



Sérgio Silvaneres Pereira Silva

DESAFIO ARGOS / MUNDO JIX
PROJETO DE ENVIO DE DADOS VIA MQTT

Aracaju

2020

1.0 - Introdução:

Creio que Nikola Tesla se sentiria maravilhado ao ver que a sua previsão, de que seria possível se comunicar instantaneamente por todo o mundo usando a tecnologia sem fio, se tornou realidade. Esse fato está tão arraigado em nossa civilização atual que fica difícil pensar sobre as relações humanas sem relacioná-la com a internet. Ela facilita a vida das pessoas de todas as formas, proporcionando, por exemplo, a realização de reuniões no conforto da sua casa, procedimentos médicos à distância e inúmeros outros benefícios.

Dentro destes sistemas sem fio, uma área que vem crescendo abruptamente nos últimos anos é a de internet das coisas, onde, além dos computadores e smartphones, diversos aparelhos também são conectados à rede e executam diversas funções, como o controle de plantas industriais, monitoramento e sensoriamento remoto, automação de casas e etc.

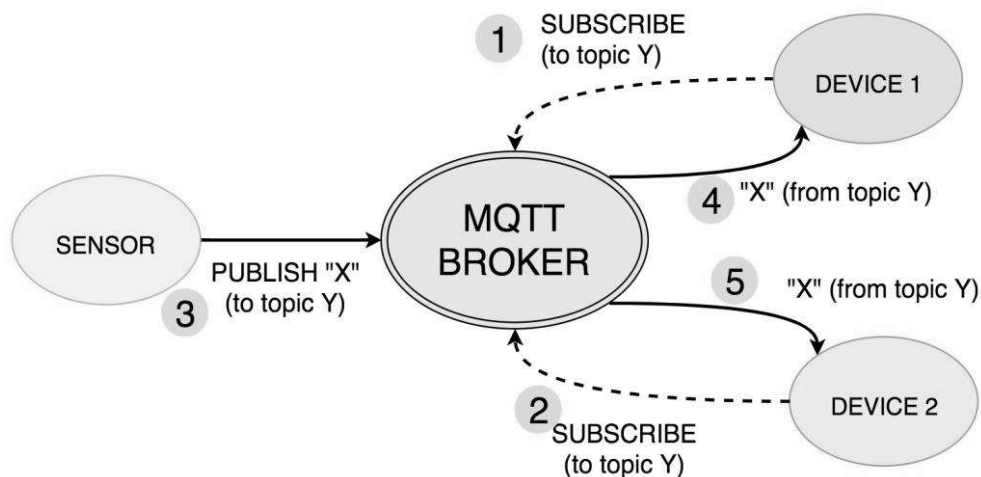
Neste cenário de IOT (*Internet Of Things*) e seu uso para o monitoramento e envio de informações à nuvem, realizou-se um projeto, que é o tema desta apresentação, mostrando o envio de dados à rede via protocolo MQTT utilizando um módulo com o Esp32, Display lcd, teclado matricial e outros componentes que serão descritos no decorrer deste trabalho.

2.0 - Protocolo MQTT e ESP32

2.1 - MQTT

No final da década de 1990, as comunicações via satélite estavam se popularizando e diversos dispositivos foram criados para se conectar a esses equipamentos. A sua implementação exigia um alto custo financeiro devido à complexidade nos algoritmos que existiam, demandando muito tempo de programação e a exigência de hardwares cada vez mais potentes. Diante deste problema, engenheiros da IBM criaram um protocolo de comunicação que possibilitou a utilização de aparelhos com capacidade de processamento mais limitado e com algoritmos de fácil implementação, diminuindo assim os custos para se utilizar essa tecnologia.

Esse protocolo foi chamado MQTT (MQ Telemetry Transport), fundamentado no modelo Publicador-Subscritor, extremamente simples e leve, justamente para o sensoriamento e comunicação remota com dispositivos mais simples, mas que consigam se conectar à rede de internet. Atualmente ele está aberto para quem quiser utilizá-lo e seu uso é voltado para sistemas IOT. Abaixo podemos ter uma ideia de como ele funciona:

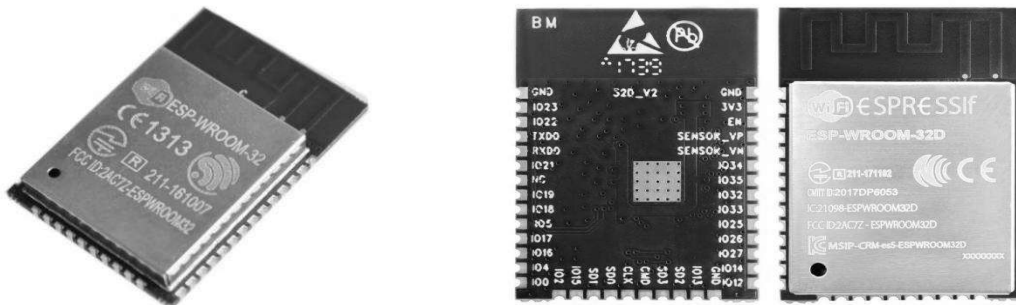


Tanto os sensores como os “Devices” podem receber e enviar informações, bastando inserir o tópico da informação e dependendo, o nome do cliente e a senha (para mais segurança). O broker funciona como intermediador e numa questão didática, é semelhante a um servidor. Ele pode se alocar tanto na nuvem como pode ser implementado em uma central como um computador ou raspberry. Existem organizações que possibilitam utilizar o broker gratuitamente na rede, como o mosquitto-eclipse, dashboard e outros que possibilitam um *Test Trial* e versões *free* mais limitadas. Isso é muito bom pois possibilita uma tecnologia onde várias pessoas podem utilizar, fazer seus testes, inovações e, com esse compartilhamento de informações, a inovação tecnológica.

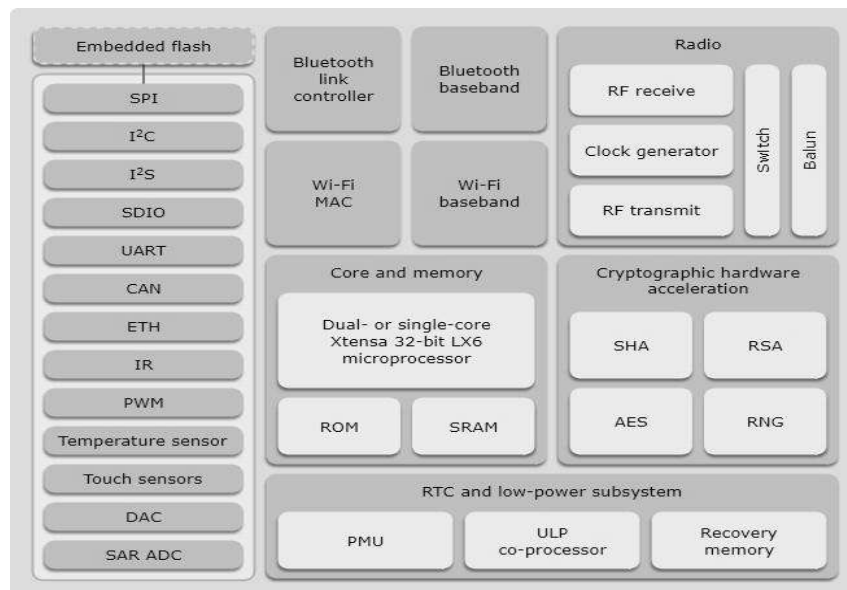
2.2 - Esp32

Existem atualmente vários hardwares que possibilitam a conexão com a internet, desde módulos que são ligados a simples placas de prototipação, como o Arduino, e placas mais sofisticadas como o Raspberry, beaglebone, kinetis, etc...

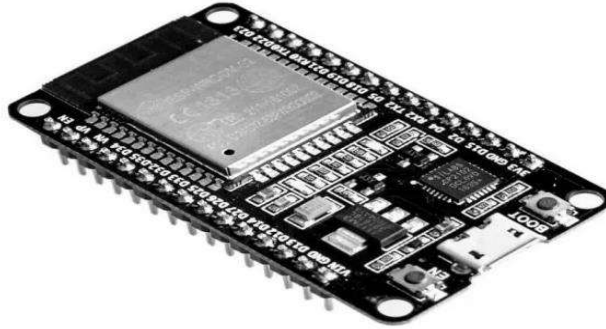
O projeto apresentado aqui é baseado no Esp32, que se trata de um chip com invólucro metálico que engloba um microcontrolador de alto desempenho e circuitos de RF para conexão à internet via WiFi e comunicação via bluetooth. Podemos vê-lo logo abaixo:



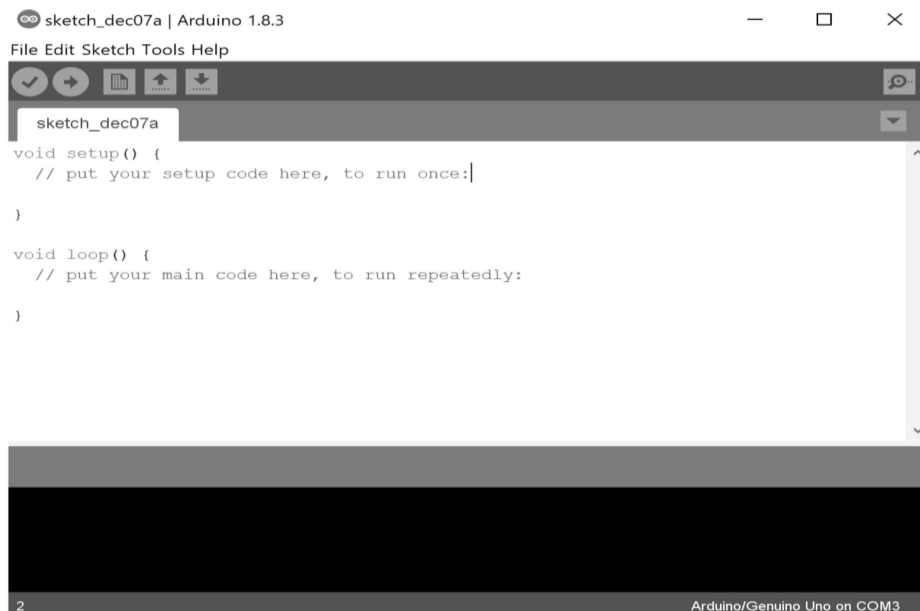
Ele possui um preço muito acessível diante da capacidade de processamento e utilidade que oferece. Seu funcionamento é 3.3 Volts e além dos circuitos de rádio frequência, há outros periféricos cujo diagrama de blocos interno pode ser visto logo abaixo:



Existe no mercado alguns módulos que possuem um hardware básico para a utilização do Esp32 na prototipação de sistemas embarcados. Um deles é o DEV-KitV1, que é alimentado em 5V, possui um circuito para comunicação USB e leds de indicação, como é possível ver na seguinte imagem:



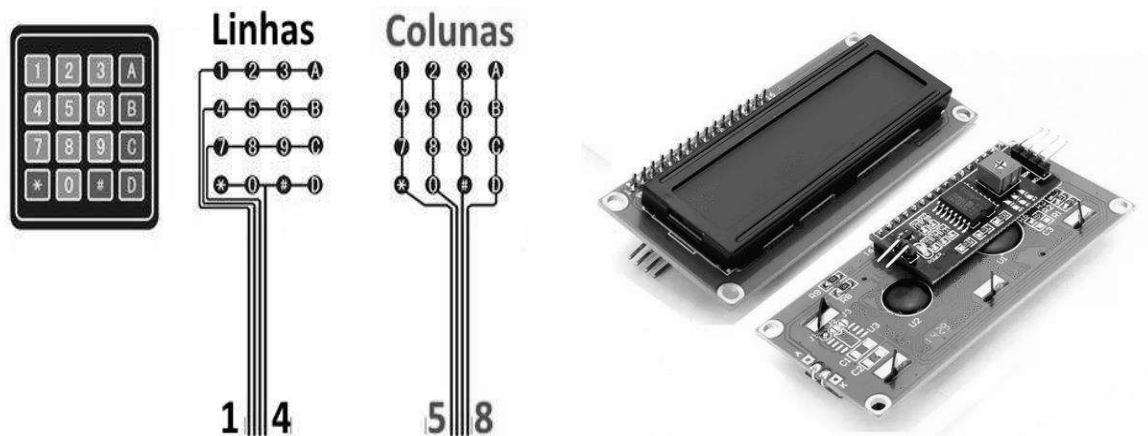
Há alguns ambientes para programá-lo, podendo usar tanto a linguagem python (essencial no dia a dia do engenheiro e programador) como a linguagem C (a mais utilizada para sistemas embarcados mais limitados). A IDE utilizada foi a do próprio Arduino, bastando apenas baixar o plug-in com as ferramentas para usar o Esp32. Praticamente todas as bibliotecas são compatíveis para as duas placas, facilitando assim a programação e um menor tempo para desenvolver a ideia do projeto. Abaixo podemos ver uma imagem desse ambiente:



O módulo do Esp32 utilizado foi o DEV-KitV1, que estava disponível para realizar o trabalho.

3.0 - Desenvolvimento do Projeto

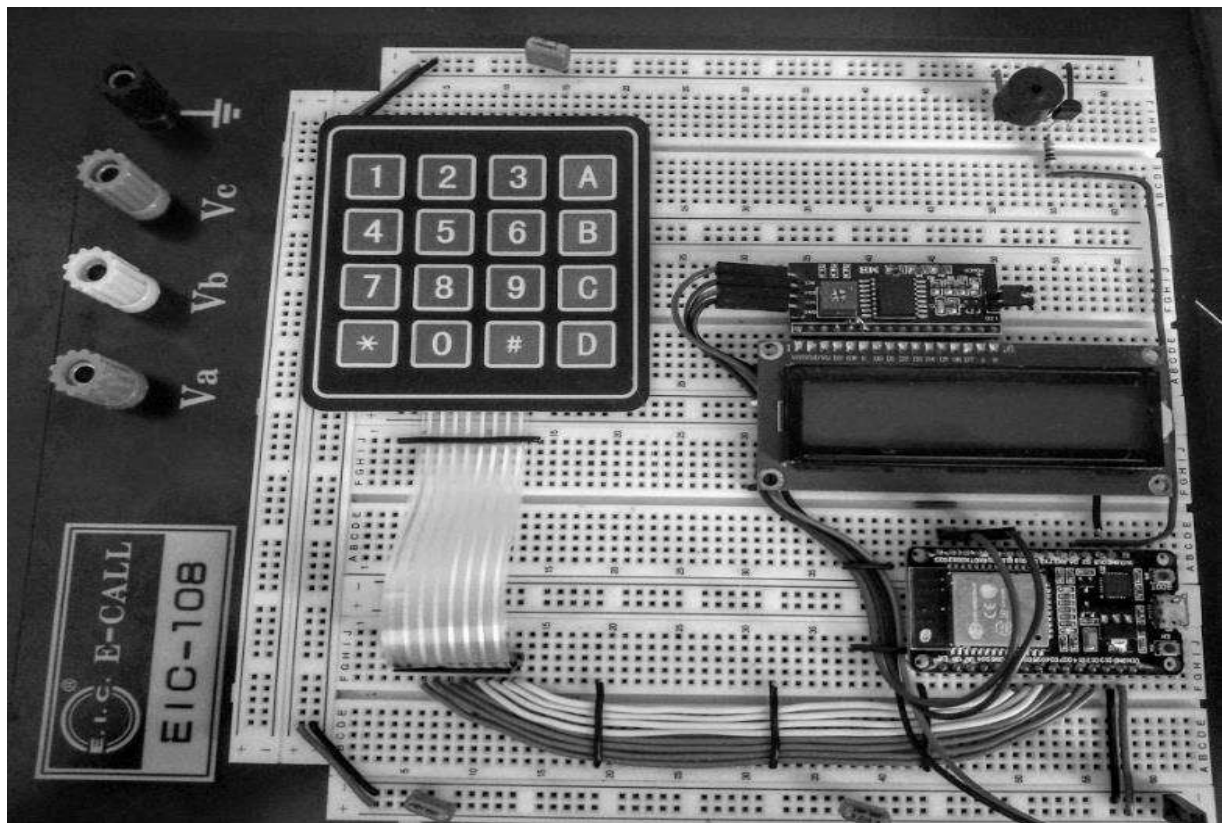
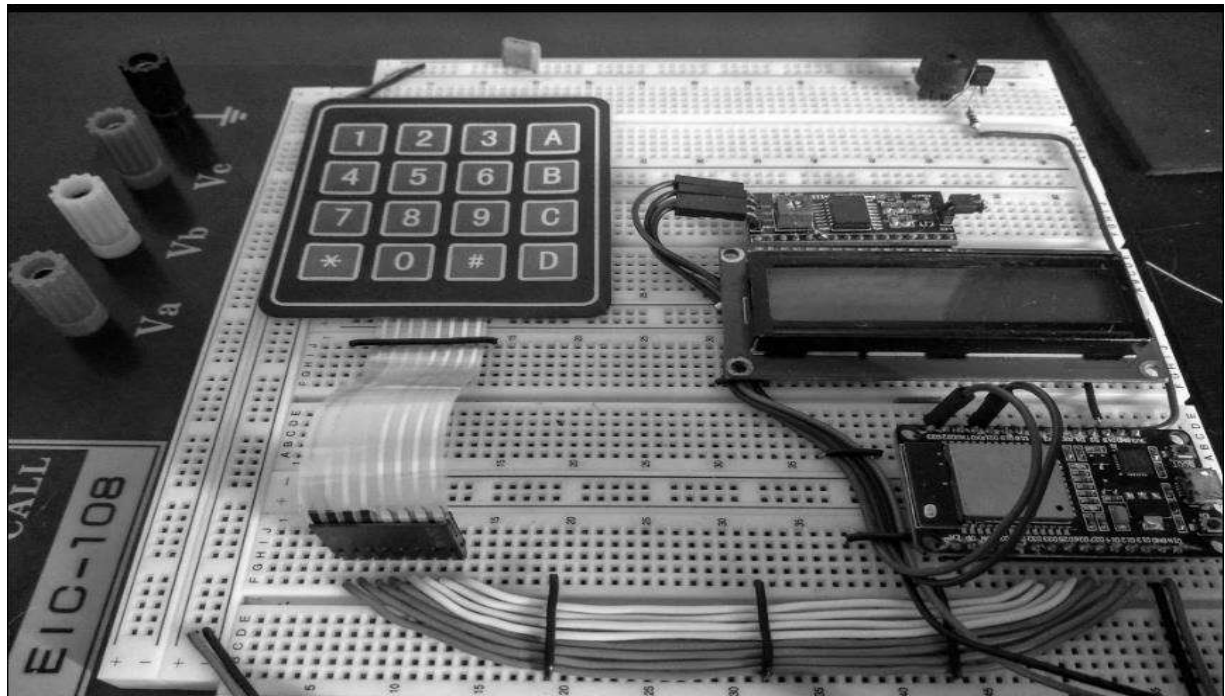
Como o desafio envolvia uma interface homem-máquina para envio de caracteres, utilizou-se um teclado matricial 4x4 tipo membrana e um display LCD alfanumérico 16x2 com comunicação I2C.



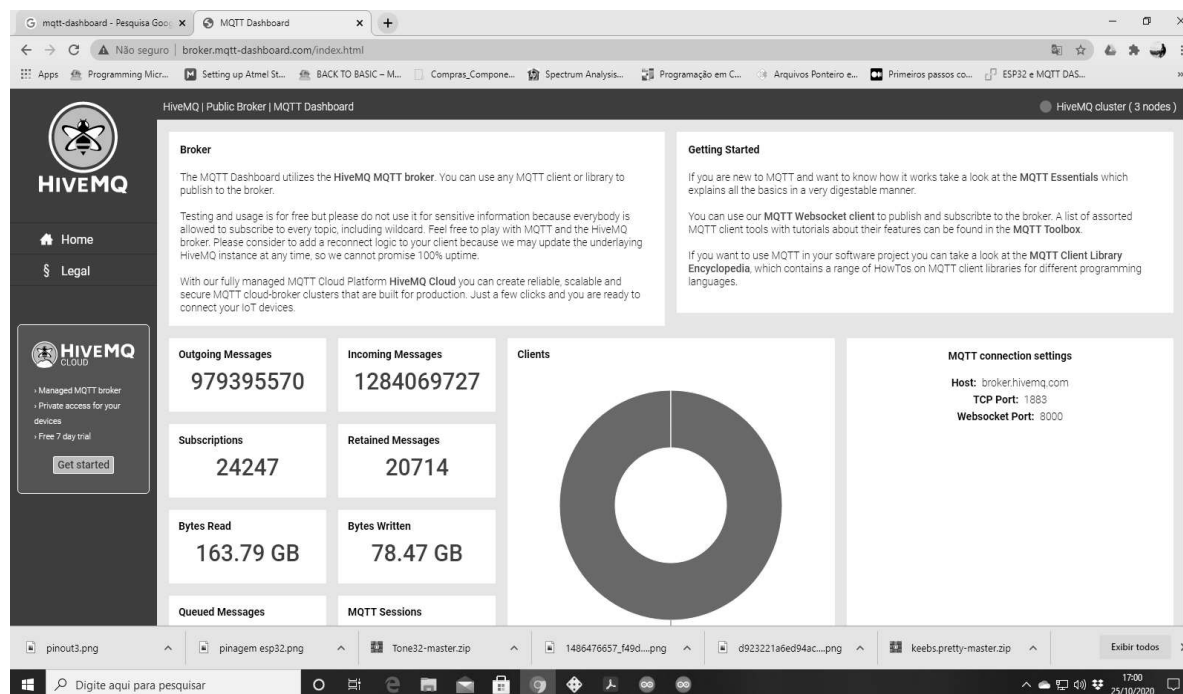
O I2C é um barramento serial criado pela Phillips para comunicação de baixa velocidade. Ele utiliza dois canais, sendo um de sincronia (SCLOCK) e um de dados (SDATA), possibilitando a conexão de vários dispositivos nessa mesma linha, bastando apenas mudar o endereço de cada um. O responsável por enviar os dados ao LCD via I2C é um circuito integrado produzido pela NXP, o PCF8574, que possui um endereçamento de 3 bits, permitindo o uso de 8 unidades no mesmo barramento.

Também foi utilizado um *buzzer* passivo com um transistor funcionando como *booster* de corrente. Sua utilização é emitir indicativos sonoros durante o funcionamento das instruções.

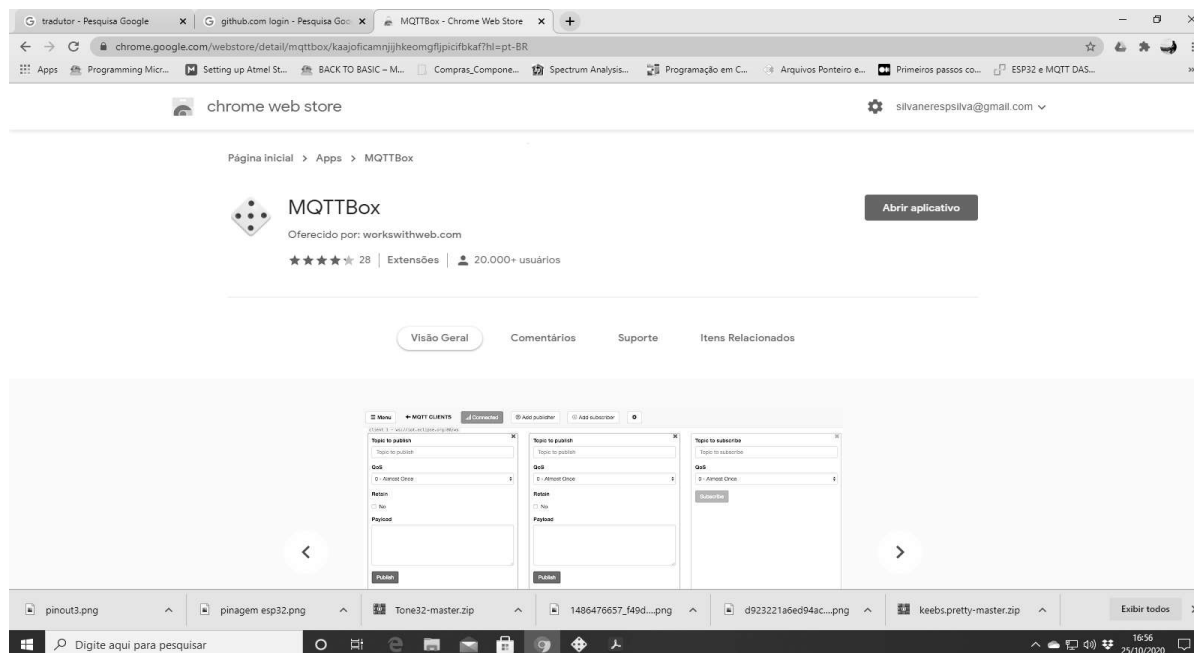
O projeto foi implementado numa protoboard e a programação se deu através da IDE do Arduino e *upload* do código via USB. Podemos ver o projeto nas imagens que se seguem:



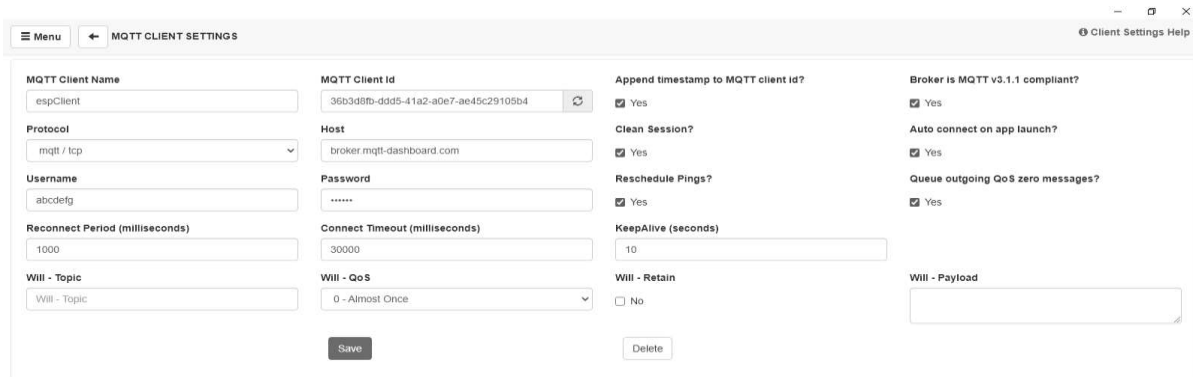
O broker utilizado foi o broker.mqtt-dashboard.com, que possui uma versão gratuita com alguns limites como a segurança, mas o suficiente para o projeto protosto. O site deles é esse:



Como subscritor, foi utilizado o MQTTBox, que é uma extensão do Google Chrome, onde é possível se conectar ao broker, fazer as configurações necessárias, escrever o tópico do dado (payload) que deseja receber e assim efetuar a comunicação, a janelas de extensão é essas:



Para as configurações de conexão ao broker, é aberta a seguinte janela:

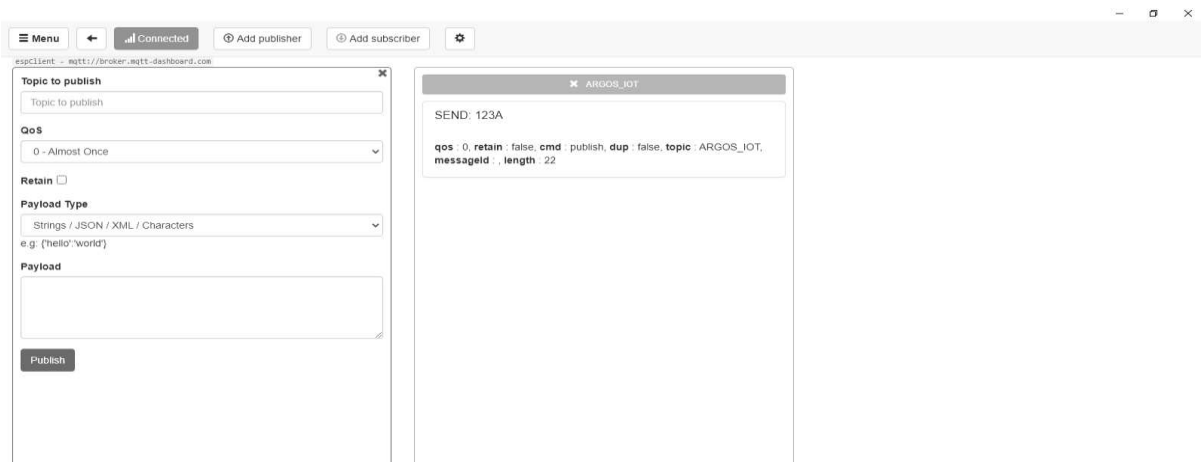


The image shows the 'MQTT CLIENT SETTINGS' window. It contains several sections for configuration:

- MQTT Client Name:** espClient
- MQTT Client Id:** 36b3d8fb-ddd5-41a2-a0e7-ae45c29105b4
- Protocol:** mqtt / tcp
- Host:** broker.mqtt-dashboard.com
- Username:** abcdefg
- Password:** *****
- Reconnect Period (milliseconds):** 1000
- Connect Timeout (milliseconds):** 30000
- Will - Topic:** Will - Topic
- Will - QoS:** 0 - Almost Once
- Append timestamp to MQTT client id?** ☒ Yes
- Clean Session?** ☒ Yes
- Reschedule Pings?** ☒ Yes
- KeepAlive (seconds):** 10
- Will - Retain:** ☐ No
- Broker is MQTT v3.1.1 compliant?** ☒ Yes
- Auto connect on app launch?** ☒ Yes
- Queue outgoing QoS zero messages?** ☒ Yes
- Will - Payload:** (empty text area)

Buttons: Save, Delete

Salvando, ele apresenta um pequeno indicador no canto superior da tela informando a conexão ao broker. Depois de configurado, podemos abrir a janela e inserir o tópico para o dado que queremos receber:



The image shows the MQTT Client window after saving settings. The status bar at the top indicates 'Connected'. The main area is divided into two panels:

- Left Panel (Publish Settings):**
 - Topic to publish:** (empty text area)
 - QoS:** 0 - Almost Once
 - Retain:** ☐
 - Payload Type:** Strings / JSON / XML / Characters
 - Payload:** (empty text area)
 - Publish:** (button)
- Right Panel (Message Log):**
 - SEND: 123A**
 - qos: 0, retain: false, cmd: publish, dup: false, topic: ARGOS_IOT, messageid: , length: 22**

Para a conexão do Esp32 ao broker, foi configurado no programa, a rede WiFi que ele será conectado, que no caso, foi um roteador residencial, e depois o nome do Server que será usado para a conexão:

```
// Declaração de variáveis =====

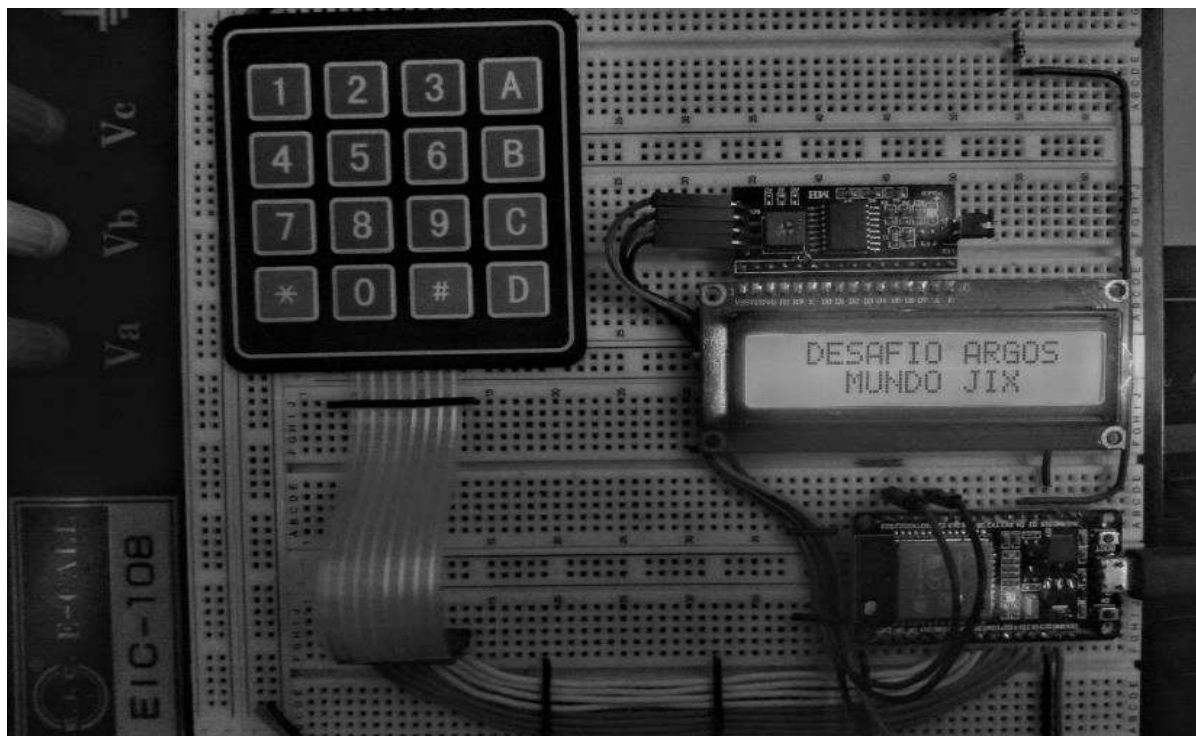
const int pin = 2;
const char* ssid = " *****"; // nome da rede WiFi que será conectado
const char* password = " *****"; // Senha da rede Wifi
const char* mqttServer = "broker.mqtt-dashboard.com"; // Nome para o servidor Broker
const int mqttPort = 1883; // porta
const char* mqttUser = "abdefg"; // Usuário do cliente
const char* mqttPassword = "123456"; // Senha de usuário do cliente
char mensagem[16];
```

O array “mensagem” é o conteúdo do payload, onde é alocado os caracteres digitados pelo teclado matricial.

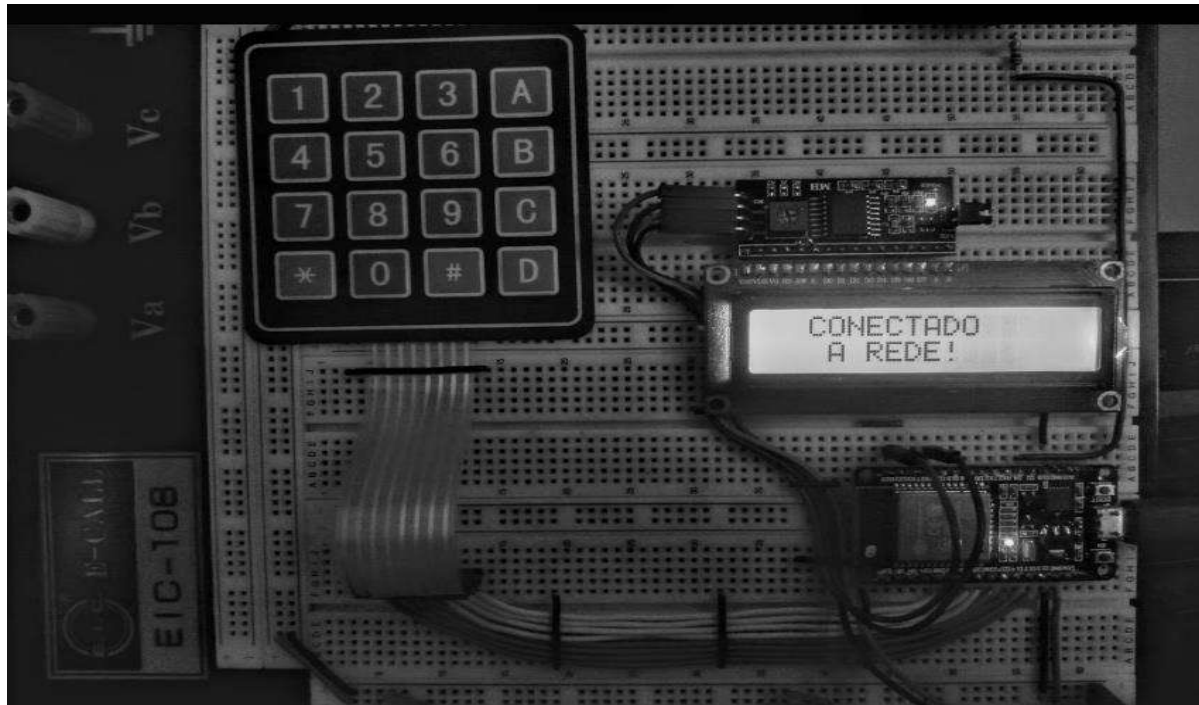
A partir disso o código escrito foi gravado, resolvido alguns debugs e então realizado os testes para verificar se o sistema estava funcionando.

4.0 - Funcionamento do Projeto

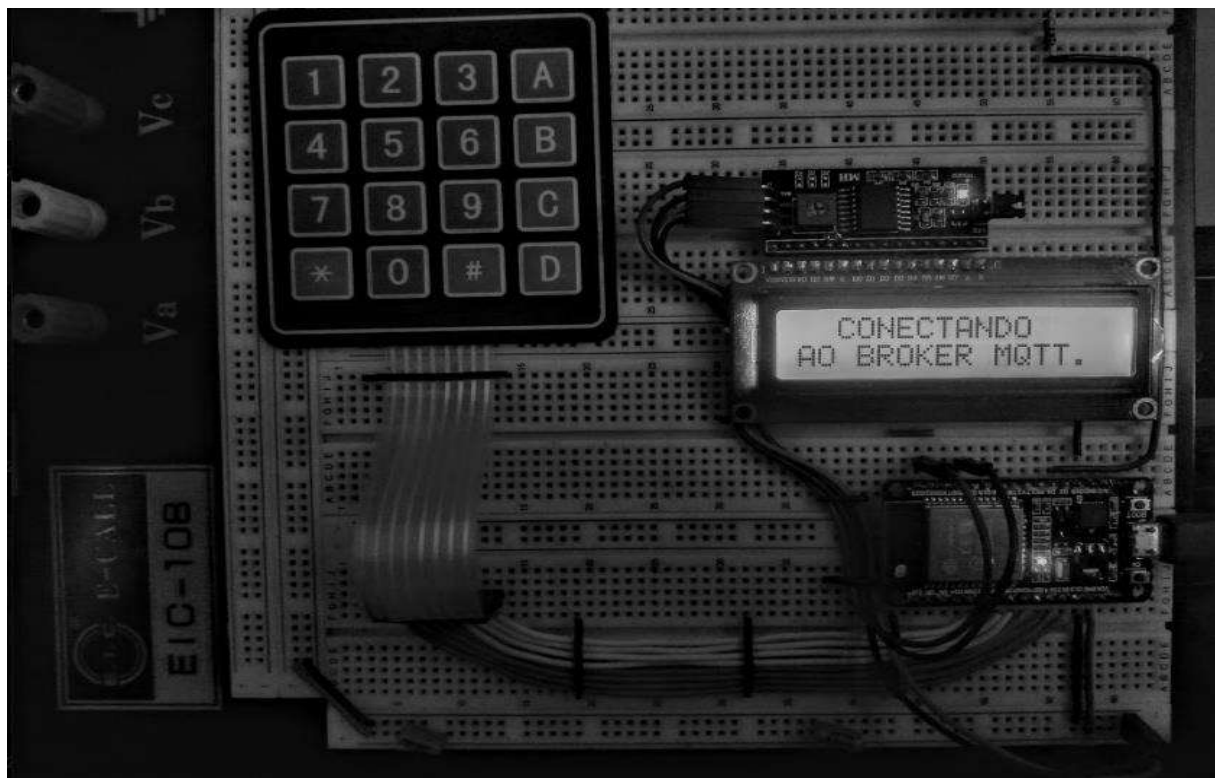
O sistema se inicia informando as empresas responsáveis pelo desafio apresentado:



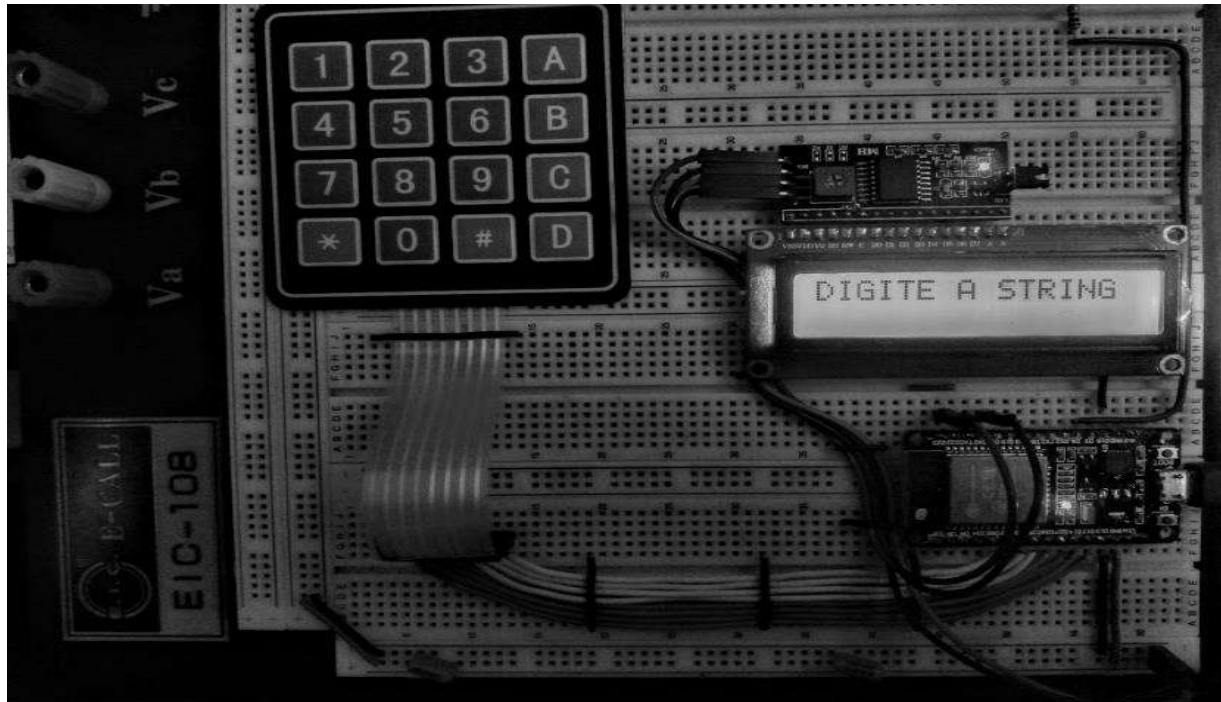
Depois ele dá alguns bips e tenta se conectar à rede WiFi indicando a mensagem “CONECTANDO” e depois de conectado, envia a mensagem:



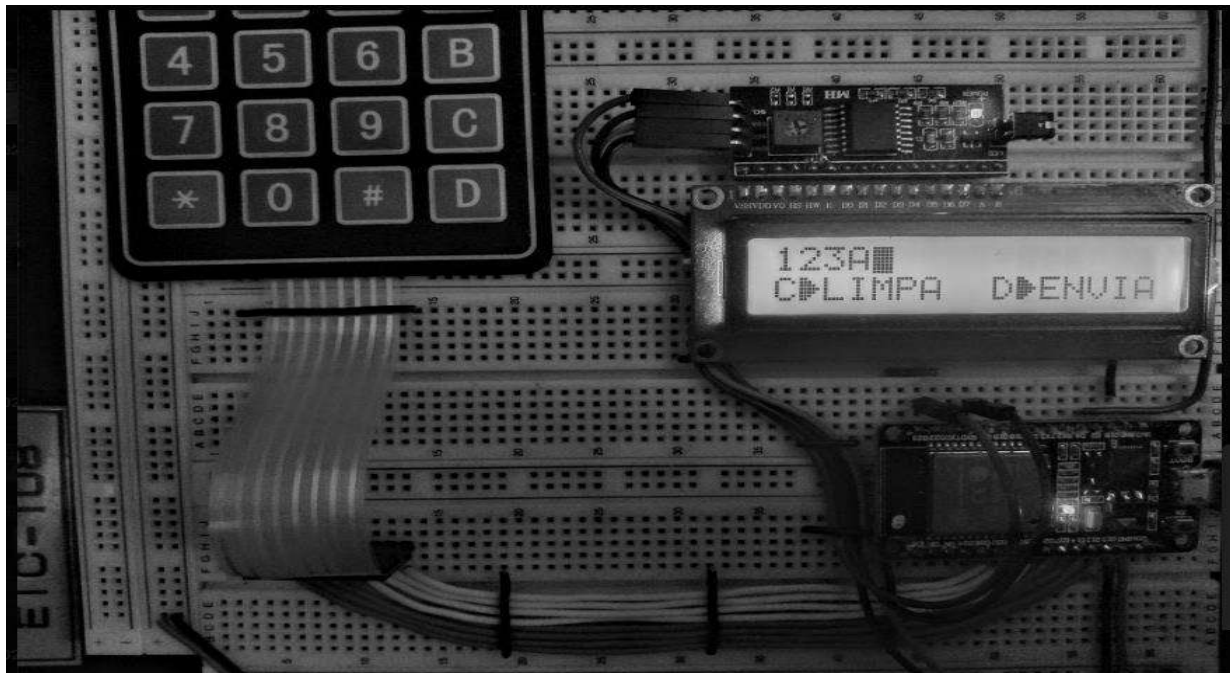
Depois de conectado, ele emite uns bips e tenta se conectar ao broker indicando outra mensagem:



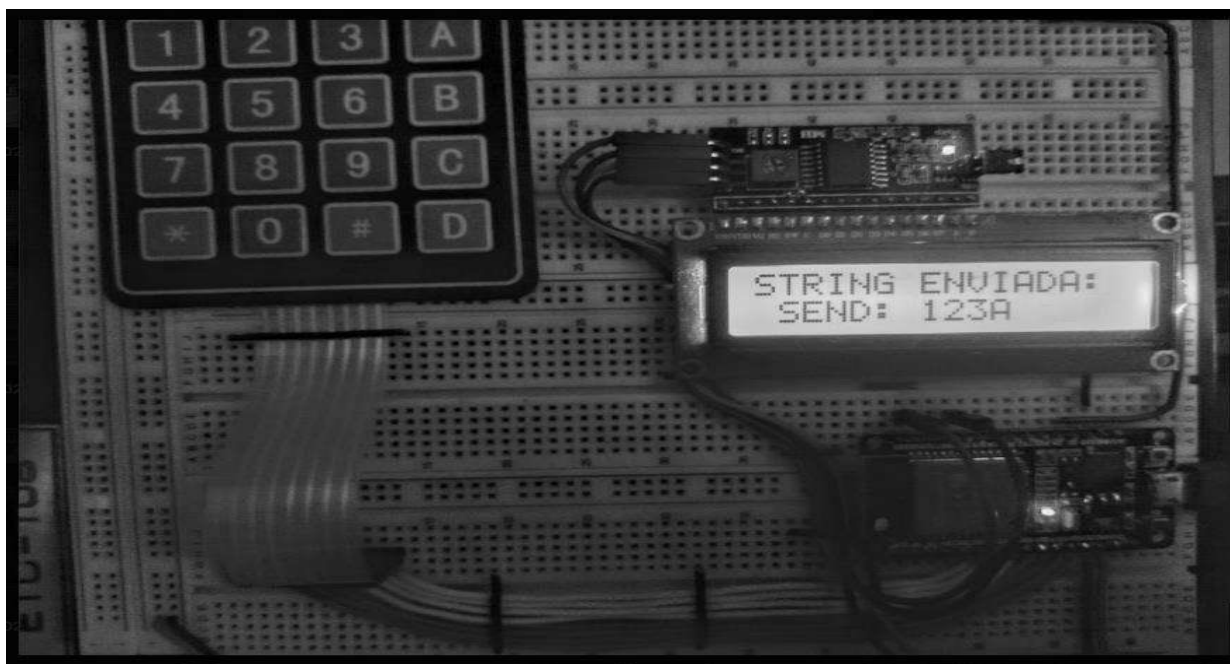
Conectado então ao broker, é emitido outro Bip e então solicita ao usuário para digitar uma string (caracteres disponíveis no teclado utilizado, exceto o 'C' e o 'D'), dá uma pequena pausa para visualização, soa outro bip e então libera espaço para digitação pelo usuário com as opções de 'C' para limpar, caso tenha errado a string e 'D' para enviar ao broker:



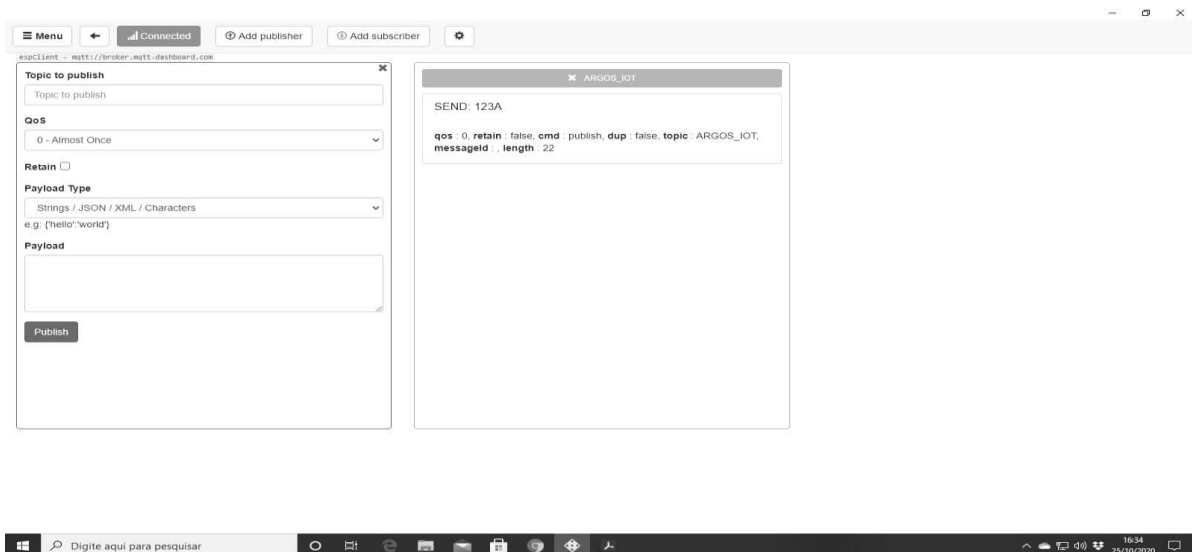
Digitando então, através do teclado matricial a string “123A”:



Apertando a tecla ‘C’, a string digitada é apagada e é possível escrever uma nova. Como o conteúdo escrito estava correto para o envio, foi pressionado o ‘D’. A partir disso, há um processamento interno e então é enviado a mensagem ao broker junto com o termo “SEND: “:

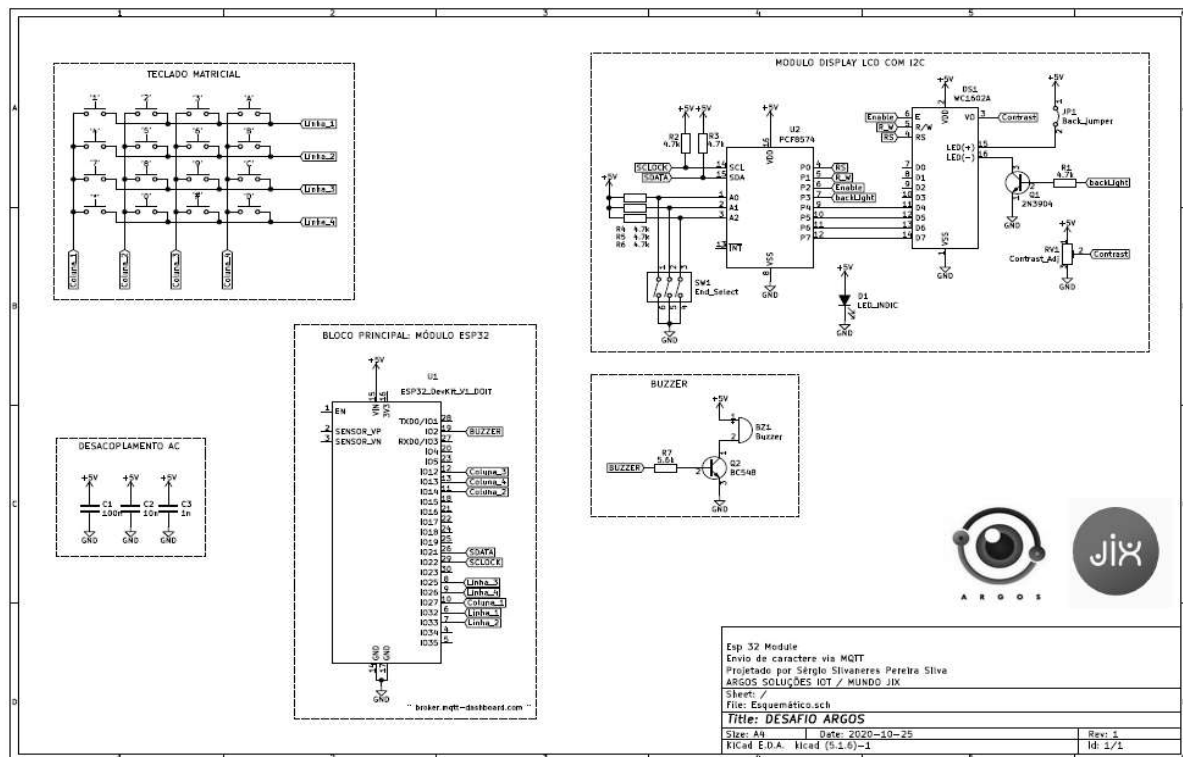


Como a extensão já estava conectada ao broker com o respectivo tópicos da string enviada, que é “ARGOS_IOT”, podemos ver a mensagem recebida no MQTTBox:



Como podemos ver, o recebimento foi feito mostrando o payload completo “SEND: 123A”, provando assim o sucesso do envio.

O esquemático do projeto do desafio é apresentado logo abaixo e também será indexado como PDF aos arquivos da ARGOS e no GitHub para melhor visualização:



5.0 - Conclusão

Vemos que há inúmeras aplicações de IOT em todas as áreas da sociedade, trazendo cada vez mais conforto, inclusão entre as pessoas e mais praticidade no nosso cotidiano. Podemos até afirmar que o limite da inovação está na capacidade de elaboração de novos projetos entre os grupos que desenvolvem estas soluções, diante da grande disponibilidade de plataformas que existem e que será cada vez mais presente, principalmente com a tecnologia 5G, onde haverá a ampliação de banda e vários outros recursos novos, integrando ainda mais os dispositivos que se conectam à rede.

6.0 - Referências Bibliográficas

- <https://www.espressif.com/en/products/socs/esp32>
- <https://www.filipeflop.com/blog/esp32-e-mqtt-dashboard-android/>
- <https://medium.com/@flaviofagundes/primeiros-passos-esp32-e-broker-mqtt-b8b3e41297c>
- https://www.nxp.com/docs/en/data-sheet/PCF8574_PCF8574A.pdf
- <https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjjhkeomgflpicifbkaf?hl=pt-BR>
- <https://www.newtoncbraga.com.br/index.php/microcontrolador/143-tecnologia/17117-comunicando-se-via-mqtt-com-o-esp32-mic404>

7.0 - Link do Repositório GitHub com o projeto:

- https://github.com/silvaneres/Argos_challenge_JIX.git