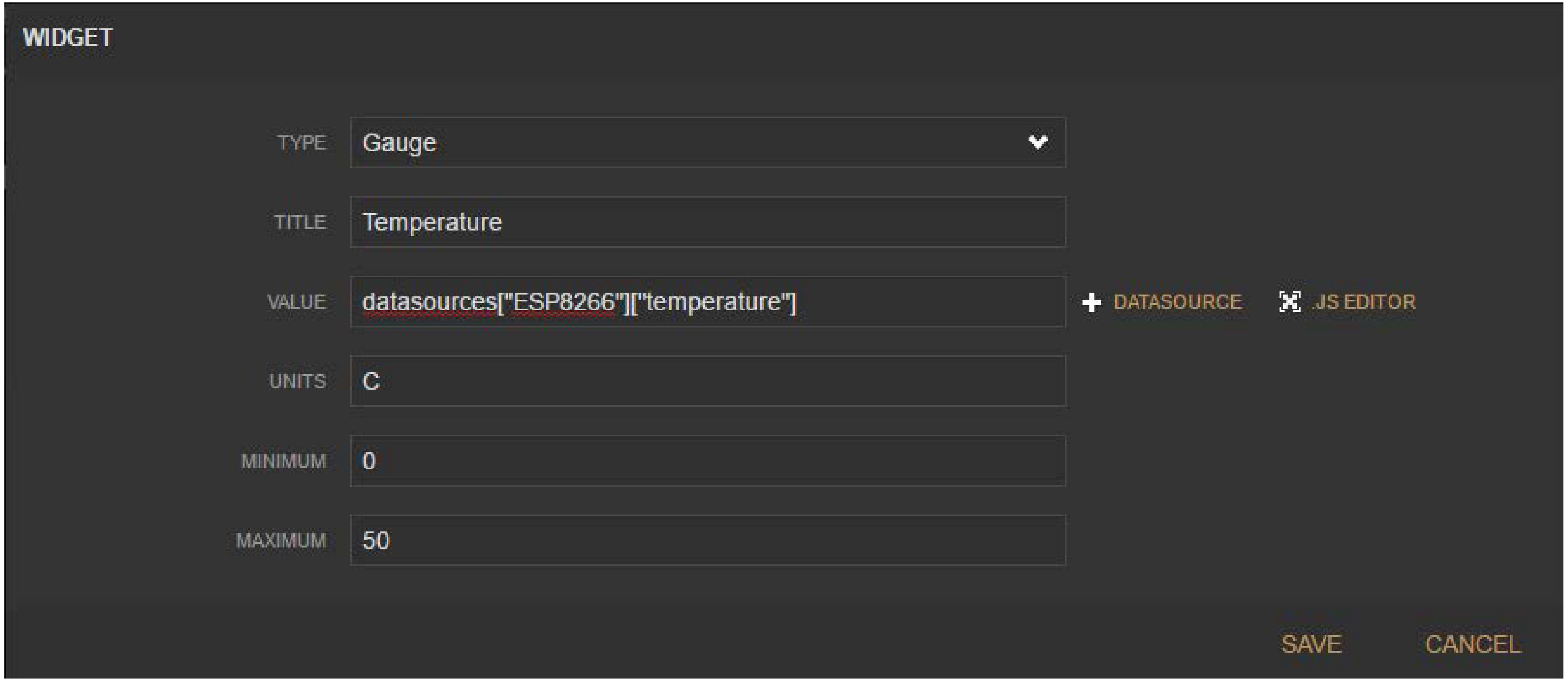
SAVE CANCEL

How to Do It – Freeboard.io



Name	Last Updated
ESP8266	5:27:46 PM

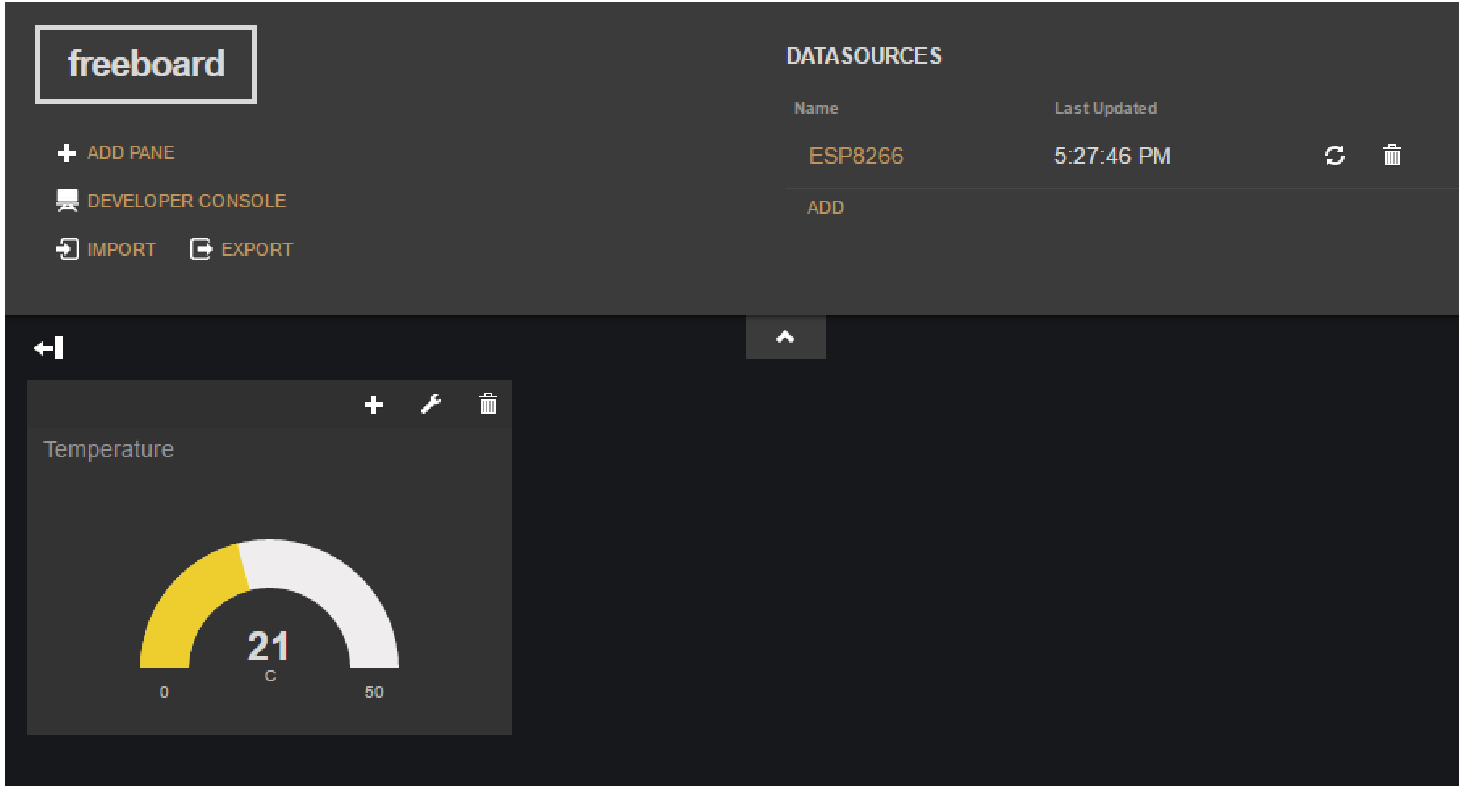
How to Do It – Freeboard.io



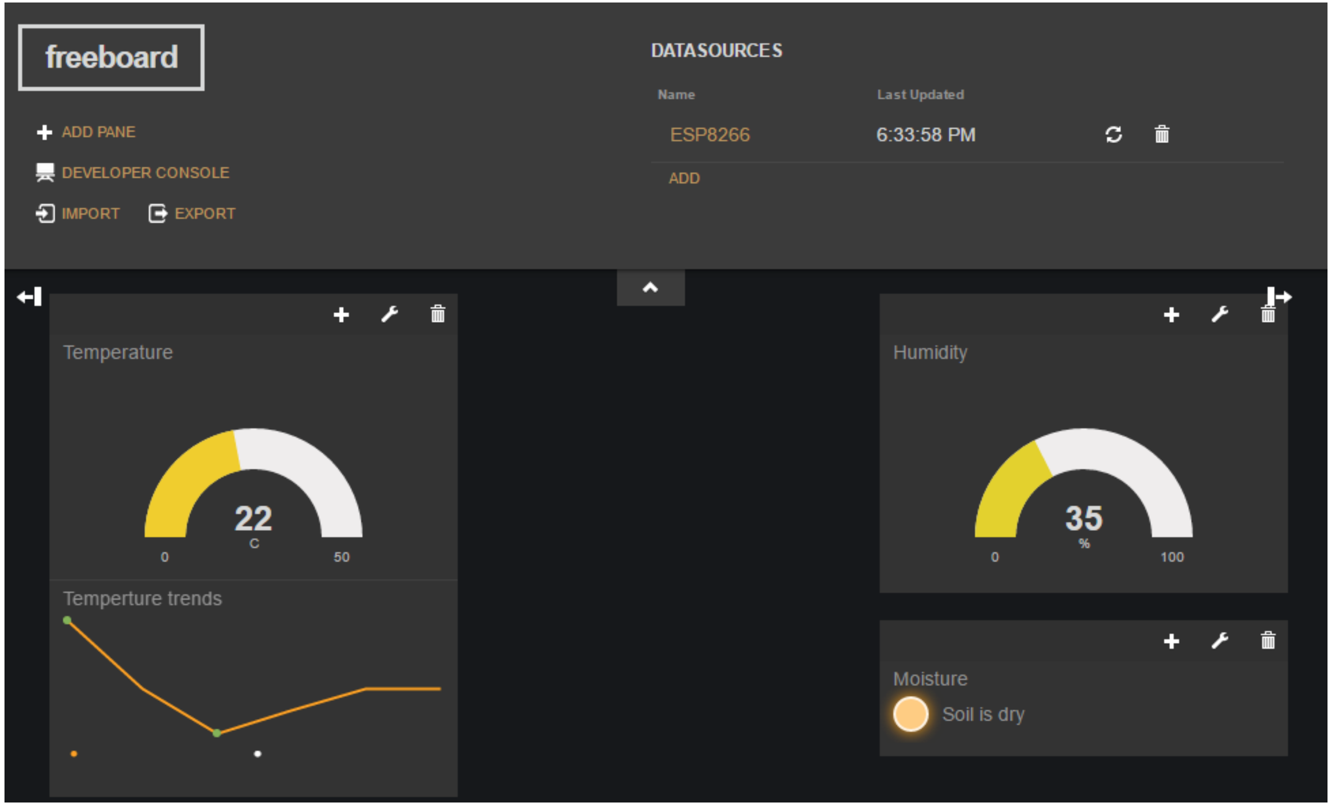
WIDGET

TYPE	Gauge
TITLE	Temperature
VALUE	datasources["ESP8266"]["temperature"]
UNITS	C
MINIMUM	0
MAXIMUM	50

How to Do It – Freeboard.io



How to Do It – Freeboard.io



4.6 Creating Automated Alerts

How to Do It

1) To create an alert, make an http request using a URL with this format:

```
https://dweet.io/alert/{recipients}/when/{thing}/{condition}?key={key}
```

In the preceding URL:

- {recipients}: This refers to e-mail addresses you want dweet.io to send notifications to. If there are several e-mail addresses, separate them using commas.
- {thing}: This is the valid name for your thing
- {condition}: This is a JavaScript expression that you use to evaluate data stored in dweet.io.

For example: if(dweet.temp <= 32) return "frozen"; else if(dweet.temp >= 212) return "boiling";

- {key}: This is the valid key for your locked dweet

How to Do It

2) To create an alert when the temperature exceeds 25 degrees Celsius, we will use this URL:

```
https://dweet.io/alert/my-email@domain.com/when/garden-monitor-
11447/if(dweet.temperature > 25) return "Too hot";?key={key}
```

You can remove alerts using this URL format:

```
https://dweet.io/remove/alert/for/{thing}?key={key}
```

Therefore, use your web browser to call this URL to lock your thing:

```
https://dweet.io/lock/garden-monitor-11447?lock={your_lock}&key={your_key}
```

How to Do It

3) Replace {your_lock} with the lock ID you were sent via e-mail and replace {your_key} with the master key that was provided in the e-mail

- Once the thing has been locked successfully, create an alert to send you notifications when the temperature exceeds 25 degrees Celsius. Do that by calling this URL in your web browser: https://dweet.io/alert/ my-email@domain.com/when/garden-monitor-11447/if(dweet. temperature > 25) return "Too hot";?key={key}

4) Remember to replace {key} with your master key

5) After using your web browser to lock your thing and create an alert, upload the data

logging sketch to your ESP8266. This will log sensor data on dweet.io and inform

you when the temperature goes above 25 degrees Celsius: