FACULDADE DE TECNOLOGIA DE SÃO PAULO

EMERSON CANILLE XAVIER DA SILVA

Armazenamento e Análise de Dados em Nuvem Utilizando ESP8266

SÃO PAULO

2023

FACULDADE DE TECNOLOGIA DE SÃO PAULO

EMERSON CANILLE XAVIER DA SILVA

Armazenamento e Análise de Dados em Nuvem Utilizando ESP8266

Trabalho submetido como exigência parcial para a obtenção do Grau de Tecnólogo em Análise e Desenvolvimento de Sistemas

Orientador: Professor Milton Silva da Rocha

SÃO PAULO
2023

RESUMO

Este projeto teve como objetivo o desenvolvimento de uma aplicação simples de Internet das Coisas, capaz de apresentar e analisar dados obtidos de um sensor DHT22 em um dashboard na plataforma Tago.io, utilizando o módulo NodeMCU-ESP12 para captura e envio dos dados para a nuvem.

Palavras-chave: Internet das Coisas, NodeMCU-ESP12, ESP8266, DHT22, dados, nuvem

ABSTRACT

This project aimed to develop a simple Internet of Things application, capable of presenting and analyzing data obtained from a DHT22 sensor in a dashboard on the Tago.io platform, using the NodeMCU-ESP12 module to capture and send data to the cloud.

Keywords: Internet of Things, NodeMCU-ESP12, ESP8266, DHT22, data, cloud

LISTA DE ILUSTRAÇÕES

SUMÁRIO

1 INTRODUÇÃO

Com o recente avanço em várias tecnologias como Big Data e 5G, a ideia por trás do termo Internet das Coisas criado por Kevin Ashton em 1999, que visualizava os objetos conectados na internet assim como as pessoas, com a capacidade de se comunicarem de forma autônoma, está se tornando cada vez mais presente na realidade atual.

Em paralelo tivemos o avanço tecnológico no hardware, possibilitando a diminuição e barateamento dos componentes eletrônicos, o que resultou em produtos como a plataforma de desenvolvimento Arduino que integra Hardware e Software, que facilitou a entrada dos iniciantes neste ambiente de desenvolvimento.

Unindo estes fatos e alguns outros, chegamos num momento em que surgiu o movimento *maker*, que tem origem no conceito do DIY (do inglês, “do it yourself”), que significa “faça você mesmo”. No universo das Internet das Coisas esse movimento é muito presente, graças ao fácil acesso ao hardware com capacidade de se comunicar com as placas de desenvolvimento Arduino e o próprio NodeMCU-ESP12.

Este trabalho apresenta o desenvolvimento de uma infraestrutura da Internet das Coisas e hardware de baixo custo, capaz de capturar dados do sensor de humidade e temperatura através do sensor DHT22 do módulo NodeMCU-ESP12 e enviar estes dados para o serviço de nuvem da Tago via WiFi. Estes dados são analisados e apresentados via um *Dashboard*, um painel personalizável que apresenta informações, tais como: variáveis de temperatura e umidade.

O ambiente de desenvolvimento deste projeto foi bem simples, sendo necessário além do hardware citado, um computador ou notebook intermediário para que seja feita a programação e a configuração dos serviços em nuvem.

Ao final deste trabalho o projeto atingiu seu objetivo de enviar os dados para nuvem para análise e apresentação em *Dashboard*, e foram citadas algumas possibilidades de aplicações e expansão do projeto.

2 OBJETIVOS

2.1 OBJETIVO GERAL

O presente estudo tem como objetivo desenvolver e testar um sistema de captura de dados com capacidade de envio para nuvem por meio de requisições HTTP, para que os dados possam ser tratados e apresentados em um dashboard, utilizando como circuito de aquisição de dados o módulo NodeMCU-ESP12.

2.2 OBJETIVOS ESPECÍFICOS

- Preparação do ambiente de desenvolvimento para o NodeMCU-ESP12
- Leitura de dados do sensor DHT22
- Utilizar a API da Tago.io para escrita e leitura de dados
- Armazenamento de dados na plataforma Tago.io
- Apresentação de dados no *Dashboard* da Tago.io
- Análise de dados na Tago.io utilizando a ferramenta *Analysis*
- Automação da análise de dados utilizando a ferramenta *Run*

3 METODOLOGIA

3.1 MATERIAIS

O projeto consiste no sensor DHT22 conectado ao módulo NodeMCU-ESP12. O módulo usa mecanismo de acesso baseados na internet para enviar e buscar dados na nuvem através da API da Tago.io.

O ambiente de desenvolvimento utilizado na programação do NodeMCU-ESP12 será o Platform.io como extensão do editor de texto Visual Studio Code.

3.2 PESQUISA

3.2.1 VISUAL STUDIO CODE

Lançado em 2015 pela Microsoft, o Visual Studio Code (VSCode), disponível para download em https://code.visualstudio.com, é um dos principais editores de código-fonte da atualidade. O VSCode é um editor de texto conhecido por ser um editor de código *open-source* muito intuitivo. Além disso, ele também é muito popular por ser multiplataforma e estar disponível para os sistemas operacionais Linux, Mac e Windows.

3.2.2 PLATFORM.IO

Platform.io é um ecossistema de *open-source* para desenvolvimento de software para sistemas embarcados. Oferece suporte para diversos tipos de placas e módulos, o que possibilita desenvolver para diversos microcontroladores utilizando apenas uma ferramenta. A forma mais comum de ser utilizado é como uma extensão do editor de texto VSCode.

3.2.3 NodeMCU-ESP12

O módulo NodeMCU-ESP12 é uma placa que foi criada para facilitar o desenvolvimento de protótipos utilizando o chip ESP8266. Ele consiste na combinação de um módulo ESP-12, chip de interface Serial-USB, regulador de tensão, LEDs indicadores, botões de controle (Reset e Flash) e barramentos de pinos para permitir o uso em Protoboards. (MURTA).

O ESP8266 é um SoC (System-on-Chip), ou seja, em um único chip temos CPU, memória, várias interfaces de comunicação, inclusive WiFi e Bluetooth. O que tornou o ESP8266 muito popular entre a comunidade Maker e projetos de Internet of Things (IoT).

3.2.4 SENSOR DE UMIDADE E TEMPERATURA DHT22

O sensor de baixíssimo custo DHT22 é capaz de mensurar temperatura e umidade relativa do ar e transformar essa informação em um sinal digital, é muito simples de ser utilizado, porém com intervalo entre medidas da ordem de 2 segundos.

3.2.5 PLATAFORMA TAGO.IO

A plataforma TAGO IO possibilita a integração com redes de monitoramento para processamento dos dados e assim o usuário poderá ter acesso a informações para monitoramento através de Dashboards. Além da possibilidade de gerar análises gráficas, o usuário poderá também atribuir que a plataforma tome decisões a partir de determinados parâmetros.

3.3 MÉTODO

3.3.1 HARDWARE

Para a realização do projeto, foi utilizado o seguinte hardware:

- Placa: NodeMCU-ESP12
- Sensor: DHT22

3.3.2 SOFTWARE

O software utilizado no projeto foi o editor de texto Visual Studio Code, com a extensão Platform.io instalada, tendo em vista que esta extensão fornece uma IDE para embarcados.

Todo o desenvolvimento do código para o NodeMCU-ESP12 foi utilizando a framework do Arduino, assim é possível codificar de forma quase idêntica ao da plataforma Arduino IDE e também utilizar suas bibliotecas.

3.3.3 MONTAGEM DO PROTÓTIPO

O protótipo deste projeto ficou bem simples, pois possui apenas um módulo NodeMCU-ESP12 conectado no sensor DHT22. As conexões podem ser verificadas na figura a seguir.

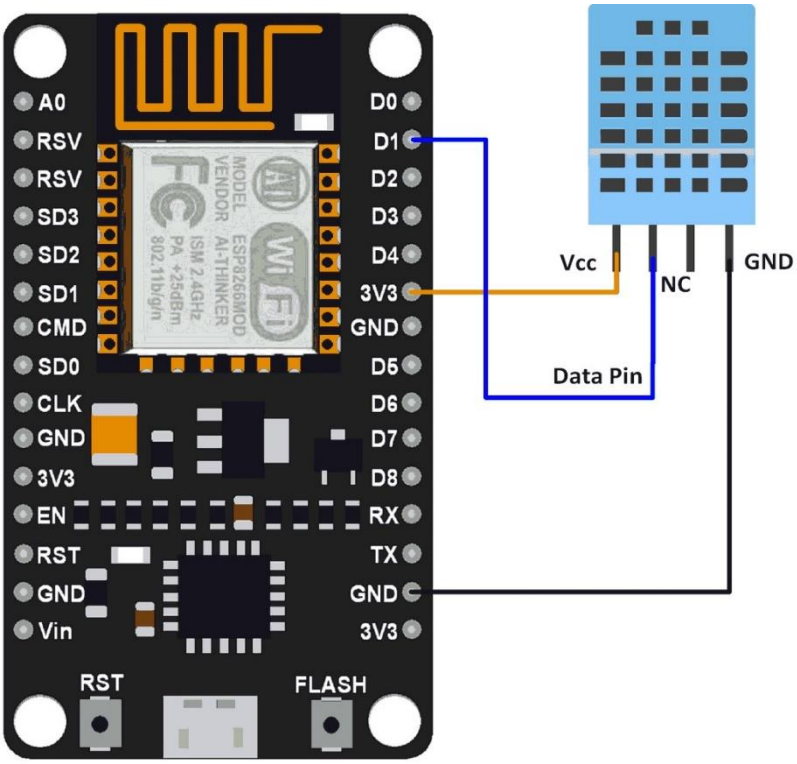

Figura 1: Protótipo do projeto

3.3.4 DESENVOLVIMENTO

Para prosseguir com etapas seguintes foi necessário criar uma conta na plataforma Tago. A conta criada vem com o plano gratuito e possui alguns limites, mas nada que possa atrapalhar um projeto de pequeno porte. A conta pode ser criada no endereço https://tago.io/. Além de estar com o Visual Studio Code instalado e com a extensão do Platform.io embutida.

3.3.4.1 CRIAÇÃO DO DEVICE

Device é o link entre o hardware e os dados na plataforma. A criação do *Device* é obrigatória para que possamos enviar ou requisitar dados. A comunicação entre o

hardware e a Tago é feita através de métodos HTTP utilizando o formato JSON (JavaScript Object Notation).

Para criar um *Device* foi necessário entrar na conta, no menu clicamos na opção “Devices”.

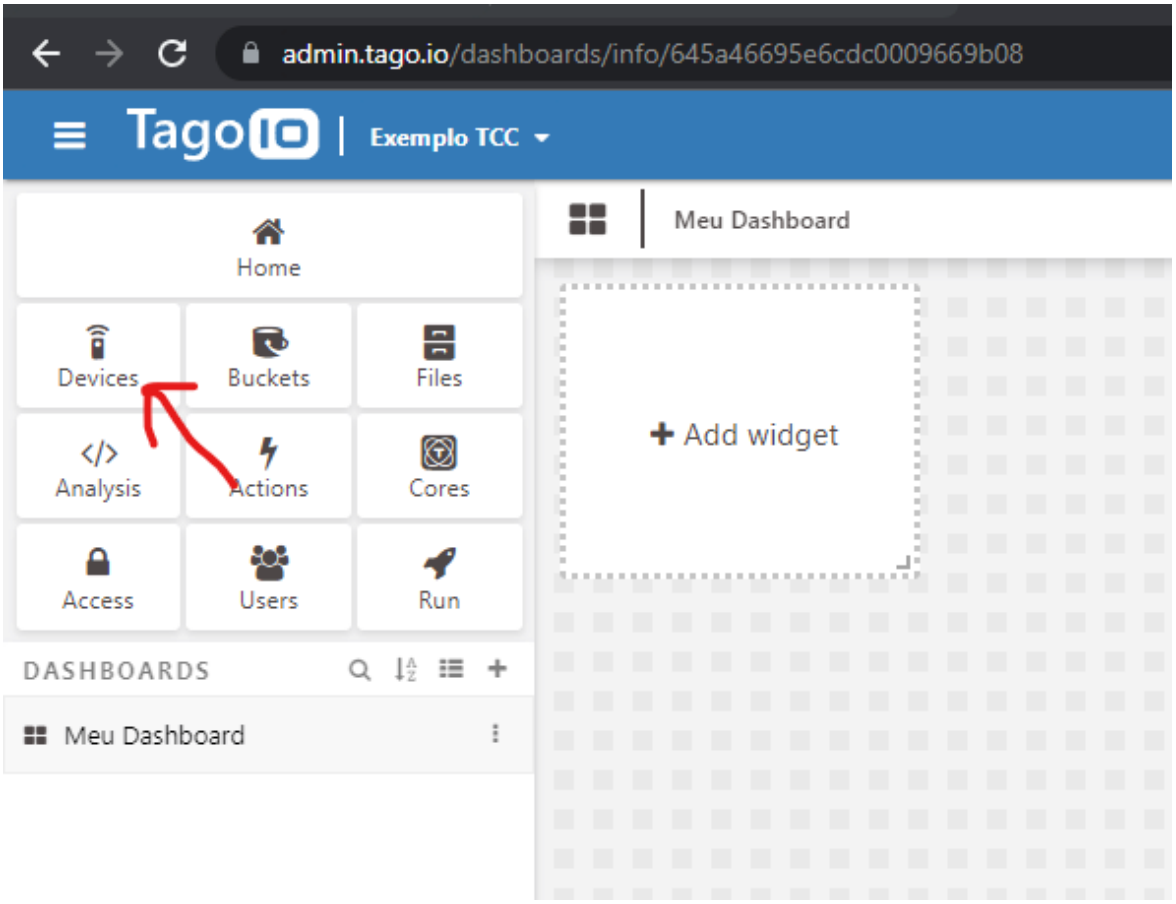

Figura 2: Criação de um Device na Tago

Na seleção de conector, foi selecionado o conector “Custom HTTPS”, pois este conector é capaz de funcionar com qualquer hardware que suporte o protocolo HTTP/HTTPS.

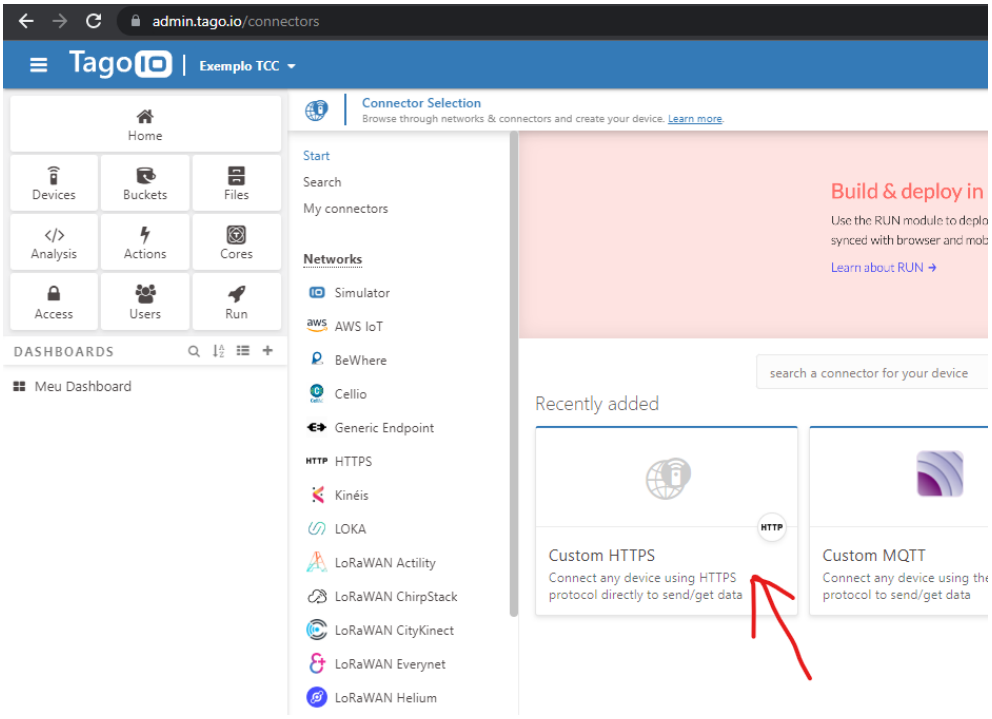

Figura 3: Seleção do conector do Device

Na próxima etapa, bastou inserir o nome do *Device* e confirmar sua criação no botão “Create my Device”.

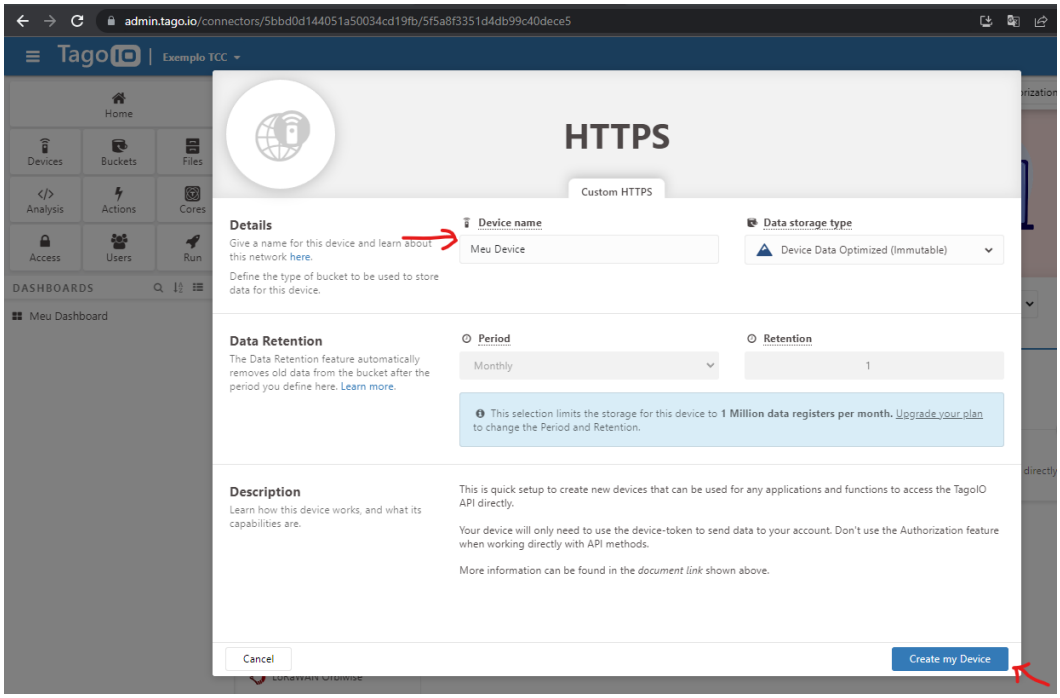

Figura 4: Nomeação e confirmação de criação do Device

Por fim, a Tago apresentou a confirmação da criação do *Device* e do bucket correspondente. Um *bucket* nada mais é do que o local onde os dados do *Device* são armazenados. Quando criamos um *Device*, a Tago automaticamente cria um *bucket* para este *Device* utilizando o mesmo nome, descrição e tags.

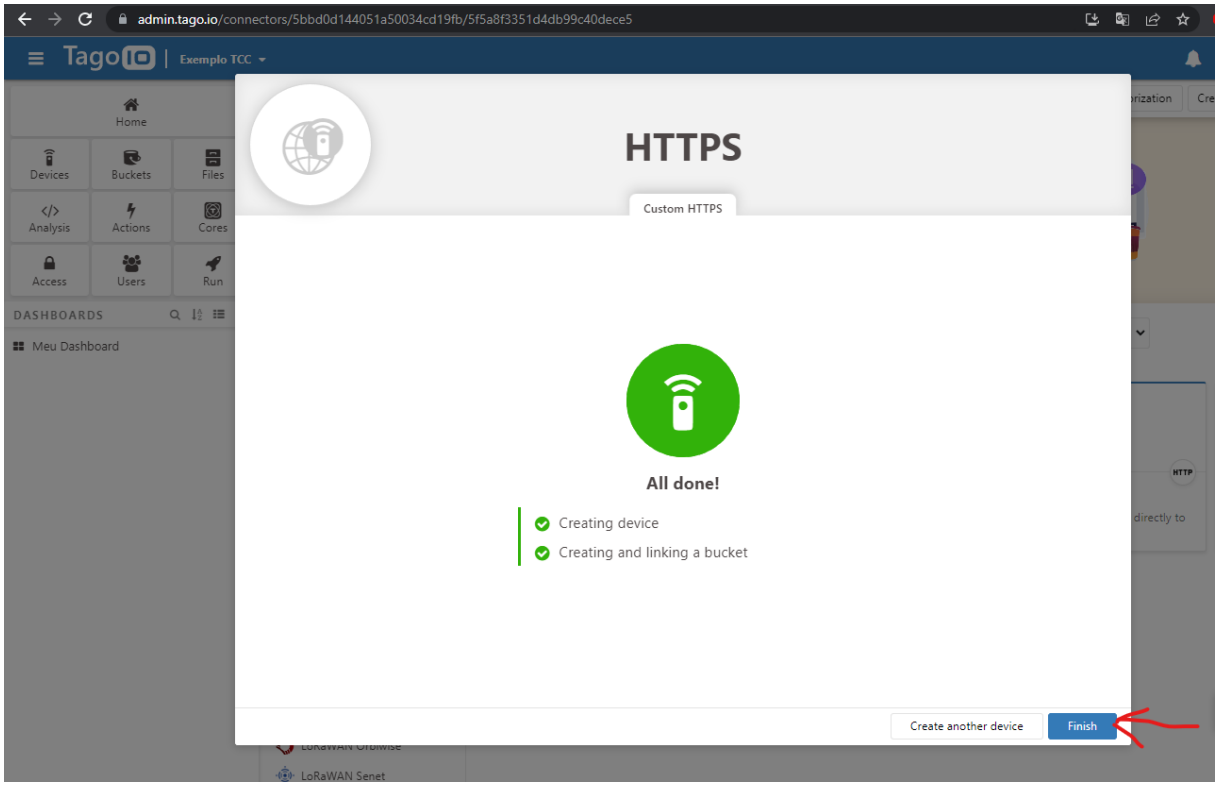

Figura 5: Fim da criação do Device

Seguindo estes passos, foi criado o *Device* responsável por representar o módulo NodeMCU-ESP12, então todo dado enviado pelo módulo irá ser representado pelo *Device* na Tago.

A plataforma Tago fornece um emulador para enviar dados para um *Device*, basta acessar “Devices > O *Device* desejado > Emulator”. Por padrão vem como “JSON payload” onde podemos inserir o corpo da requisição no formato JSON, enviar a requisição e verificar a resposta da API da Tago.

Na figura a seguir, foi feita uma simulação das variáveis do sensor DHT22. Assim a Tago criou as variáveis que nosso módulo NodeMCU-ESP12 irá capturar do sensor DHT22 e enviar para a Tago.

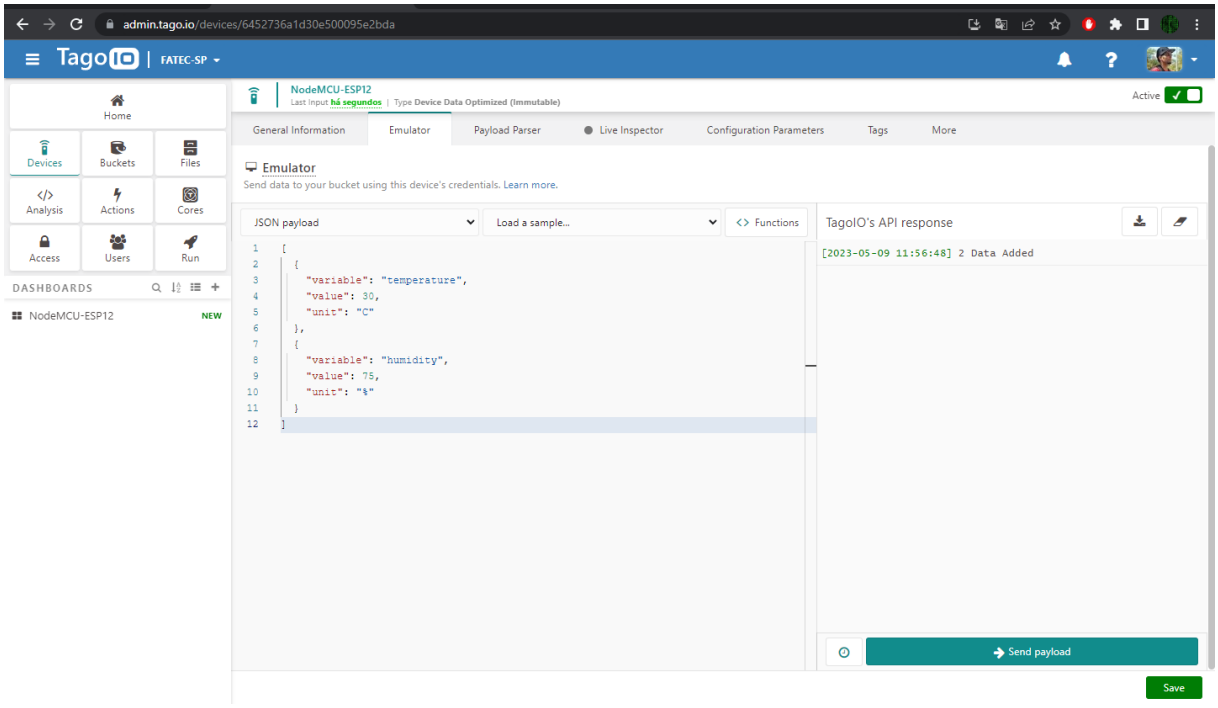

Figura 6: Simulando envio de dados para Tago

É possível verificar os registros indo em “Buckets > Nome do Bucket desejado > Variables”.

3.3.4.2 CRIAÇÃO DO DASHBOARD

Com um *Device* criado, foi possível criar um *Dashboard* com alguns *Widgets*. Para atingir este objetivo, foi necessário ir na opção “Add Dashboard” na página Home da Tago, conforme a seguinte figura.

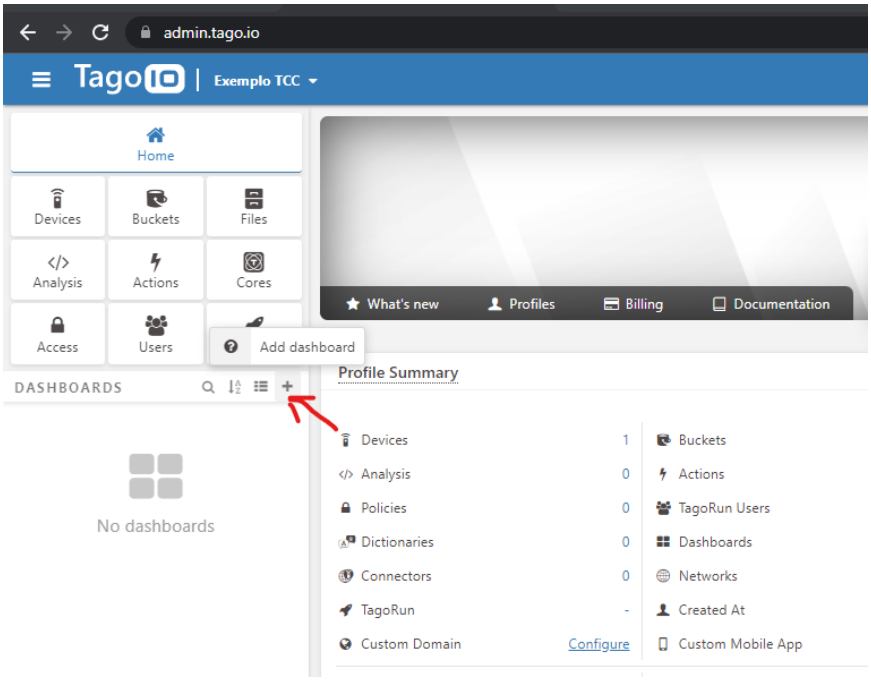

Figura 7: Criando Dashboard

Após esta etapa, foi necessário nomear o *Dashboard*, utilizamos a opção “Normal” por conta da baixa complexidade do projeto e então confirmamos a criação.

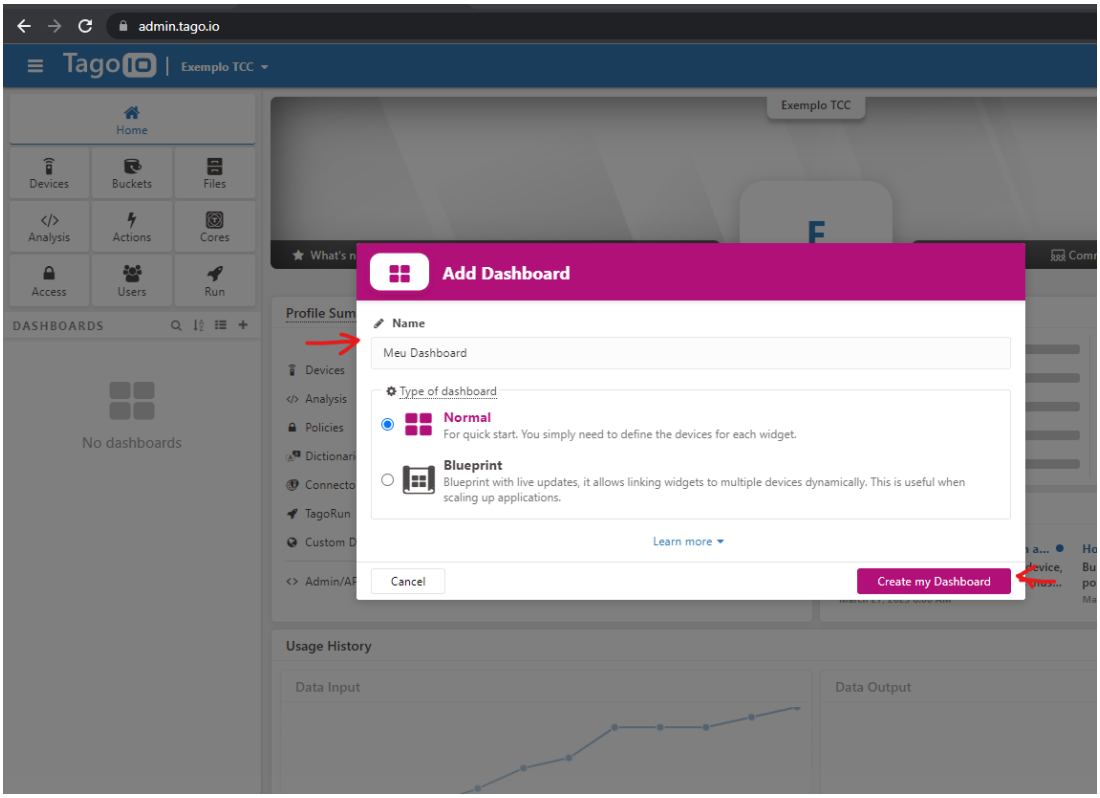

Figura 8: Nomeação e confirmação de criação do Dashboard

3.3.4.3 CRIAÇÃO DE WIDGETS

Neste ponto tínhamos um *Dashboard* vazio, pronto para receber os *Widgets* que apresentarão os dados enviados pelo NodeMCU-ESP12. Para criar um *Widget* bastou selecionar o *Dashboard* desejado e clicar na opção “Add widget”.

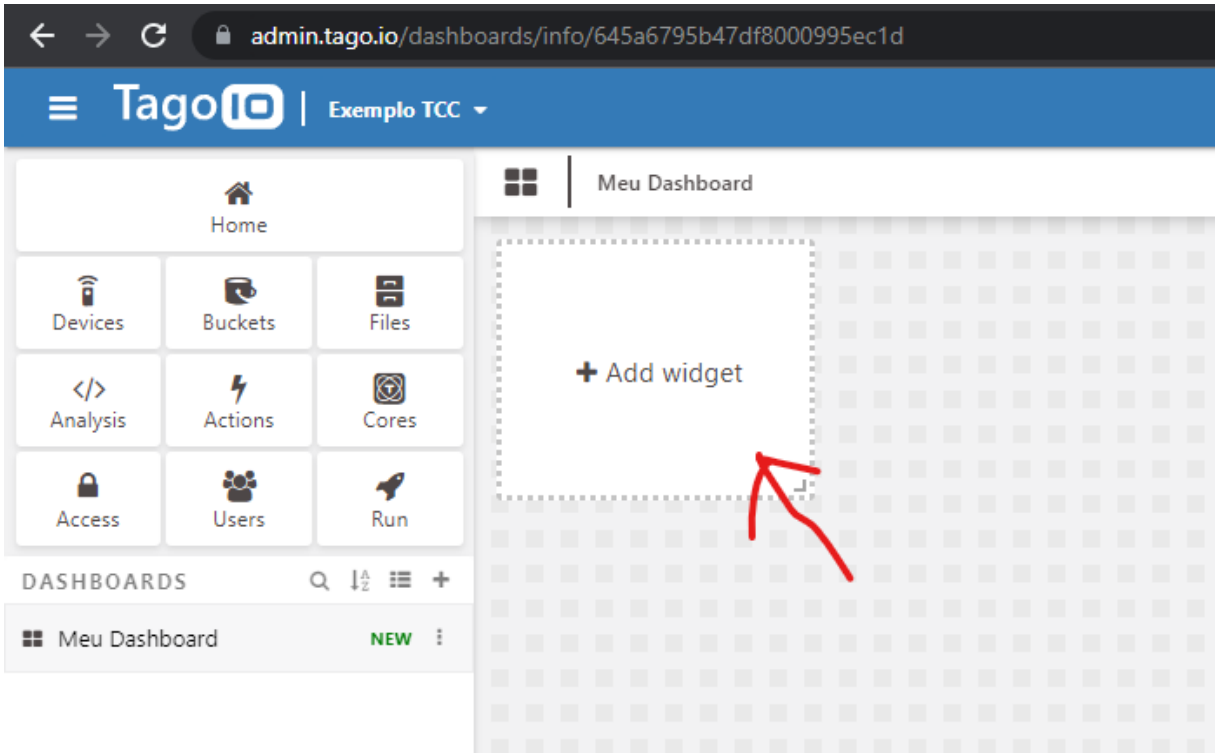

Figura 9: Criação de um Widget

A Tago oferece uma grande variedade de *widgets* prontos, para apresentar as informações de umidade e temperatura foi utilizado o *widget* Line, ele mostra um gráfico de linhas. Assim que foi selecionado apresentou alguns parâmetros para torna-lo funcional, o mais importante é o campo “Data from” que irá definir de onde virão os dados apresentados no gráfico.

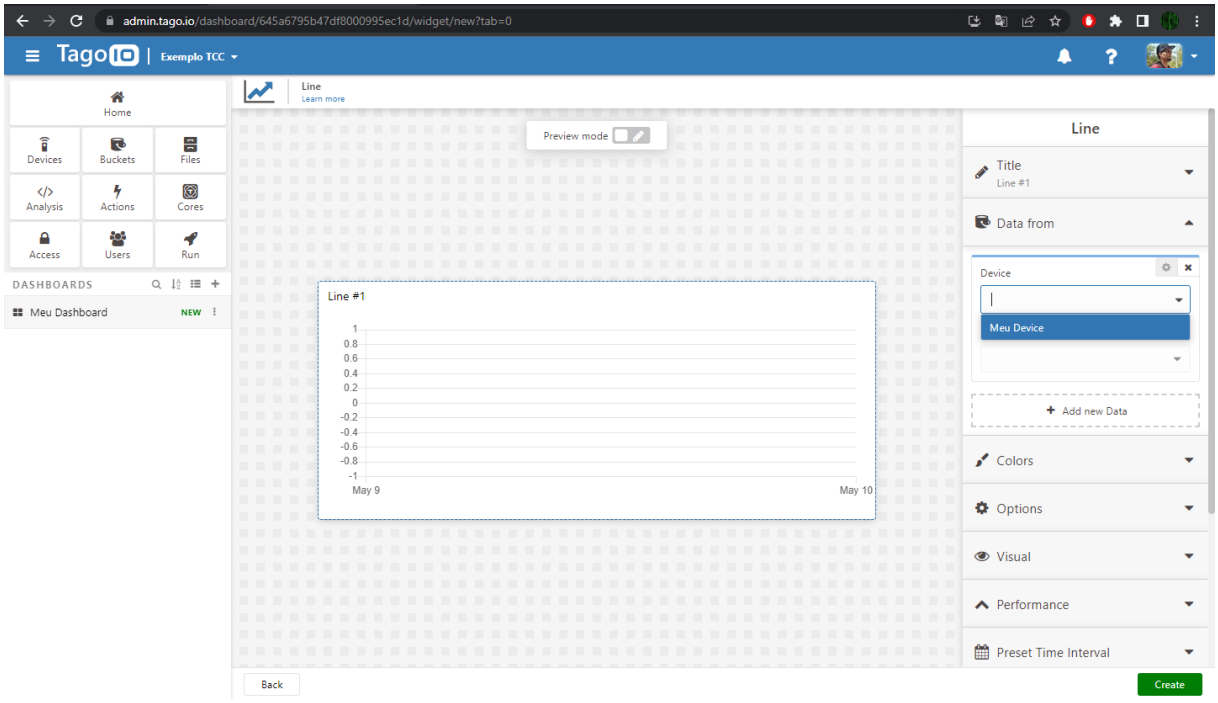

Figura 10: Seleção do Device de origem dos dados

Após ter selecionado o *Device* de origem dos dados, basta ir inserindo as variáveis que desejamos que o gráfico apresente. Na Figura 11 temos duas variáveis, a *temperature* e a *humidity*.

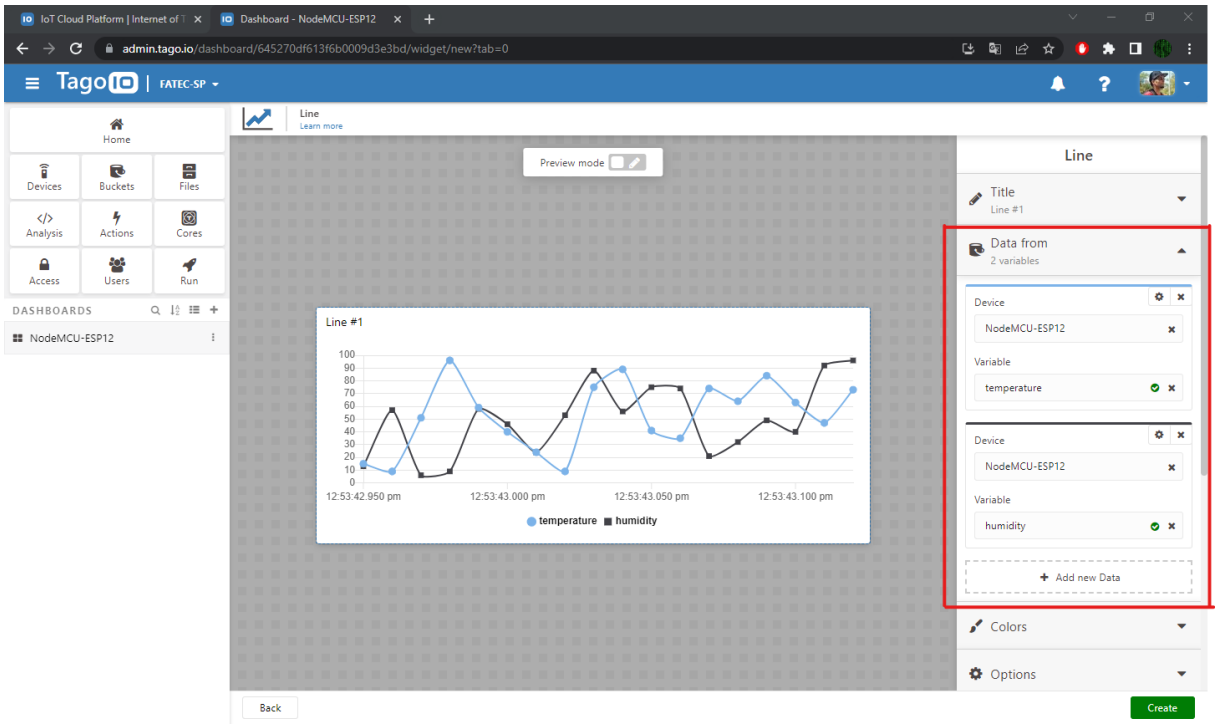

Figura 11: Configuração do campo "Data from"

No caso do sensor DHT22 que irá fornecer dados de temperatura e umidade relativa do ar, o resultado são duas linhas no gráfico. É possível adicionar quantas variáveis forem necessárias utilizando o campo “Add new Data”.

Os demais campos de configuração permitem alterar o título, a unidade das variáveis, a cor das linhas etc.

3.3.4.4 LEITURA DO SENSOR DHT22

Com todo ambiente preparado na Tago, foi necessário dar início a programação do módulo NodeMCU-ESP12. Iniciamos por fazer a leitura das variáveis do sensor DHT22.

O fabricante do sensor DHT22 é a Adafruit Industries, já disponibiliza a biblioteca DHT.h com as funções para leitura do sensor DHT22, lembrando que esta tem como dependência a biblioteca Adafruit_Sensor.h, e ambas estão disponíveis no GitHub da Adafruit, ou no gerenciador de bibliotecas do Platform.io.

Foi bem simples de conseguir realizar a leitura do sensor DHT22, bastou seguir o exemplo fornecido no GitHub da Adafruit e verificou-se que estava tudo certo, utilizando o código a seguir.

```ino
#include "DHT.h"

#define DHTPIN D2 // pino conectado ao sensor DHT

#define DHTTYPE DHT22 // DHT 22  (AM2302), AM2321

DHT dht(DHTPIN, DHTTYPE); // Instancia da Classe DHT

void setup() {
  Serial.begin(9600);
  dht.begin(); // inicia o DHT22
}

void loop() {
  // Delay minímo para o sensor atualizar o valor
  delay(2000);

  // Função de leitura da umidade relativa do ar
  float h = dht.readHumidity();
  // Função de leitura da temperatura em C°
  float t = dht.readTemperature();

  // Verificação dos valores lidos
  if (isnan(h) || isnan(t)) {
    Serial.println(F("Falha na leitura do DHT22!"));
    return;
  }

  Serial.print(F("Umidade: "));
  Serial.print(h);
  Serial.print(F("%  Temperatura: "));
  Serial.print(t);
  Serial.print(F("°C "));
}
```

Figura 12: Código de teste do sensor DHT22

3.3.4.5 CONEXÃO DO NodeMCU-ESP12 NO WIFI

Para continuarmos com o projeto, foi necessário realizar a conexão do módulo NodeMCU-ESP12 com uma rede WiFi. Como já foi dito anteriormente, este módulo já vem com um chip de WiFi embutido no ESP12, então bastou utilizar a biblioteca ESP8266WiFi.h disponível no GitHub do ESP8266 Community Forum. Com esta biblioteca basta configurar o nome da rede WiFi, inserir a senha e usar alguns comandos para que o NodeMCU-ESP12 se conecte a rede WiFi. Abaixo segue um código mínimo para efetuar a conexão.

```
#include <ESP8266WiFi.h>

const char* ssid = "nome da rede";
const char* password = "senha da rede";

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

  Serial.print("Conectando-se na rede ");
  Serial.println(ssid);
  WiFi.mode(WIFI_STA); // Configura para o modo STATION
  WiFi.begin(ssid, password); // Inicia o objeto WiFi
  // Aguarda até a conexão ser bem sucedida
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("\nWiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP()); // Mostra o IP adquirido.
}

void loop() {}
```

Figura 13: Código mínimo para conexão WiFi do NodeMCU-ESP12

Com isso o NodeMCU-ESP12 já foi capaz de se comunicar nas redes WiFi, possibilitando prosseguir o projeto.

3.3.4.6 ENVIO DE DADOS PELA TAGO API

O módulo NodeMCU-ESP12 é capaz de realizar requisições HTTP ou até mesmo HTTPS, para situações que requerem maior segurança. Nesta etapa foram inseridas no projeto as bibliotecas ESP8266HTTPClient.h e WiFiClient.h, ambas disponíveis nos mesmos locais do tópico anterior.

Com esse conjunto de bibliotecas foi possível efetuar as requisições HTTP necessárias para se comunicar com a API da Tago. Assim, foi possível enviar os dados do sensor DHT22 para a plataforma da Tago, utilizando o método POST que requer o Token do *Device* alvo.

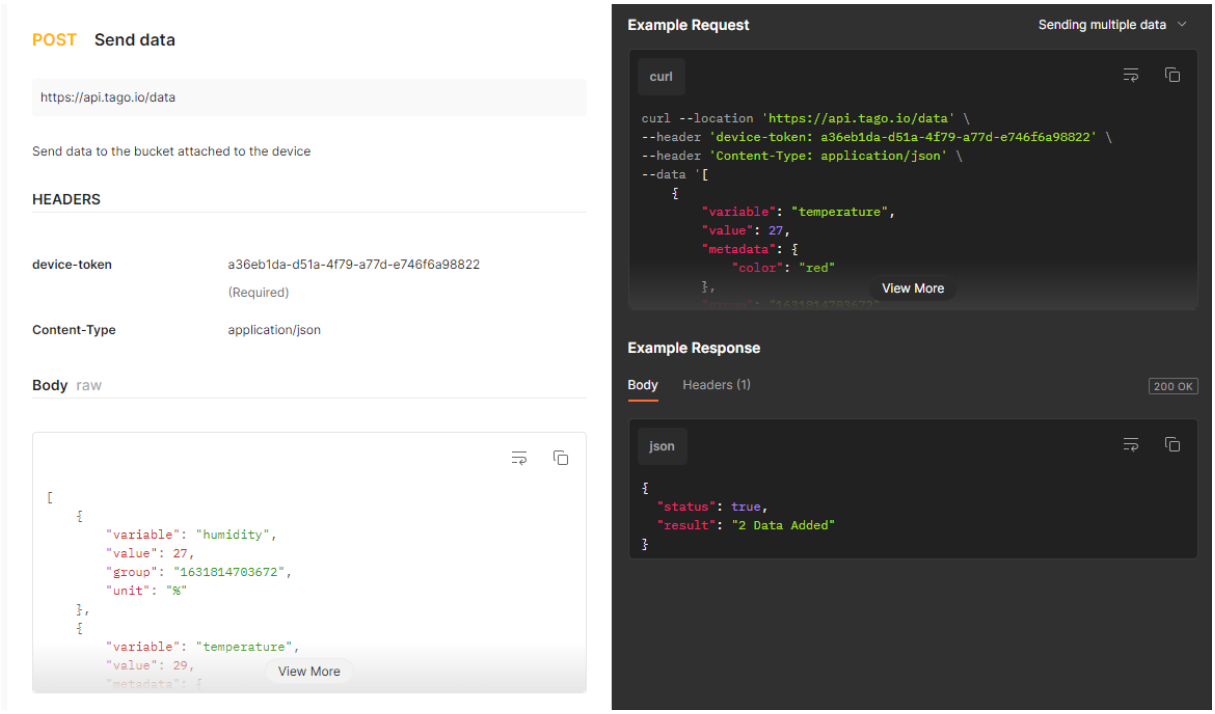

Figura 14: Exemplo do método POST via Device na Tago.io

Como o exemplo acima demonstra, a API da Tago requer um corpo de requisição no formato JSON, para facilitar a construção deste modelo de dado foi utilizada a biblioteca ArduinoJson.h, disponível no site oficial ou na biblioteca da extensão Platform.io. Essa biblioteca tem a finalidade de transformar estruturas de dados como vetores em *Strings* no formato JSON e vice e versa. Assim podemos tanto enviar requisições com o corpo JSON como interpretar as respostas do servidor e transformar o texto JSON em uma estrutura de dados feita por vetores.

Utilizando as bibliotecas citadas, podemos fazer a requisição necessária para enviar os dados lidos do nosso sensor DHT22, utilizando uma conexão WiFi para efetuar a requisição POST para a Tago API.

Como um código de demonstração ficaria muito extenso, recomenda-se que seja verificado o código fonte da classe Device, disponível no repositório GitHub deste projeto. O código encontra-se no endereço https://github.com/jarrhoolyszey/tcc-fatec-sp/blob/main/lib/Device.

Abaixo segue um fluxograma simplificado da rotina seguida pelo NodeMCU-ESP12 para facilitar a compreensão do código fonte.

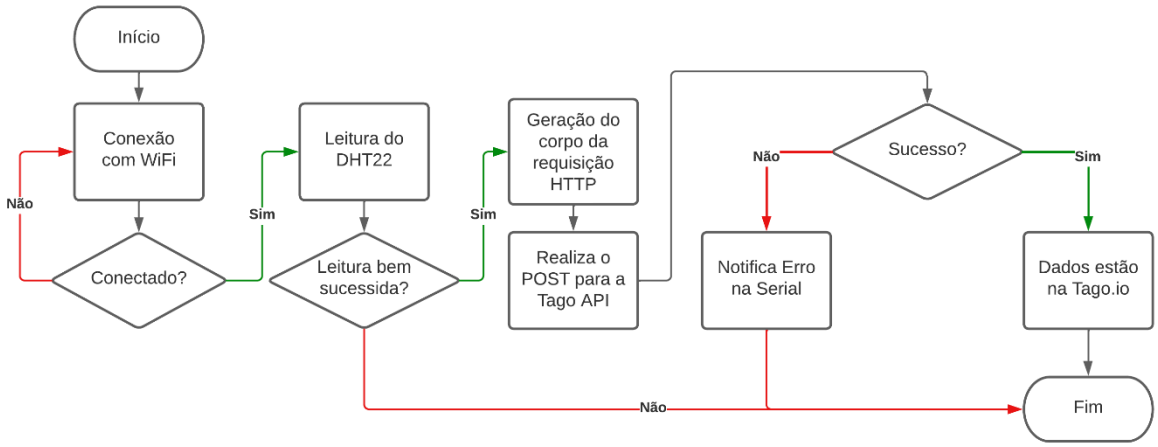

Figura 15: Fluxo simplificado de rotina do módulo NodeMCU-ESP12

3.3.4.7 ANÁLISE DOS DADOS

Para realizar a análise dos dados enviados para a plataforma Tago, usamos a ferramenta *Analysis*. Este recurso é muito poderoso e utiliza scripts para analisar ou manipular dados de qualquer *Device* em tempo real.

Para criar uma nova *Analysis* basta ir na opção “Analysis” na página principal da Tago.io, e então clicar em “Add Analysis”.

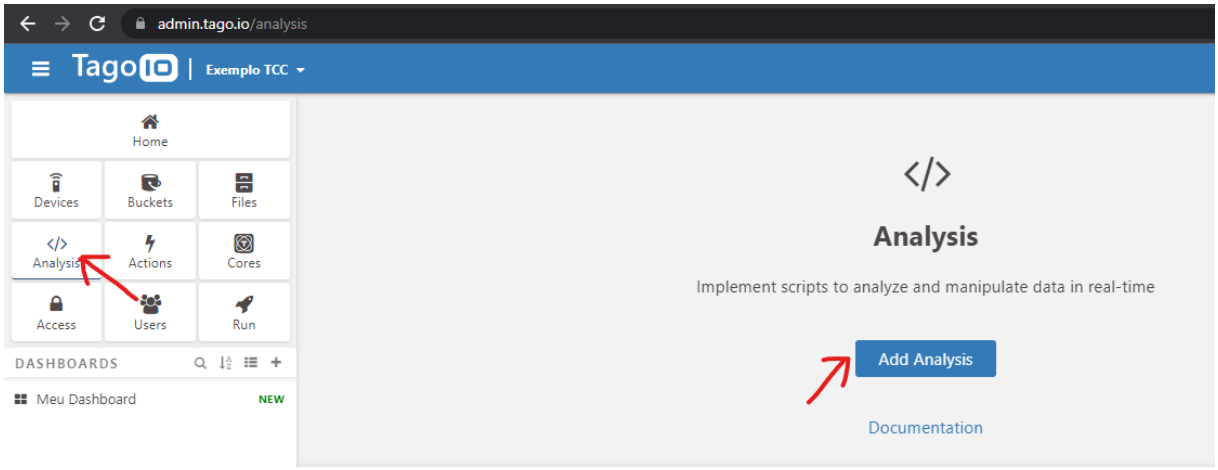

Figura 16: Opção de criar uma nova Analysis

Aparecerá uma janela para configurar a *Analysis*, basta inserir um nome, escolher a linguagem que deseja desenvolver o script, decidir se irá executar o script na própria plataforma da Tago ou em um terminal externo e clicar em “Create my Analysis”.

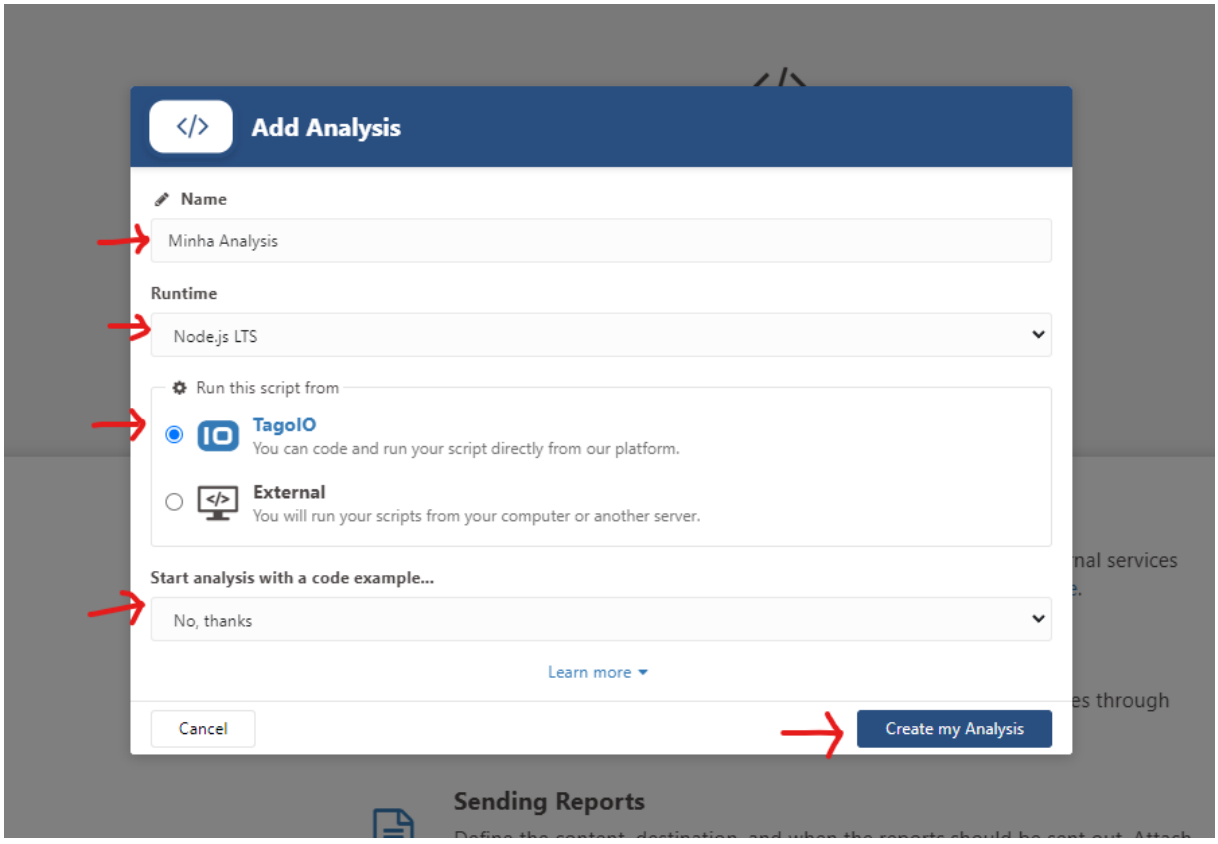

Figura 17: Configurando e criando uma Analysis

Na janela apresentada a seguir, podemos escrever nosso script seguindo as instruções da Tago SDK, utilizando a linguagem selecionada anteriormente, ou podemos carregar algum modelo e reaproveitar o código. Neste projeto carregamos o modelo “Minimum, maximum, and average” para apresentar algumas estatísticas simples dos dados enviados para a Tago.

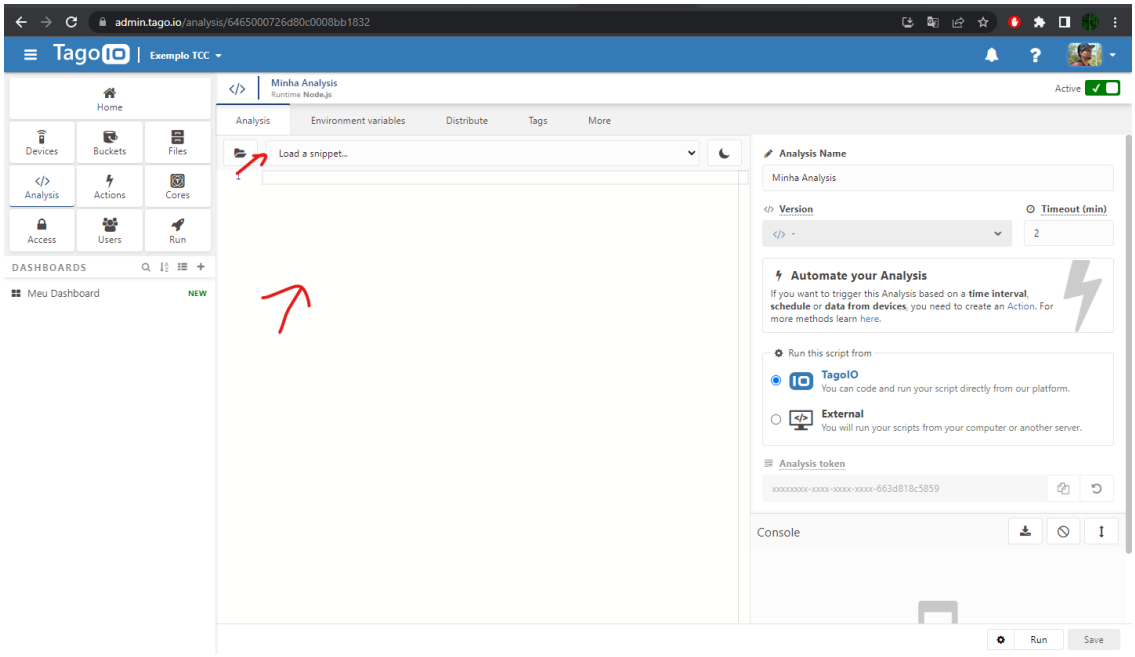

Figura 18: Janela de desenvolvimento do script da Analysis

Neste projeto o script está sendo executado na própria plataforma da Tago, então é necessário inserir uma variável ambiente para que o script funcione. Basta ir no campo “Environment variables” e inserir uma variável chamada “device_token” com o valor do token de seu *Device* criado anteriormente.

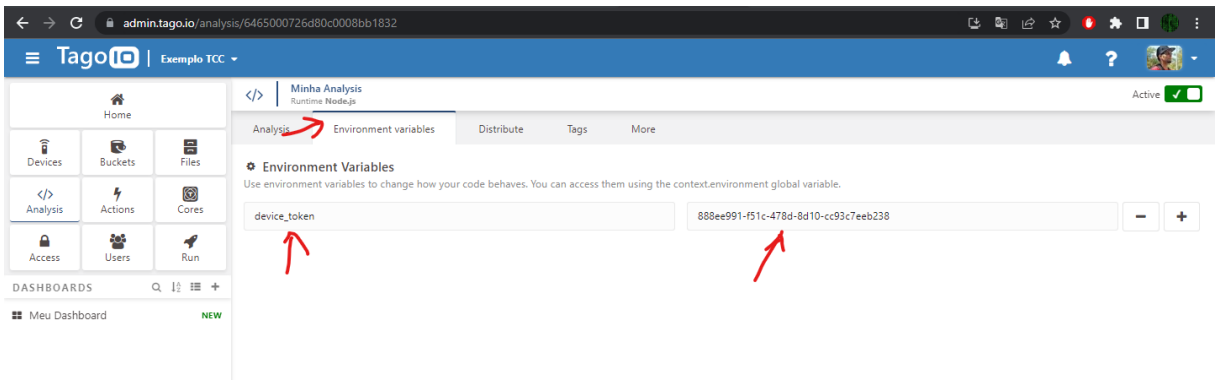

Figura 19: Criação da variável ambiente "device_token"

Feito isso, bastou clicar em “Save and Run”, assim a *Analysis* foi executada e criou as variáveis *temperature_average*, *temperature_maximum*, *temperature_minimum*. Vale ressaltar que para que a *Analysis* produzisse um resultado válido foi necessário que o módulo NodeMCU-ESP12 tivesse enviado dados para a Tago. Outro ponto é que conforme pode ser visto no script “tago_script.js”, as variáveis estatísticas estão utilizando os dados das últimas 24 horas.

3.3.4.8 AUTOMAÇÃO DE ANÁLISE

Até o momento nossa *Analysis* só era executada manualmente, para automatizar esse processo utilizamos a ferramenta *Actions* da plataforma Tago. Esta funcionalidade permite que sejam executadas ações baseadas em eventos determinados por nós. Neste projeto usamos uma *Action* para executar nosso script de *Analysis* toda vez que nosso módulo NodeMCU-ESP12 enviar os dados do DHT22.

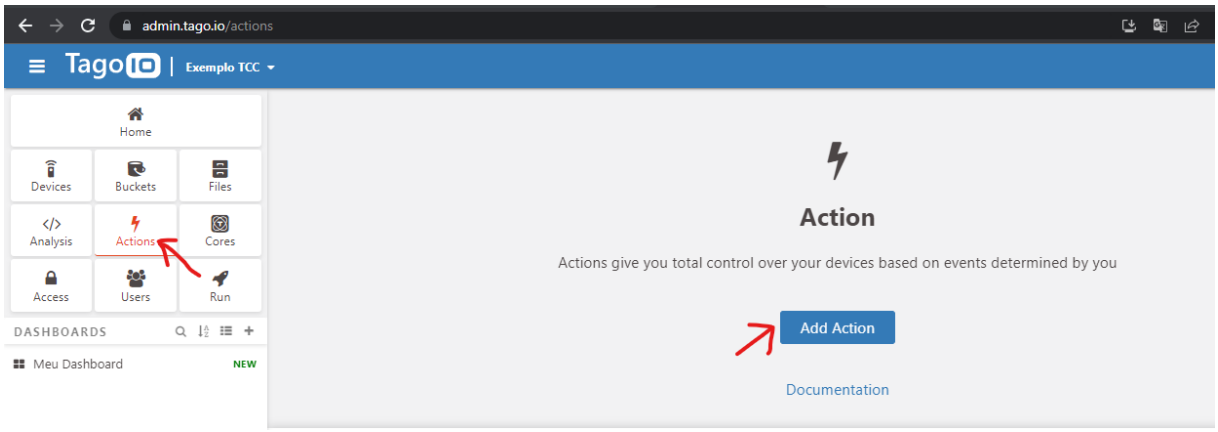

Figura 20: Criação de uma Action na Tago

Como de costume a Tago apresentou uma janela de configuração e nela colocamos o nome da nossa *Action*, escolhemos a opção “Variable” como trigger da ação, tipo da ação escolhemos “Run Analysis” e então colocamos para executar a Analysis criada anteriormente.

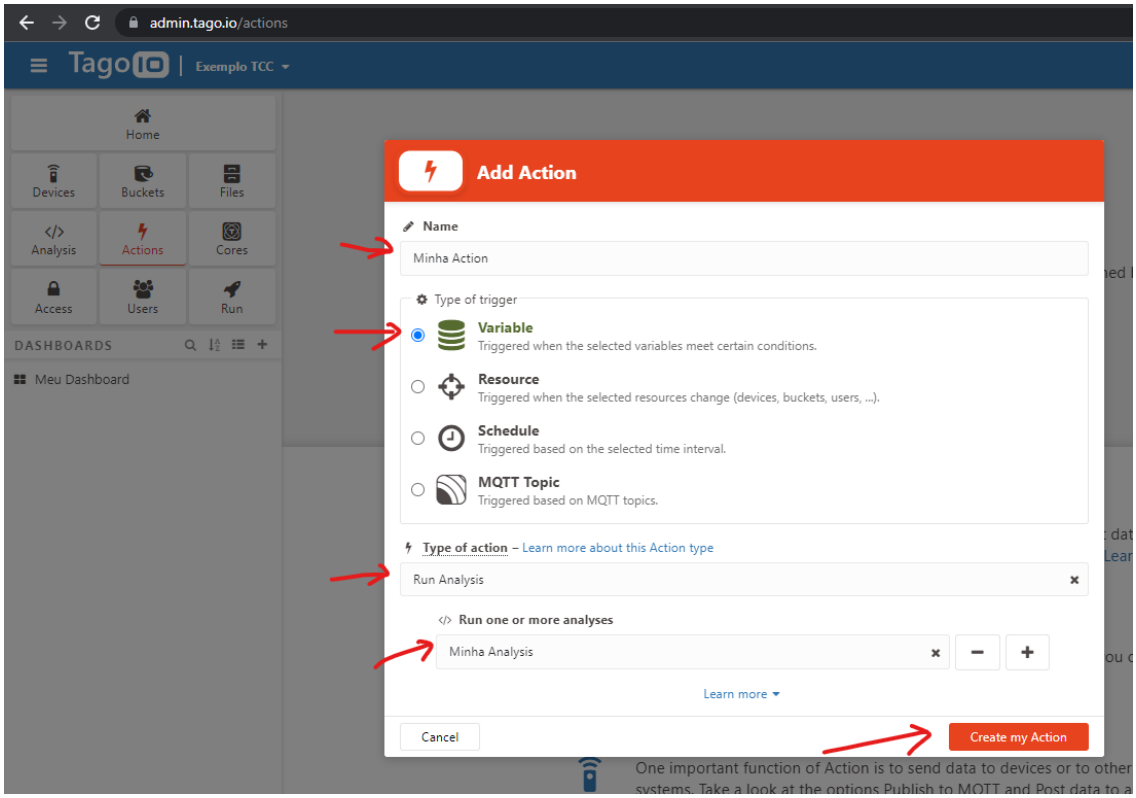

Figura 21: Configuração e criação de uma Action

Na próxima etapa configuramos os parâmetros para executar nossa *Action*, deixamos como Single Device, o *Device* criado neste projeto selecionado, como Trigger deixamos a variável "*temperature is Anything*", assim qualquer atualização feita nessa variável nossa *Analysis* será executada. Há outras formas de configurar o Trigger mas para este projeto está maneira foi suficiente.

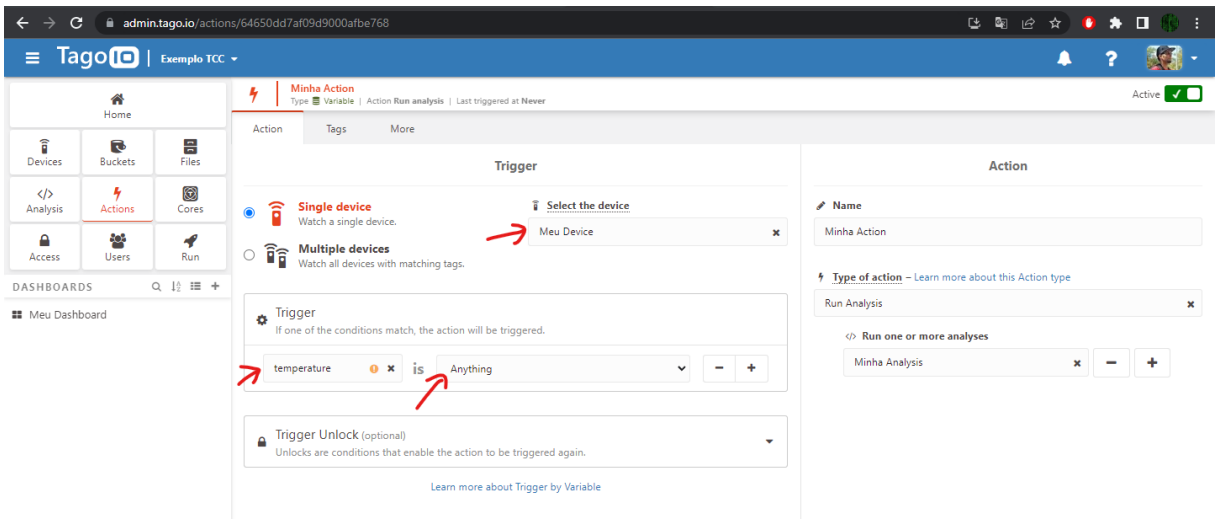

Figura 22: Configuração de Trigger da Action

Após isso, criamos novos *Widgets* no nosso *Dashboard* para apresentar os dados estatísticos gerados pela nossa *Analysis*, e ficamos com algo parecido com a seguinte figura.

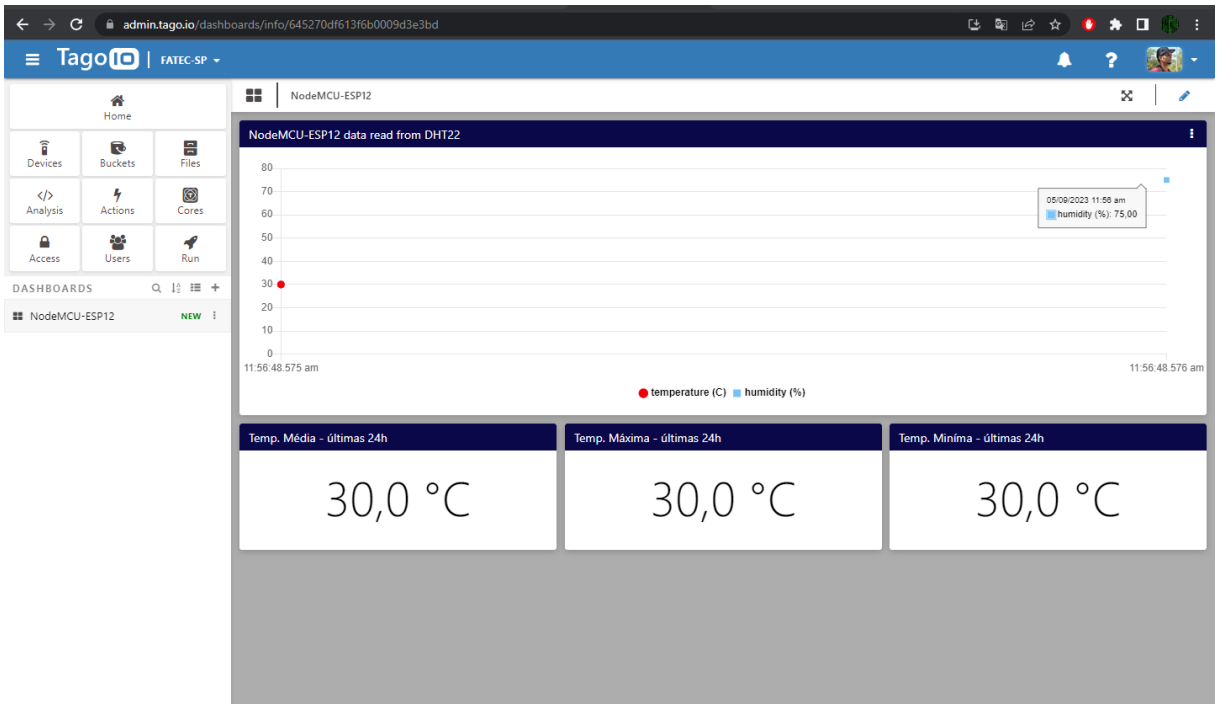

Figura 23: Dashboard com todas variáveis estatisticas

4. CONCLUSÃO

O projeto foi concluído satisfatoriamente, o módulo NodeMCU-ESP12 foi capaz de capturar os dados do sensor DHT22 e utilizar sua funcionalidade WiFi para se comunicar com a API da Tago e alimentar nosso Dashboard conforme a Figura 23. Além de configurarmos a ferramenta *Analysis* e *Run* para que sempre que chegue novos dados na nuvem, nosso Dashboard faça o processamento dos mesmos de forma automatizada para termos os valores de nossas variáveis estatísticas.

A infraestrutura de Internet das Coisas utilizada neste projeto permite uma vasta gama de aplicações, por exemplo, no campo de Industria 4.0 podemos utilizar o módulo para monitorar um termopar de uma estufa e atuar com suas saídas digitais para controlar o aquecimento e ventilação da mesma, de forma remota via Dashboard, podendo automatizar este controle ou fazer manualmente.

Outro exemplo de aplicação, poderia ser o monitoramento da energia de uma instalação e acionamento de gerador monitorando seu nível de combustível, podendo fazer nosso Dashboard enviar um e-mail ou SMS para um responsável ir verificar o que ocorreu com a energia. As aplicações são diversas, tanto para ambiente industrial ou residencial como o caso das Smart Homes.

O nosso Dashboard ficou bem simples, mas nada impede de inserirmos mais Devices monitorando outras coisas como a localização de objetos e etc.

BIBLIOGRAFIA

Página oficial do Visual Studio Code. Disponível em: <https://code.visualstudio.com/>. Acesso em: 02 mai. de 2023.

Página oficial do Platform.io. Disponível em: <https://platformio.org/>. Acesso em: 02 mai. de 2023.

Página oficial da Espressif. Disponível em: <https://www.espressif.com/en/products/socs/esp8266>. Acesso em: 02 mai. de 2023.

Página oficial da Tago.io. Disponível em: <https://tago.io/>. Acesso em: 02 mai. de 2023.

Página oficial do ArduinoJSON. Disponível em: <https://arduinojson.org/>. Acesso em: 17 mai. de 2023.

GitHub oficial da Adafruit. Disponível em: <https://github.com/adafruit/DHT-sensor-library>. Acesso em: 17 mai. de 2023.

GitHub oficial do ESP8266 Community Forum. Disponível em: <https://github.com/esp8266>. Acesso em: 17 mai. de 2023.

MURTA, José Gustavo Abreu. **Guia NodeMCU – ESP12 – Introdução (Parte 1)**. Disponível em: <https://blog.eletrogate.com/nodemcu-esp12-introducao-1/>. Acesso em 02 mai. de 2023.