

## ESP-12E NodeMcu Lua WiFi Development Board

NodeMcu Lua **ESP-12E** WIFI Development Board

## Wireless 802.11 b/g/n standard

Support STA / AP / STA + AP three operating modes

Built-in **TCP / IP protocol** stack to support multiple TCP Client connections (5 MAX)

D0 ~ D8, SD1 ~ SD3: used as GPIO, PWM, IIC, etc., port driver capability 15mA

AD0: 1 channel ADC

Input: **4.5V ~ 9V** (10VMAX), USB-powered

Current: continuous transmission:  $\approx 70\text{mA}$  (200mA MAX), Standby:  $<200\mu\text{A}$

Transfer rate:110-460800bps

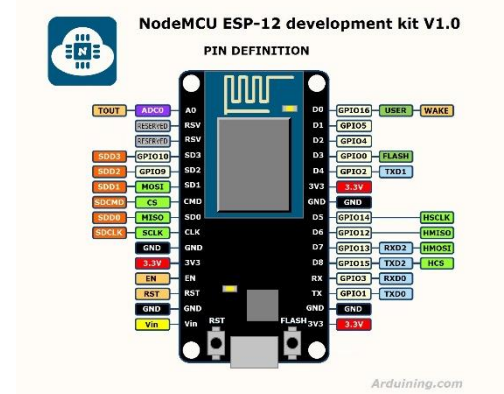
Support UART / **GPIO** data communication interface

### Remote firmware upgrade (OTA)

## Support Smart Link Smart Networking

Working temperature: -40 °C ~ + 125 °C

Drive Type: Dual high-power H-bridge driver



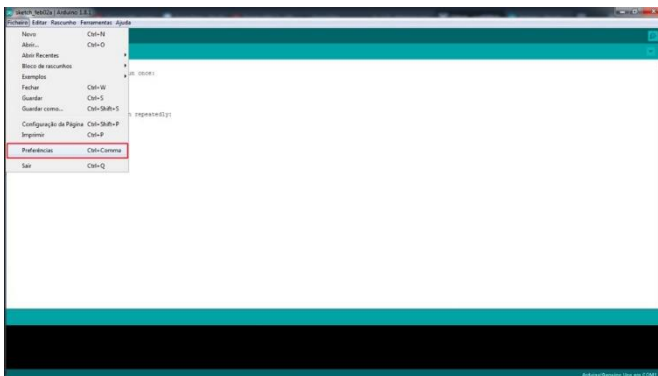
## 1. INSTALAÇÃO Placa ESP-12E NodeMCU no Arduino IDE

### a. Configuração do lick para a placa ESP

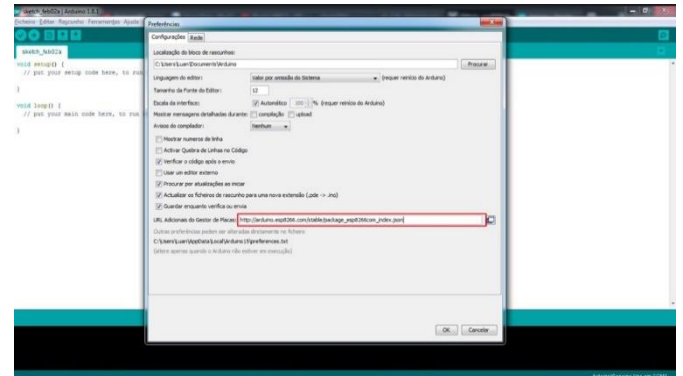
Link a ser colocado no IDE do Arduino (<https://github.com/esp8266/Arduino>):

[https://arduino.esp8266.com/stable/package\\_esp8266com\\_index.json](https://arduino.esp8266.com/stable/package_esp8266com_index.json)

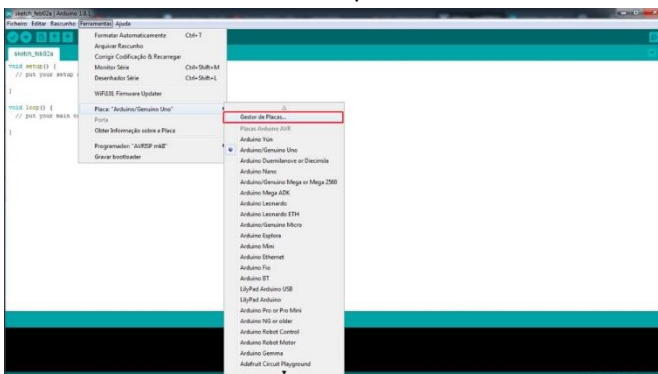
## 1: Preferências



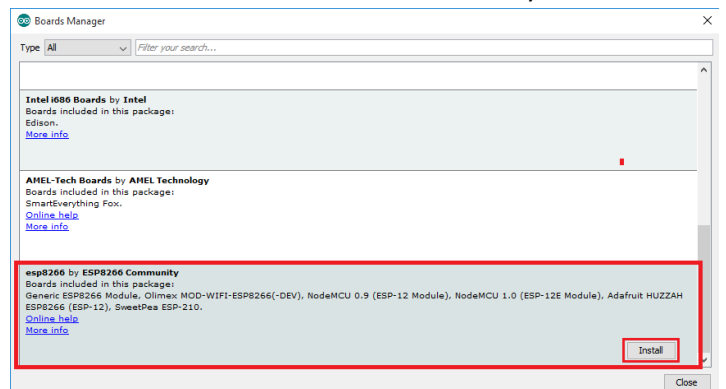
## 2: Copy e Cole LINK



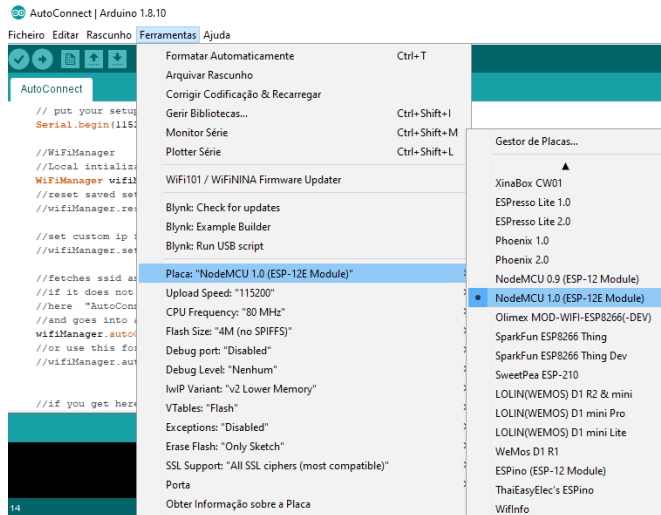
### 3: Gestão de placas



#### 4: instalar ESP8266 community

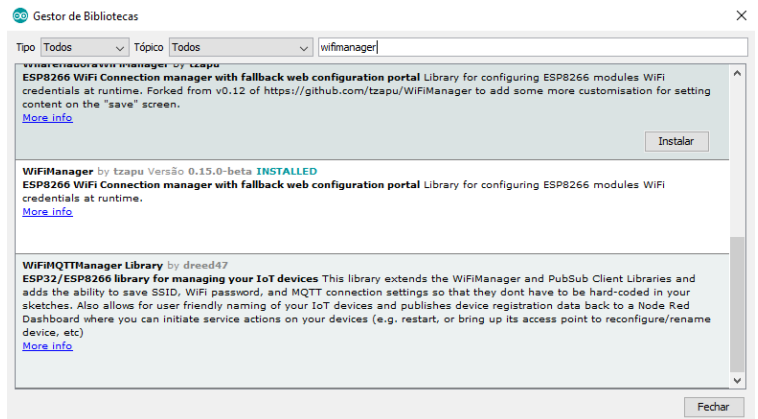
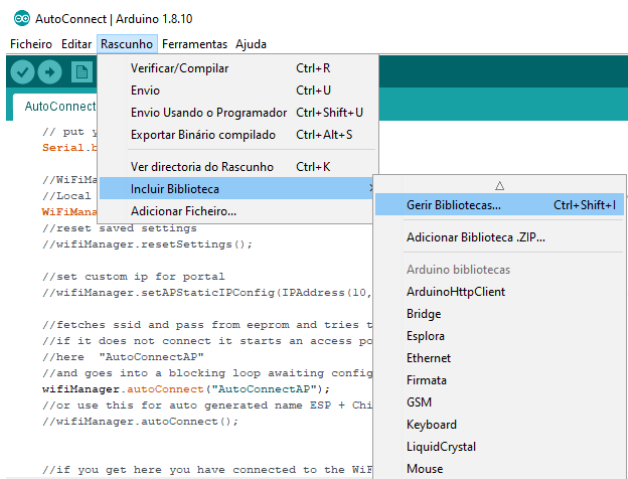


## 5: Escolher “NodeMCU 1.0 ESP-12E”

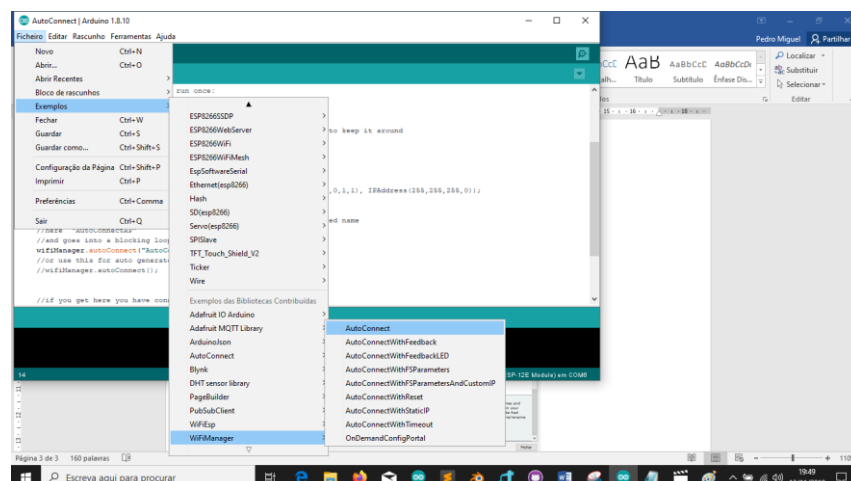


### b. Biblioteca WiFi ESP-12E

#### Adicionar biblioteca “WiFiManager”



## EXEMPLOS:



## 2. Testar o 1º Exemplo – AutoConnect

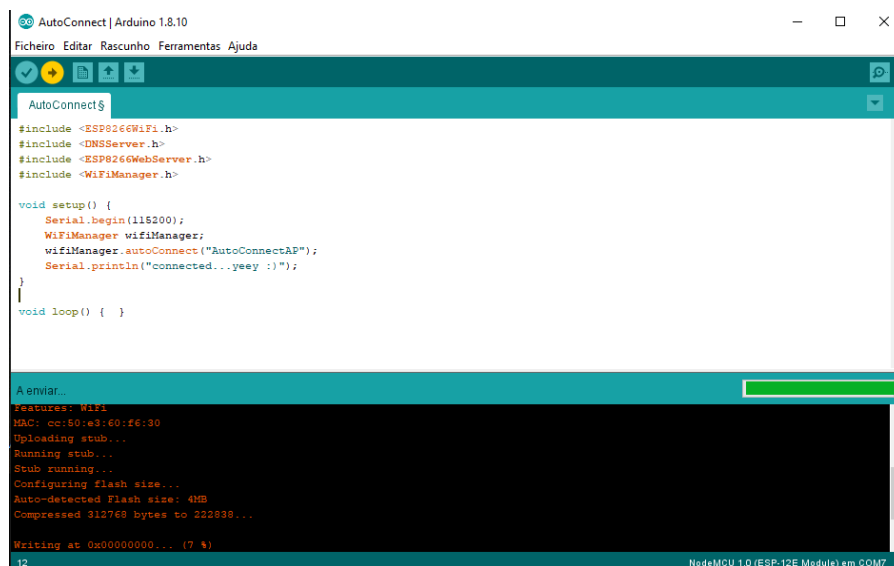
### a) Limpar o código do exemplo

```
#include <ESP8266WiFi.h>
#include <DNSServer.h>
#include <ESP8266WebServer.h>
#include <WiFiManager.h>

void setup() {
  Serial.begin(115200);
  WiFiManager wifiManager;
  wifiManager.autoConnect("AutoConnectAP");
  Serial.println("connected...yeey :)");
}

void loop() { }
```

### b) Enviar para NodeMCU ESP-12E

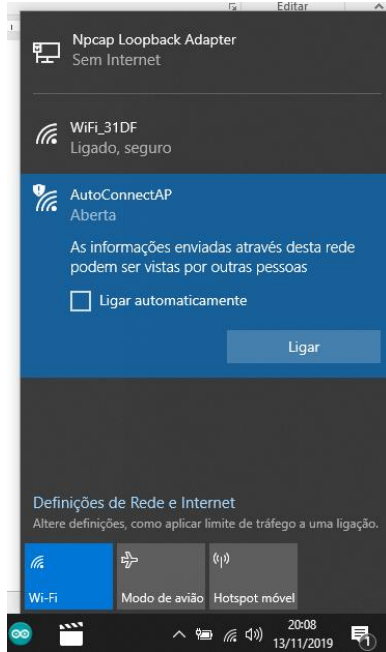


### c) Verificar se gravou e o ip atribuído pelo ESP-12E

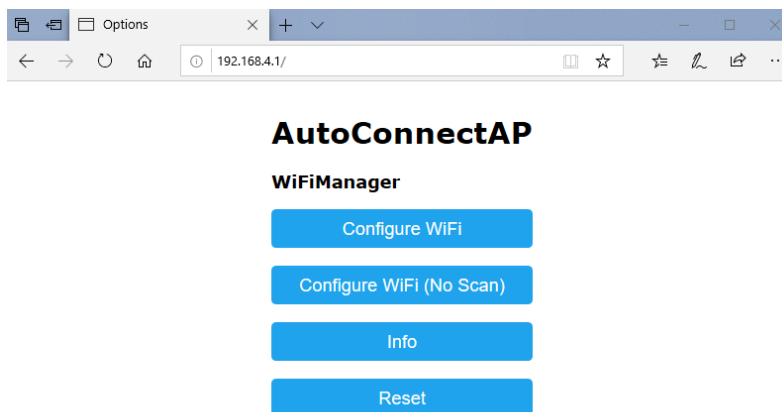
COM7

```
*WM: AutoConnect
*WM: Connecting as wifi client...
*WM: No saved credentials
*WM: Connection result:
*WM: 0
*WM:
*WM: Configuring access point...
*WM: AutoConnectAP
*WM: AP IP address:
*WM: 192.168.4.1
*WM: HTTP server started
```

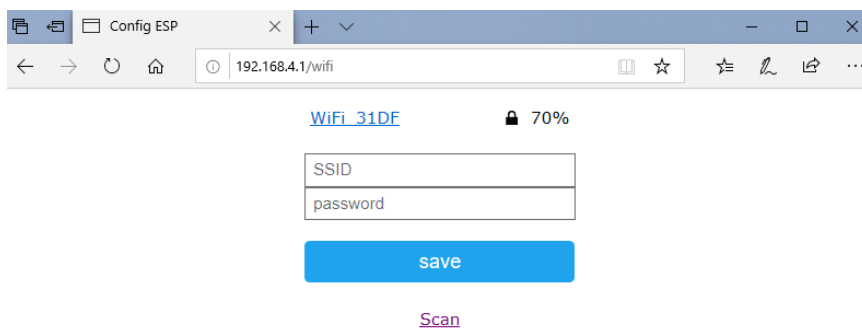
**d) Escolher e ligar a rede SSID dado pelo ESP-12E**



**e) Navegador colocar ip ( 192.168.4.1 )**

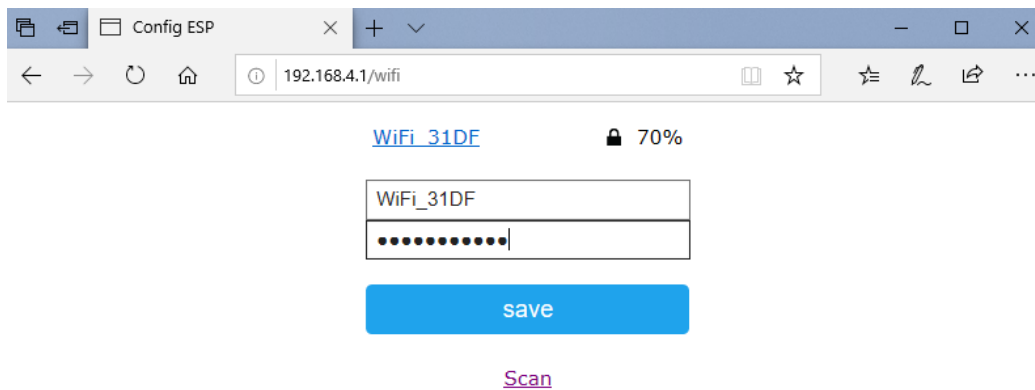


**f) Escolher Configurar WiFi**



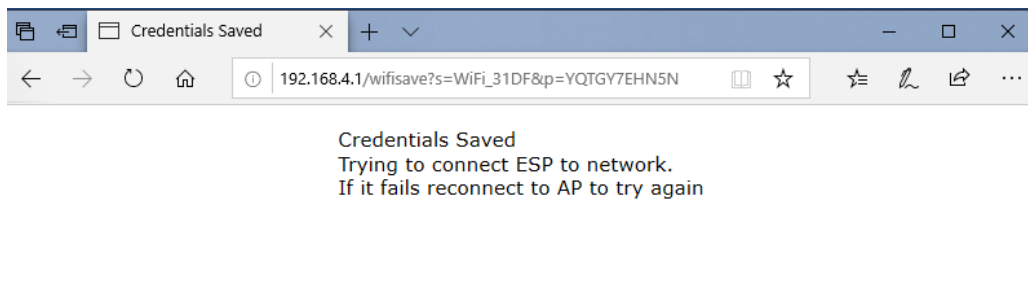
**g) Colocar SSID e a Password da rede pretendida.**

- SSID - Password - Clicar SAVE



**h) FIM da Configuração de rede**

- Desligar e tornar a ligar, está gravada na Flash do ESP-12E



### 3. Testar o 2º Exemplo – dweet

O servidor dweet.io serve para enviou de mensagens, não armazenadas, mas sim temporais, serve para testes, porque é livre de utilização.

Site: <http://dweet.io/>

Sendo o ID “ESP-12E” podemos depois verificar se estamos a enviar dados;

<http://dweet.io/follow/ESP-12E>

a) Programar o ESP-12E para enviou:

1. Escolher a placa como referido no ponto 5 do item 1;
2. O código já está preparado para o sensor de temperatura e humidade (DHT), no entanto encontra-se comentado para o 1º teste.

```
#include <ESP8266WiFi.h>           //Biblioteca ESP
#include <DNSServer.h>              //Biblioteca para resolução de nomes de DNS
#include <ESP8266WebServer.h>       //Biblioteca para apresentação página (suporta 1 cliente)
#include <WiFiManager.h>            //Permite configurar a rede WiFi e guardar na Flash, através
browser                             browser
// Biblioteca do sensor
//#include "DHT.h"
```

```

// Pin ---- para sensor DHT
    // #define DHTPIN 5
// Usa sensor DHT11
    // #define DHTTYPE DHT11
//define o sensor DHT
    // DHT dht(DHTPIN, DHTTYPE, 15);
//define o nosso server a receber os dados
    const char* host = "dweet.io";
    int cont = 0;

    void setup() {
// Start Serial
        Serial.begin(115200);
//WiFi
        WiFiManager wifiManager;
        wifiManager.autoConnect("AutoConnectAP");
        Serial.println("connected...WiFi.... :");
        delay(10);
        // Init DHT
        // dht.begin();
    }
// Loop – ciclo continuo a ser sempre executado, até desligar o ESP
    void loop() {
        cont +=1;                // Apagar só para teste
        Serial.print("Connecting to ");
        Serial.println(host);
// Usa WiFiClient class para crear TCP connections
        WiFiClient client;
        const int httpPort = 80;
        if (!client.connect(host, httpPort)) {
            Serial.println("connection failed");
            return;
        }
// Lê a temperatura e Humidade
        // int h = dht.readHumidity();
        // Read temperature as Celsius
        // int t = dht.readTemperature();
//Estas duas (2) linhas são usadas só para os testes, depois apagar.....
        int t = 2 + cont;
        int h = 99 - cont;

//MEU
// como funciona o envio
// https://dweet.io/dweet/for/ESP-12E?Temp=10&Hum=80
        client.print(String("GET /dweet/for/ESP-12E?Temp=") + String(t) + "&Hum=" + String(h) + "
HTTP/1.1\r\n" +
            "Host: " + host + "\r\n" +
            "Connection: close\r\n\r\n");
        delay(10);
// Leia todas as linhas da resposta do servidor e imprima-as em Serial
        while(client.available()){
            String line = client.readStringUntil('\r');
            Serial.print(line);

```

```

    }
    //só para saberemos que já enviou - print serial no IDE
    Serial.println();
    Serial.println("closing connection");
    delay(10000);
  }
}

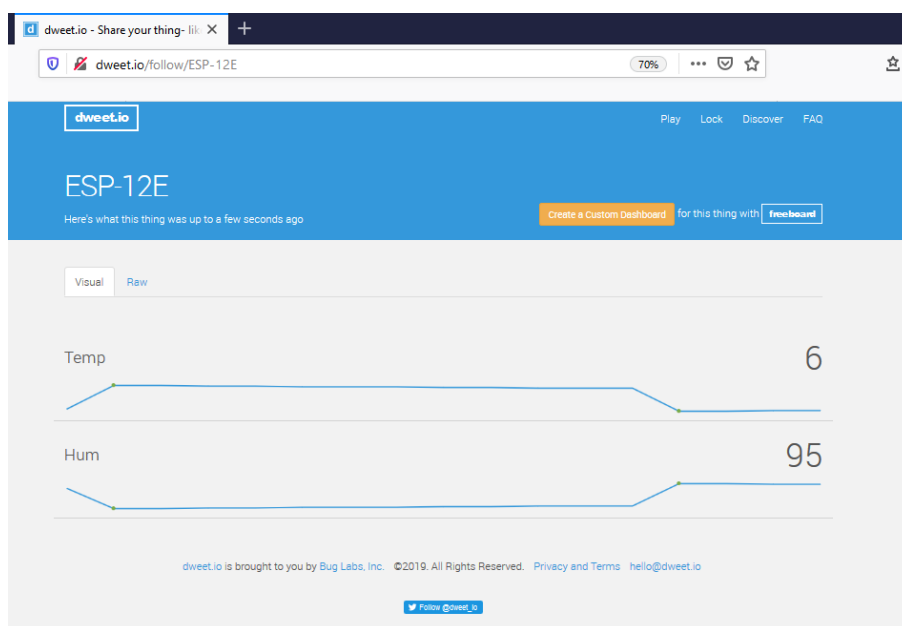
```

3. Efetuar o **Update** para a Placa ESP-12E.

---- no **botão enviar** no IDE do Arduino ----

4. **Verificar** no servidor a recepção de dados:

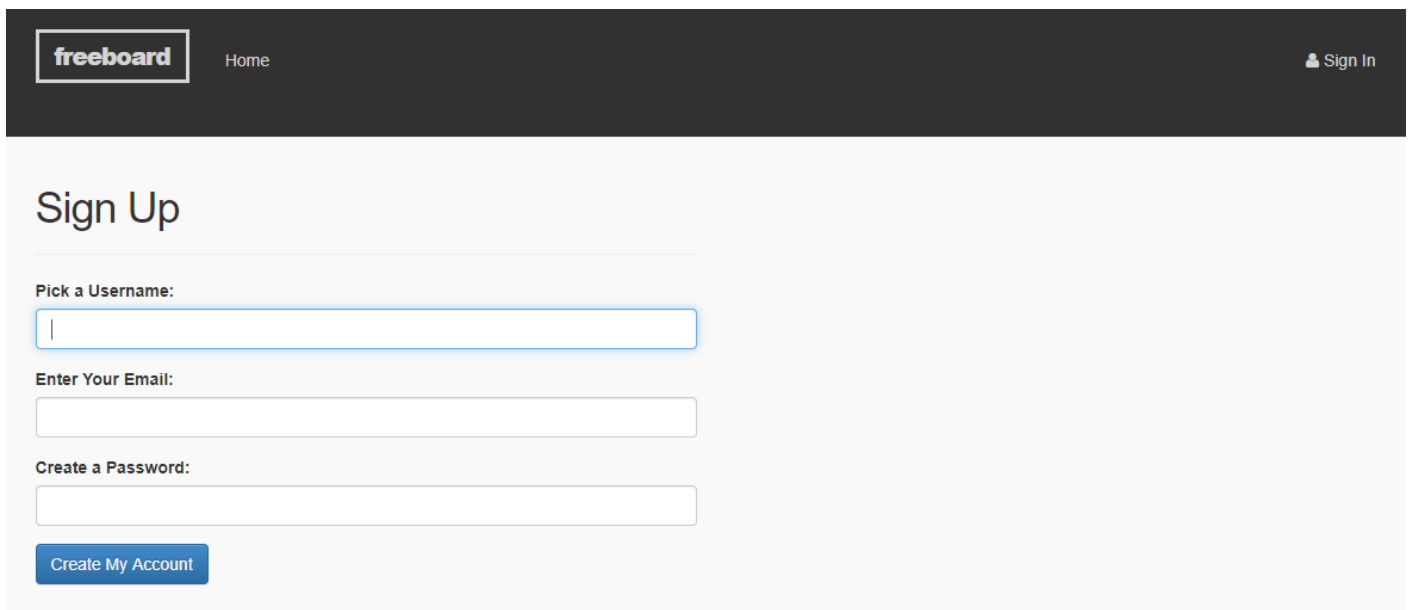
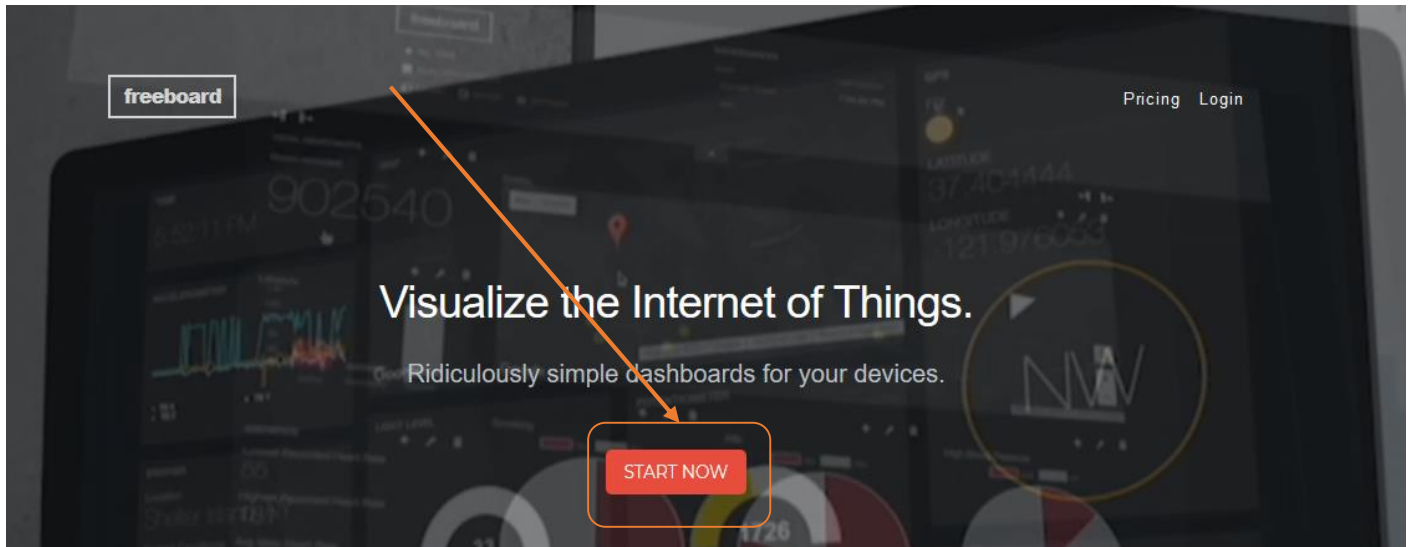
<http://dweet.io/follow/ESP-12E>



b) Configurar **Dashboard** para recepção dos dados;

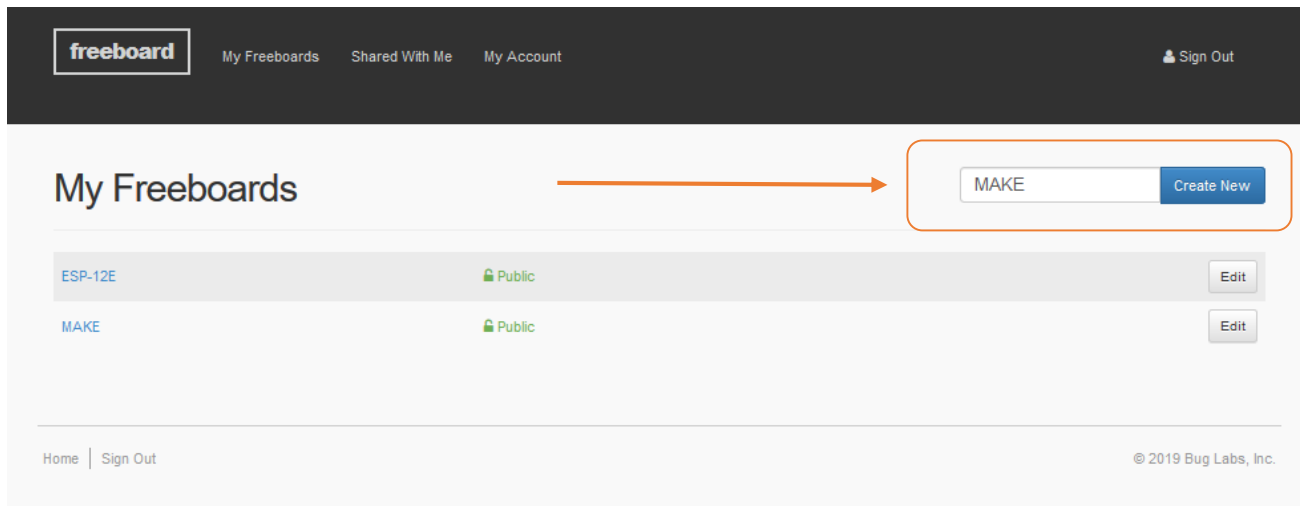
Site: <https://freeboard.io/>

1. criar uma **conta**

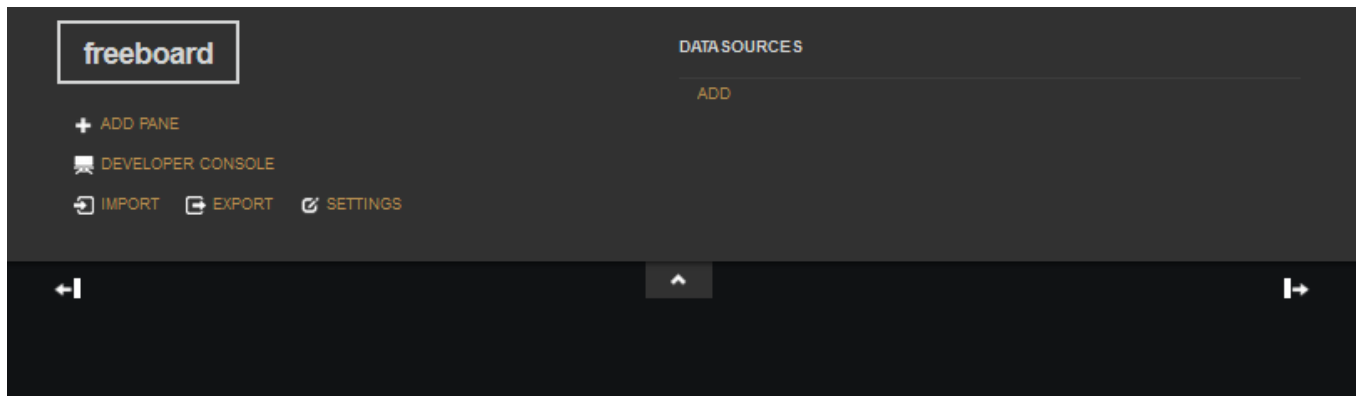
The image shows the 'Sign Up' form on the Freeboard website. The form is titled 'Sign Up' and includes three input fields: 'Pick a Username:', 'Enter Your Email:', and 'Create a Password:'. Below these fields is a blue button labeled 'Create My Account'. The form is set against a dark header with the 'freeboard' logo and a 'Home' link on the left, and a 'Sign In' link on the right.



## 2. Fazer o Login e Criar um **Freeboard**

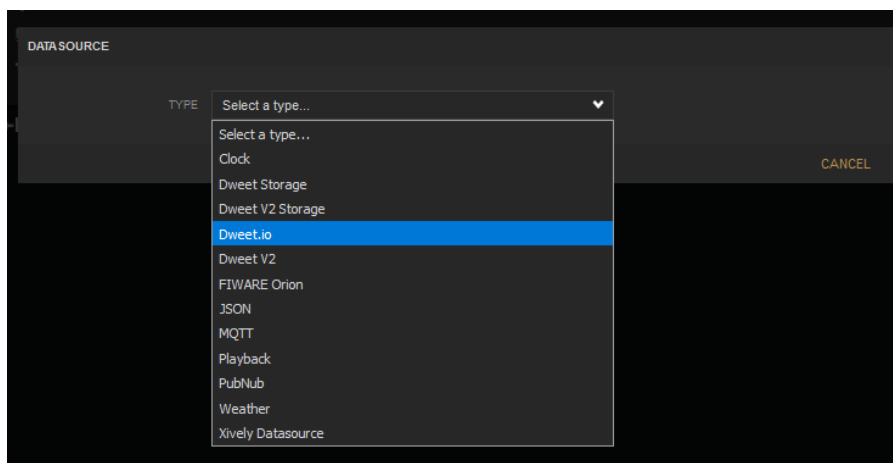


Quando criado, será apresentada esta página:



## 3. Configurar a ligação com server de dados

- a) Clicar em **“ADD”**
- b) Escolher o servidor **“Dweet.io”**



c) Configurar o nome do nosso sensor

A datasource for connecting to things at [dweet.io](https://dweet.io).

TYPE:

NAME:

THING NAME:   
Example: salty-dog-1

KEY:

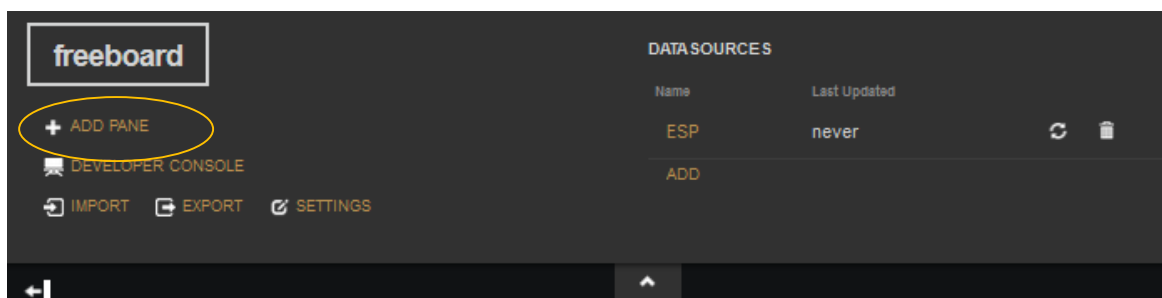
if the thing is not locked, you can ignore this field

SHOW FULL PAYLOAD: ☒ YES

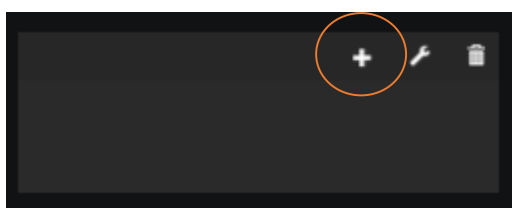
if on, gives access to the full Dweet payload (used to obtain timestamp). If not, only the Content object is captured

SAVE CANCEL

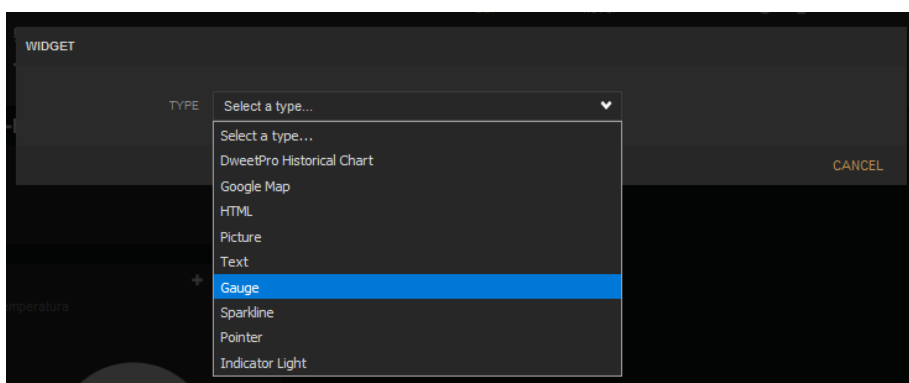
d) Criar um quadro de gráficos – clicar “**ADD PANE**”



4. Criar os gráficos – No painel clicar no +



5. Selecionar o tipo de apresentação de dados



## 6. Configurar os restantes parâmetros

WIDGET

TYPE: Gauge

TITLE: Temperatura

VALUE: datasources["ESP"]["content"]["Temp"] + DATASOURCE .JS EDITOR

UNITS: °C

MINIMUM: 0

MAXIMUM: 100

SAVE CANCEL

### Conclusão:

