UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática

Medidor de temperatura e umidade WemosHumidTemp

Matheus Sena da Graça[1], Willian França Costa[2]

[1]Universidade Presbiteriana Mackenzie (UPM)
Rua da Consolação, 930 Consolação, São Paulo - SP, 01302-907 – Brazil

31784781@mackzenzista.com.br

Abstract. *This article describes the implementation of a high precision temperature and humidity meter, which, when collecting the data, presents it on an LCD display, and using the MQTT communication protocol, sends the same data to an API on internet where it graphically represents this information. The idea is based on the principle that a temperature and humidity control in some environment is of fundamental importance, whether it is used in industry, to gauge the conditions in which the products or the environment in which it is found, in addition to other means*

Resumo. *O artigo a seguir descreve a implantação de um medidor de temperatura e umidade de alta precisão, no qual, ao coletar os dados, apresenta-os em um display LCD, e utilizando o protocolo de comunicação MQTT, envia os mesmos dados para uma API na internet onde representa graficamente essas informações. A ideia se parte do princípio de que, um controle de temperatura e umidade em algum ambiente é de fundamental importância, seja ele usado na indústria, para aferir em quais condições estão os produtos ou o ambiente em que ele se encontra, além de outros meios.*

Introdução

O projeto é baseado em um circuito onde são utilizados componentes como a placa Wemos D1, um sensor de temperatura e um Display LCD, no qual ao captar temperatura, apresenta o valor medido no Display, e envia os dados para a Internet utilizando o protocolo MQTT. Os sensores de temperatura e umidade são equipamentos capazes de mensurar a umidade relativa do ar e a temperatura de um ambiente, produto ou equipamento. Eles podem ser utilizados tanto ao ar livre como também em ambientes fechados. Trata-se de um instrumento muito utilizado em farmácias, laboratórios,

almoxarifados e hemocentros, entre outros ambientes. O equipamento pode ser portátil e possui um tempo de resposta bastante baixo, em apenas alguns segundos, já é capaz de indicar a temperatura relativa do ar, além de ter uma poderosa aliada da tecnologia, chamada Internet das coisas (IoT) nas quais facilitam a vida e a manipulação de dados entre dispositivos, que podem ser importantes na geração de relatórios para organizações.

Materiais e Métodos

Para a montagem do circuito, utilizamos uma placa de ensaio para o auxílio na ligação dos componentes, além de um sensor de temperatura e umidade de alta precisão, um display LCD, além de uma plataforma Wemos D1 que já possui um módulo WiFi acoplado. Agora vamos explicar, em maiores detalhes sobre os componentes utilizados, os softwares utilizados, além de como foi feita a montagem do circuito para esse projeto.

Plataforma Wemos D1

A plataforma Wemos D1 lembra um pouco a estrutura de um Arduino Uno, conforme apresentado na figura 1. Porém, existem diferenças importantes entre as duas placas. A Wemos D1 basicamente possui as mesmas dimensões do Arduino Uno R3, as diferenças mais básicas em relação ao Arduíno são o conector micro USB e um ESP8266-12E já acoplado na placa. A Wemos D1 já conta com WiFi nativo, o que a torna uma plataforma extremamente atraente para desenvolvimento de projetos IoT. Atualmente a placa se encontra na versão R2.

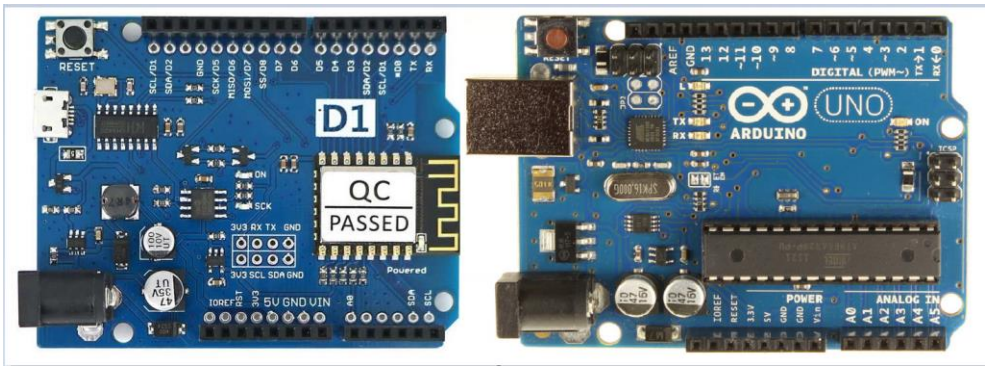

Figura 1. Do lado esquerdo, temos a placa Wemos D1, e do lado direito, temos a placa Arduino Uno R3

Abaixo temos algumas características importantes dessa placa:

- Processador ESP8266-12E
- Arquitetura RISC de 32 bits
- Processador pode operar em 80MHz / 160MHz
- 4Mb de memória flash
- 64Kb para instruções
- 96Kb para dados
- WiFi nativo padrão 802.11b/g/n
- Opera em modo AP, Station ou AP + Station
- Pode ser alimentada com 5VDC através do conector micro USB
- Através do pino jack pode ser alimentada com tensão na faixa de 9 a 24VDC
- Possui 11 pinos digitais
- Possui 1 pino analógico com resolução de 10 bits
- Pinos digitais, exceto o D0 possui interrupção, PWM, I2C e one wire
- Pinos operam em nível lógico de 3.3V
- Pinos não tolerantes a 5V
- Conversor USB Serial CH340G
- Programável via USB ou WiFi (OTA)
- Compatível com a IDE do Arduino
- Compatível com módulos e sensores utilizados no Arduino
- Compatível com alguns Shields da linha Arduino

Na figura 2 a seguir, temos uma breve descrição da composição da placa:

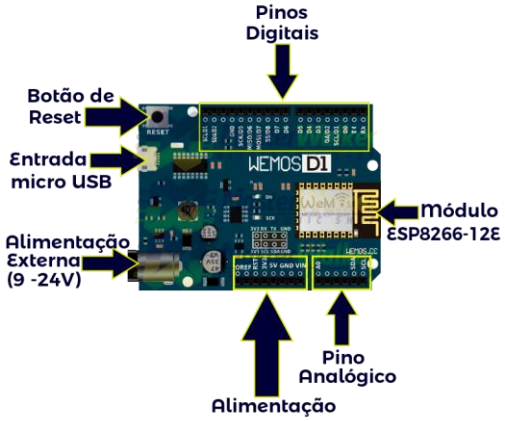

Figura 2. Composição da placa Wemos D1

Vale ressaltar também, que a Wemos D1 foi construída contendo um divisor de tensão interno para garantir que a leitura no pino ADC (conversor analógico digital) seja entre 0 e 1V.

Sensor de Temperatura e Umidade HDC1080 Alta Precisão

O Sensor de Temperatura e Umidade de alta precisão, sensor HDC1080 utiliza a comunicação I2C com o microcontrolador, para utilização em projetos eletrônicos, estações meteorológicas, controle de temperatura ambiente e equipamentos médicos, entre outros. O sensor HDC1080 possui 14 bits de resolução e precisão de umidade de 2% e de temperatura de 0.2°C, fornecendo informações precisas e confiáveis além de um baixíssimo consumo de energia em modo sleep. O sensor possui 14 bits de resolução e precisão de umidade de 2% e de temperatura de 0.2°C, fornecendo informações precisas e confiáveis além de um baixíssimo consumo de energia em modo sleep. Abaixo temos algumas de suas especificações:

- Tensão de operação: 2.7 a 5.5VDC
- Faixa de operação: -40 a 125°C
- Precisão umidade: ±2%
- Precisão temperatura: ±0.2°C
- Resolução de medição de 14 bits
- Interface de comunicação I2C
- Consumo de 100 nA em modo sleep
- Calibração de fábrica
- Dimensões: 20 x 11 x 2 mm

Figura 3. Sensor de temperatura e umidade de alta precisão HDC1080

Display LCD

O Display LCD 16x2 é um modelo de display vastamente utilizado em projetos onde se necessita uma interface homem-máquina. Ele é composto por 16 colunas e 2 linhas com a escrita na cor branca e sua backlight (luz de fundo) azul para exibição de caracteres, letras e números de forma clara e nítida, melhorando a visibilidade para quem recebe a informação. São 16 colunas por 2 linhas, backlight azul e escrita branca. O Display LCD 16x2 utiliza o controlador HD44780, utilizado em toda indústria de LCD's como base de interface que pode ser operado em 4 ou 8-bits paralelamente. Sua conexão é feita através de 16 pinos, sendo 12 deles para conexão básica com o microcontrolador e 11 deles pinos de entrada/saída (I/O) e os demais pinos para ajuste de contraste através de potenciômetros, trimpots e afins e para a alimentação da backlight. Também é possível fazer a comunicação I2C com um microcontrolador. Fácil interação com qualquer microcontrolador, como Arduino, Raspberry, Pic, entre outros. Aqui estão algumas de suas especificações:

- Cor backlight: Azul
- Cor escrita: Branca
- Dimensão Total: 80mm X 36mm X 12mm
- Dimensão Área visível: 64,5mm X 14mm
- Dimensão Caractere: 3mm X 5,02mm
- Dimensão Ponto: 0,52mm X 0,54mm

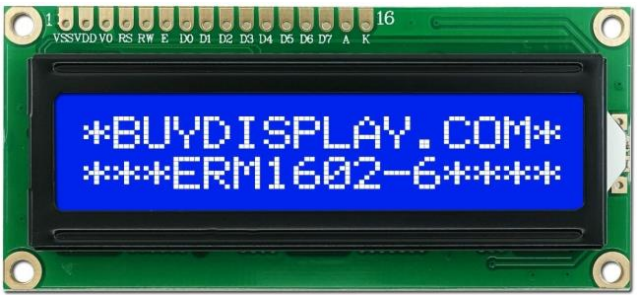

Figura 4. Display LCD 16x2

Montagem do circuito

Na montagem do circuito utilizamos uma protoboard e 8 jumpers para auxiliar nas ligações entre os componentes. Ligamos a entrada SCL e SDA da placa Wemos D1 nos pinos SCL e SDA respectivamente tanto no display LCD como também no Sensor de temperatura e umidade. O mesmo foi feito com o pino 5VV e GND, ligando o sensor e o Display através da protoboard, conforme exemplificado nas figuras abaixo

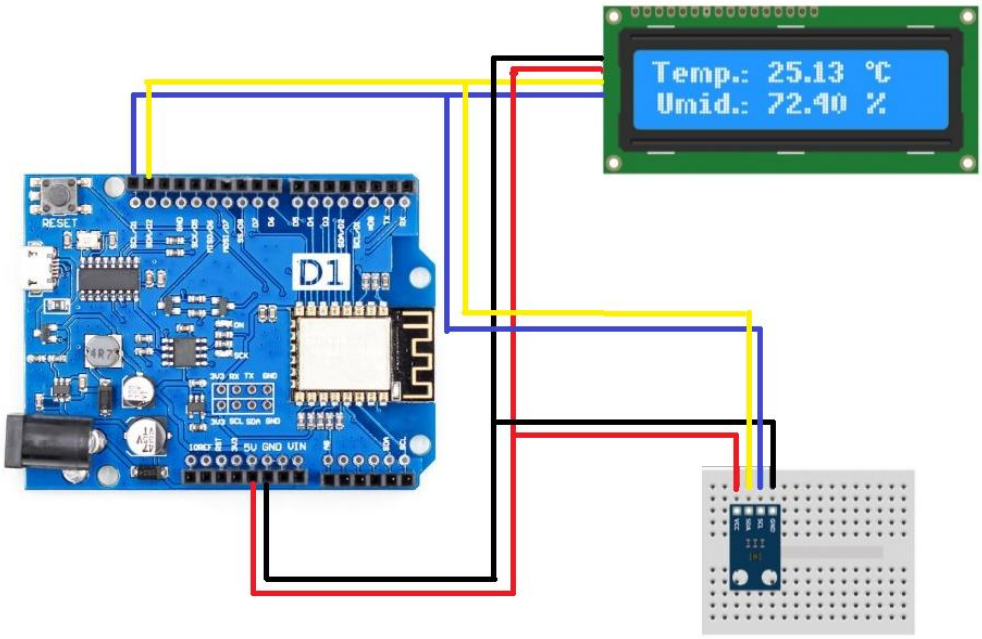

Figura 5. Esquematização do circuito

Primeiramente, ligamos a placa com o sensor de acordo com a figura 6 abaixo, através dos pinos indicados pelo próprio sensor

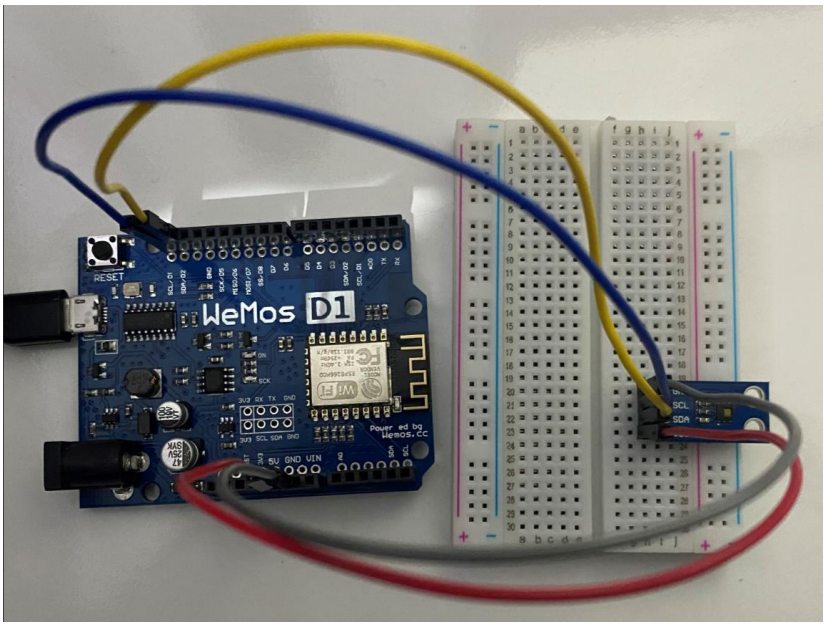

Figura 6. Ligação do Sensor com a placa Wemos D1

E após isso, ligamos o display LCD no mesmo barramento das ligações do sensor na protoboard, conforme figura 7, como se fosse um compartilhamento da Ligação com a placa.

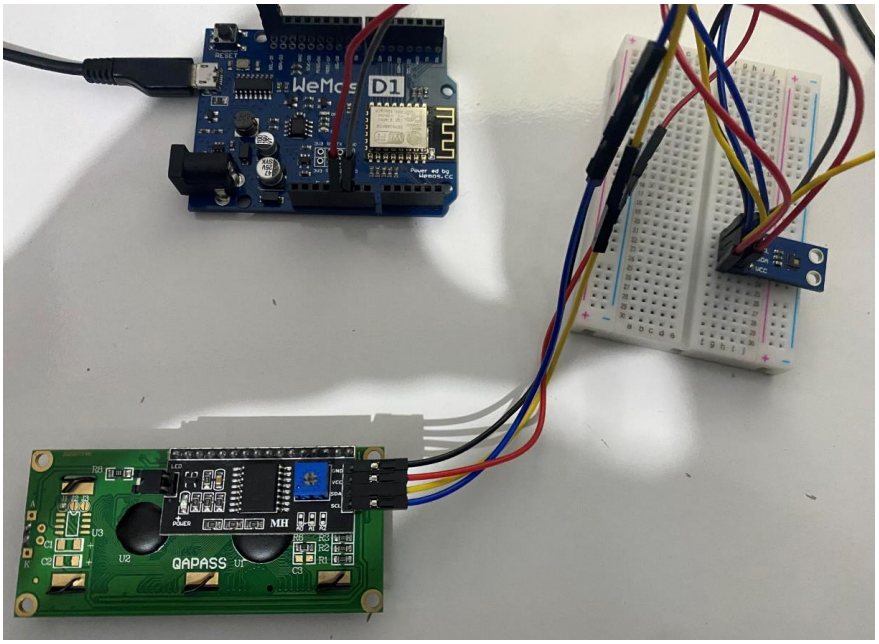

Figura 7. Ligando o Display LCD ao circuito

Desenvolvimento do Código

Para desenvolver o Código de nossa aplicação, usamos a IDE Arduino. Ela pode ser baixada no site oficial do Arduino, de acordo com o seu Sistema operacional, seja ele Windows, Mac OS ou Linux. Uma grande vantagem dessa IDE, é a possibilidade de usar muitos módulos do Arduino, assim como a facilidade na instalação de bibliotecas que axiliam no funcionamento do nosso projeto

Figura 8. Banner para download da IDE do Arduino

Além da IDE, utilizamos a biblioteca Wire para a comunicação I2C com a Wemos D1, a biblioteca ClosedCube_HDC1080, para aferir junto com o sensor, os dados de temperatura e umidade ambiente, a biblioteca LiquidCrystal_I2C para a utilização do Display LCD, e claro, a biblioteca ESP8266WiFi para poder estabelecer uma conexão com a rede. Para instalar as bibliotecas, basta realizar o download do arquivo ZIP no repositório Git , e após isso, adicionar no gerenciador de bibliotecas da IDE, conforme figuras 9 e 10, abaixo

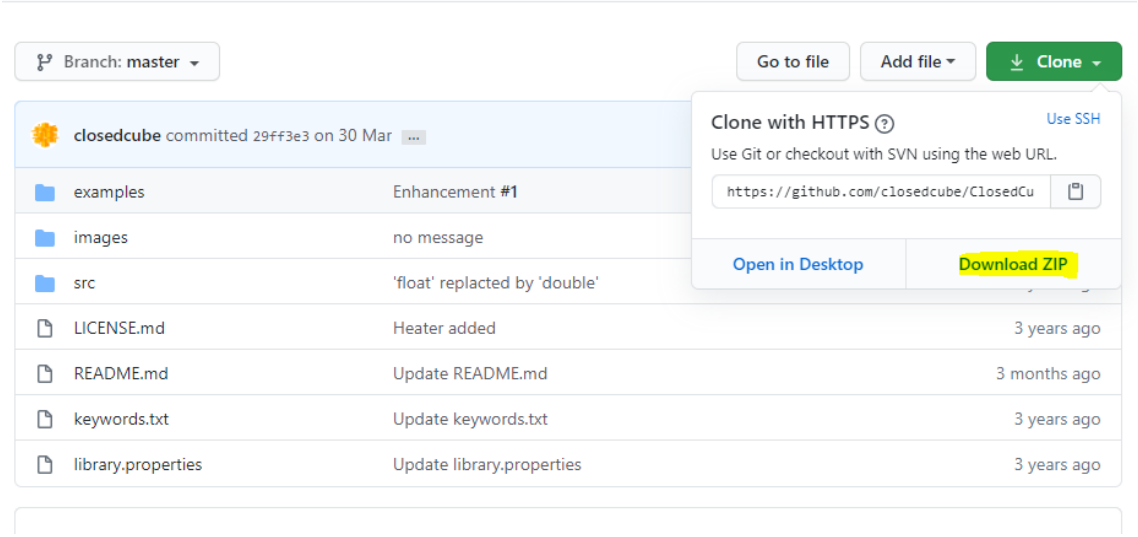

Figura 9. GitHub da biblioteca ClosedCube_HDC1080

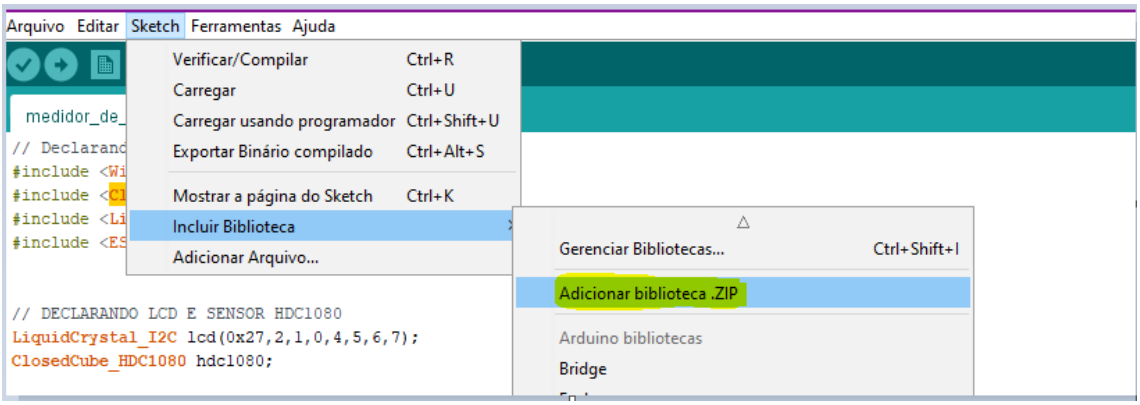

Figura 10. Como adicionar bibliotecas na IDE do Arduino

Outra opção também é o download pelo próprio gerenciador de tarefas. Basta abrindo o menu: Sketch > Incluir Biblioteca(Include Library) > Gerenciar Bibliotecas... Nele você tem disponível algumas bibliotecas que simplificará sua busca por outras bibliotecas. Como exemplo confira a figura 11 abaixo

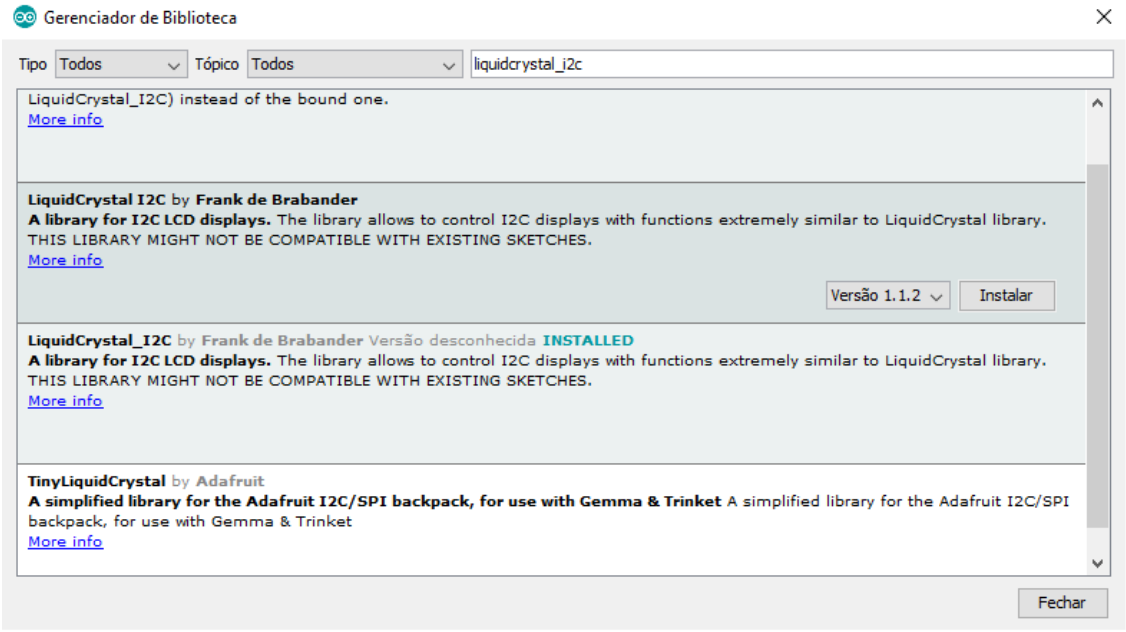

Figura 11. Gerenciador de bibliotecas da Arduino IDE

Após adicionar as bibliotecas necessárias, basta apenas declarar-las no começo do seu código, conforme exemplo na figura, e após isso, você está apto para poder utilizar-las em seu script. Com isso, podemos inicializar os componentes de nosso cricuito através de variavéis conforme a figura 12 nos mostra, além de definir também os dados de conexão com sua rede wifi e com a API na qual você deseja utilizar, no nosso caso, utilizamos a API Thingspeak.

```
//Programa: Sensor de temperatura e umidade HDC1080 com Arduino
//Autor: Matheus Sena

// Declarando as bibliotecas
#include <Wire.h>;
#include <ClosedCube_HDC1080.h>;
#include <LiquidCrystal_I2C.h>;
#include <ESP8266WiFi.h>;


// DECLARANDO LCD E SENSOR HDC1080
LiquidCrystal_I2C lcd(0x27,2,1,0,4,5,6,7);
ClosedCube_HDC1080 hdc1080;

//CONECTANDO COM A API
String apiKey = "AACZG356LWB3HDKU";//API DISPONIBILIZADA PELA API
const char* ssid = "Casa 2.4g"; //SSID DA REDE WIFI
const char* password = "guilherme"; //SENHA DA REDE WIFI
int duration=5;//TEMPO DE ENVIO DE DADOS PARA API
const char* server = "api.thingspeak.com";
WiFiClient client; //START CLIENTE
```

Figura 12. Trecho inicial do código

No Setup, inicializamos o Display LCD, assim como o Sensor, abrimos a conexão Wifi, e setamos uma formatação de dados para o Display, conforme apresentado na figura

```
void setup(){
  Serial.begin(115200);

  //INICIALIZANDO O SENSOR HDC1080
  hdc1080.begin(0x40);
  // INICIALIZANDO O MÓDULO WIFI
  WiFi.begin(ssid, password);
  Serial.println("Sensor HDC1080 com Arduino");
  //INICIALIZA O DISPLAY LCD
  lcd.begin(16,2);
  lcd.setBacklightPin(3, POSITIVE);
  lcd.setBacklight(HIGH);
  lcd.home();

  //APRESENTA AS INFORMAÇÕES INICIAIS NO DISPLAY E A FORMATAÇÃO PADRÃO DA APRESENTAÇÃO DAS INFORMAÇÕES
  lcd.setCursor(0, 0);
  lcd.print("Temp.: XX.XX C");
  lcd.setCursor(0, 1);
  lcd.print("Umid.: YY.YY %");

  Serial.println();
  Serial.println();
  Serial.print("Connecting to ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);

  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("");
  Serial.println("WiFi connected");
}
```

Figura 13. Setup do código da aplicação

Na parte do Loop, incluimos o que queremos que a aplicação em si faça. Nesse caso, começamos aferindo a temperature e a umidade ambiente junto com o sensor. Após isso, fazemos uma validação para verificar se a leitura ocorre de forma correta, caso, não ocorra, será apresentada uma mensagem no Monitor Serial da aplicação, ou se a leitura for feita com sucesso, conectamos com a API pela porta 80, concatenamos os dados medidos em uma String, e esses dados serão enviados a cada medição feita para os campos determinados na nossa API (No caso, utilizamos dois campos, um para temperatura e outro para a umidade).

```
void loop(){
  //LEITURA DE VALORES DE TEMPERATURA E UMIDADE PELO SENSOR
  float t = hdc1080.readTemperature();
  float h = hdc1080.readHumidity();

  // VERIFICA SE O SENSOR ESTÁ REALIZANDO A LEITURA, SE NÃO, APRESENTA MENSAGEM DE ERRO O MONITOR SERIAL
  if (isnan(h) || isnan(t)) {
    Serial.println("FALHA NA LEITURA DE TEMPERATURA E UMIDADE!");
    delay(1000);
    return;
  }

  // SE CONSEGUIU REALIZAR A LEITURA, CONCATENA OS DADOS MEDIDOS EM UMA STRING PARA ENVIO PARA A API
  if (client.connect(server, 80)) {
    String postStr = apiKey;
    postStr += "&field1=";
    postStr += String(t);
    postStr += "&field2=";
    postStr += String(h);
    postStr += "\r\n\r\n";
  }

  client.stop();

  // A API DEMORA CERCA DE 15 SEGUNDOS PARA RECEBER OS DADOS
  delay(duration*1000);
  //APRESENTA OS VALORES NO LED
  lcd.setCursor(7, 0);
  lcd.print(t);
  lcd.setCursor(7, 1);
  lcd.print(h);

}
```

Figura 14. Loop do código da aplicação

Para mais detalhes sobre o circuito, e o Código, assim como os arquivos ZIPs das bibliotecas utilizadas nesse projeto, estão no nosso repositório feito no GitHub.

O protocolo MQTT

O protocolo MQTT, ou Message Queuing Telemetry Transport (MQTT) é um protocolo de mensagens destinado a sensores e outros dispositivos. Seu principal uso é fazer as máquinas trocarem informações entre elas, modalidade de comunicação conhecida como Machine-to-Machine (M2M, traduzindo para o português, de máquina para máquina). Essa tecnologia foi desenvolvida pela IBM no final dos anos 90, e sua finalidade original era conectar sensores de satélites ou pipelines de petróleo. Apesar de ter criado há um tempo, sua aplicabilidade ainda é excepcionalmente útil na atualidade, inclusive em diversos ramos empresariais.

A comunicação entre aparelhos é assíncrona, isso significa que os dados podem ser transmitidos com intervalos em um fluxo estável. Isso ocorre porque ele utiliza um paradigma de publishers (publicadores) e subscribers (assinantes) baseado em TCP/IP, cliente e broker.

Seu funcionamento não é tão complicado quanto parece: o publicador envia a mensagem ao broker, que enfileira e dispara as informações recebidas aos assinantes

(que podem ser múltiplos aparelhos). Esses últimos recebem as mensagens que possuem interesse. O TCP/IP citado é uma forma de identificação entre os dispositivos.

No nosso projeto, estamos adotando o paradigma publisher, isso, porque coletamos de dados para que os mesmos possam ser enviados para um broker publico. O brokler que utilizamos é a API Thingspeak, onde com ela, conseguimos demonstrar as medições de forma gráfica em um determinado período de tempo. Abaixo, temos um esquema funcional de como funciona o protocolo MQTT em nosso projeto.

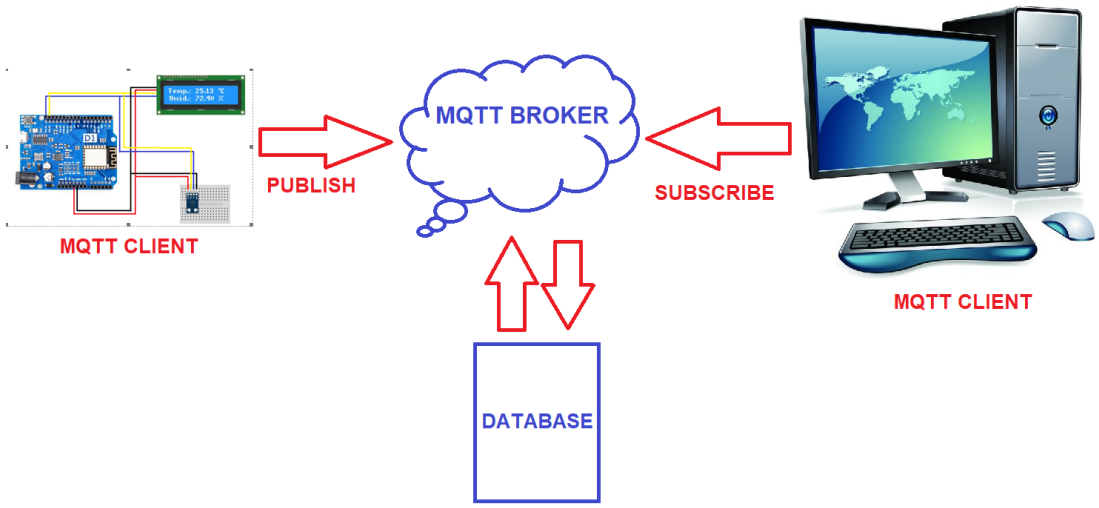

Figura 15. Esquematização do funcionameno do protocolo MQQT do nosso projeto

O MQTT possui dois componentes: Agente MQTT, que seria é um ponto central de comunicação, nesse caso, chamamos de broker, no qual é responsavel por despachar todas as mensagens entre os clients, e os Clientes MQTT, que são os dispositivos que se conectam ao broker, no nosso caso, o módulo ESP8266 faz essa comunicação ao enviar os dados para o API utilizada nesse projeto.

A Plataforma Thingspeak

A Plataforma Thingspeak funciona como um banco de dados que recebe informações geradas por dispositivos IoT e as disponibiliza para acesso por qualquer navegador web e por aplicativos móveis. Desta forma, o dispositivo que gera a informação não precisa ser acessado diretamente. Isto torna muito mais fácil a obtenção da informação, principalmente nos casos de IP flutuante e redes locais roteadas através

de um único IP. Com o Thingspeak podemos visualizar os dados em forma de gráficos e vários outros formatos e APIs.

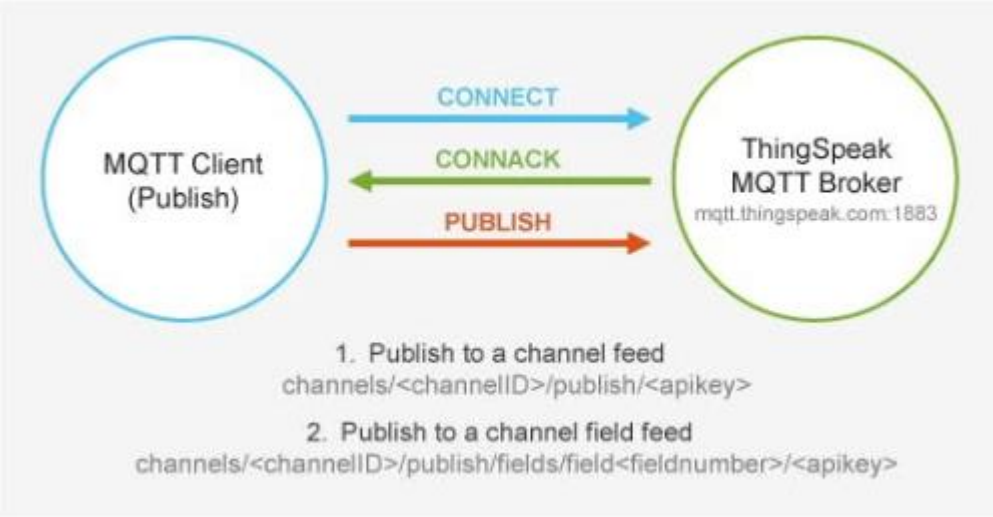

Figura 16. Funcionamento do MQTT na API Thingspeak

O elemento básico é o canal., cujo principal objetivo é armazenar os dados que são enviados. O canal é composto de 8 campos para armazenar dados de qualquer tipo. Estes são os dados enviados pelos dispositivos remotos e 3 campos de localização. Geralmente são utilizados para latitude, longitude e elevação. Excelente para rastrear um dispositivo em movimento.

Para que você possa utilizer a API em questão, basta você criar uma conta no site official do servidor. Quando a suca conta é criada, você poderá criar seus canais, conforme pode ser visto na figura 17, você pode utilizer 8 campos para armazenar os dados.

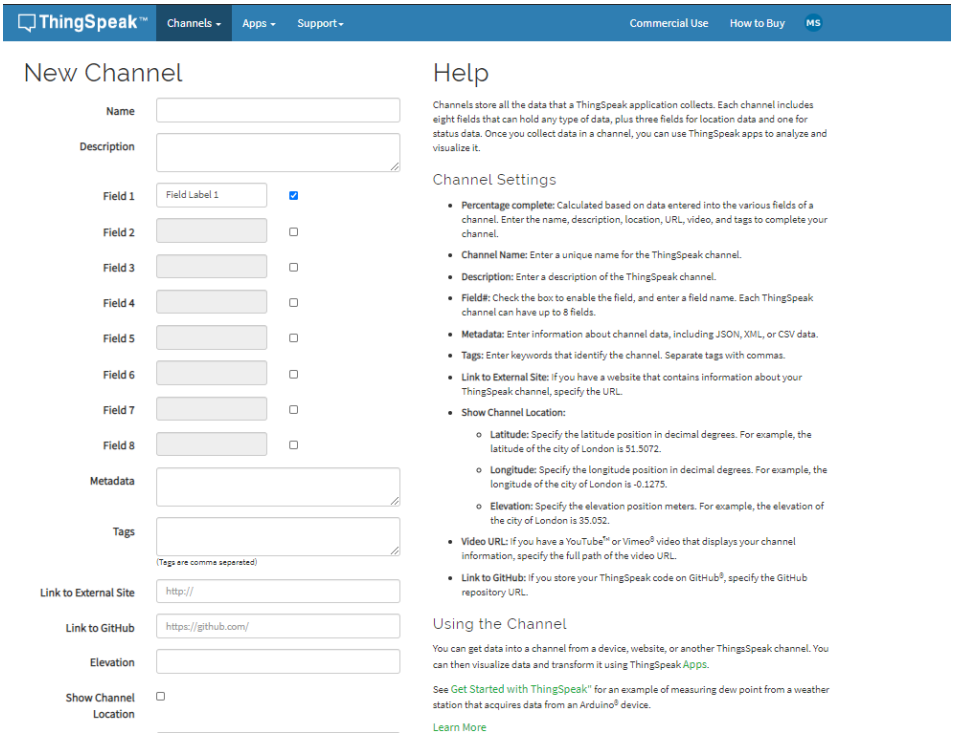

Figura 17. Criaçao de canal na Thingspeak

Ao criar o canal, será disponibilizado para você uma chave de acesso, onde você deverá colocar-la em seu código para a comunicação com o broker.

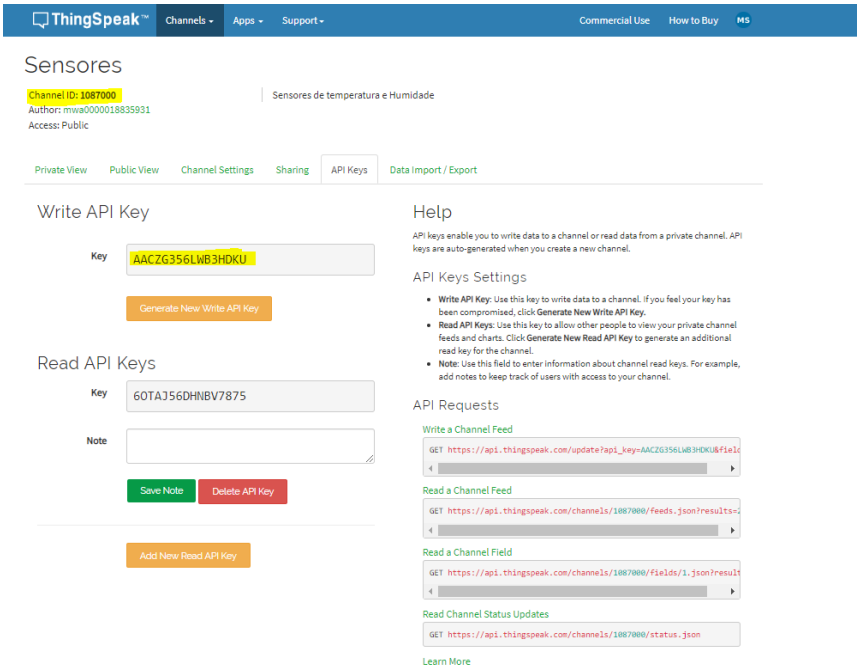

Figura 18. Geração de API Key na Thingspeak

Resultados

A montagem do circuito foi feita com sucesso conforme pode ser visto na figura. A precisão do sensor é tão precisa que em ambientes internos a temperatura por muitas vezes é maior, pelo fato dos ambientes serem mais quentes devido ao abafamento. Porém, quando testamos em um ambiente externo, a temperatura e a umidade relativa do ar foram medidas com sucesso.Em relação ao display, a sua instalação dentro d circuito também não foi dificil, visto que já tinhamos o o controlador HD44780 já soldado na hora da compra do componente em questão

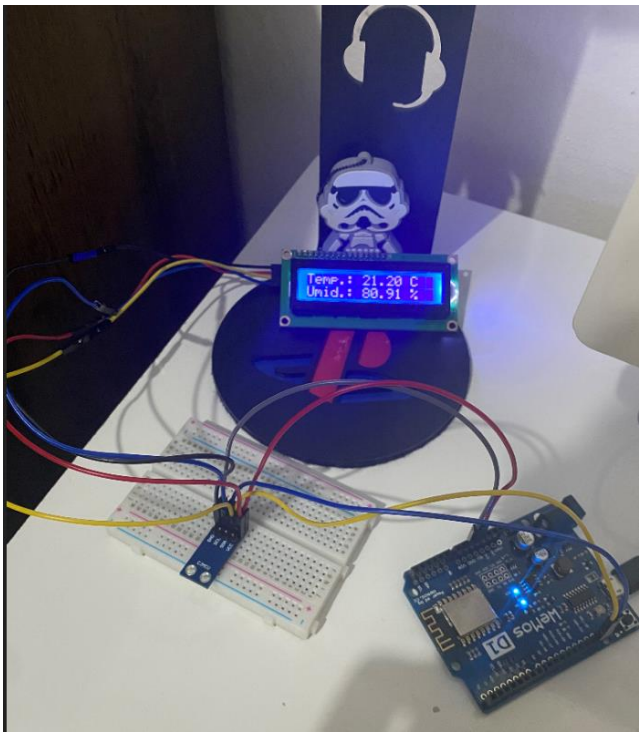

Figura 19. Circuito montado e funcionando

Referente a implantação do MQTT, a mesma foi realizada com sucesso devido a facilidade com a comunicação com a a API que possui uma boa base de documentação explicativa sobre MQTT, no qual facilitou no entendimento e conforme citado anteriormente, na construção do projeto. Outro ponto importante, é a disponibilidade de gráficos e estilização na qual podemos aplicar, isso é importante depenendo de como queremos visualizar os dados. Conforme verificdo na figura, temos as últimas medidas aferidas representadas no gráfico, como temos a informação da temperatura atual.

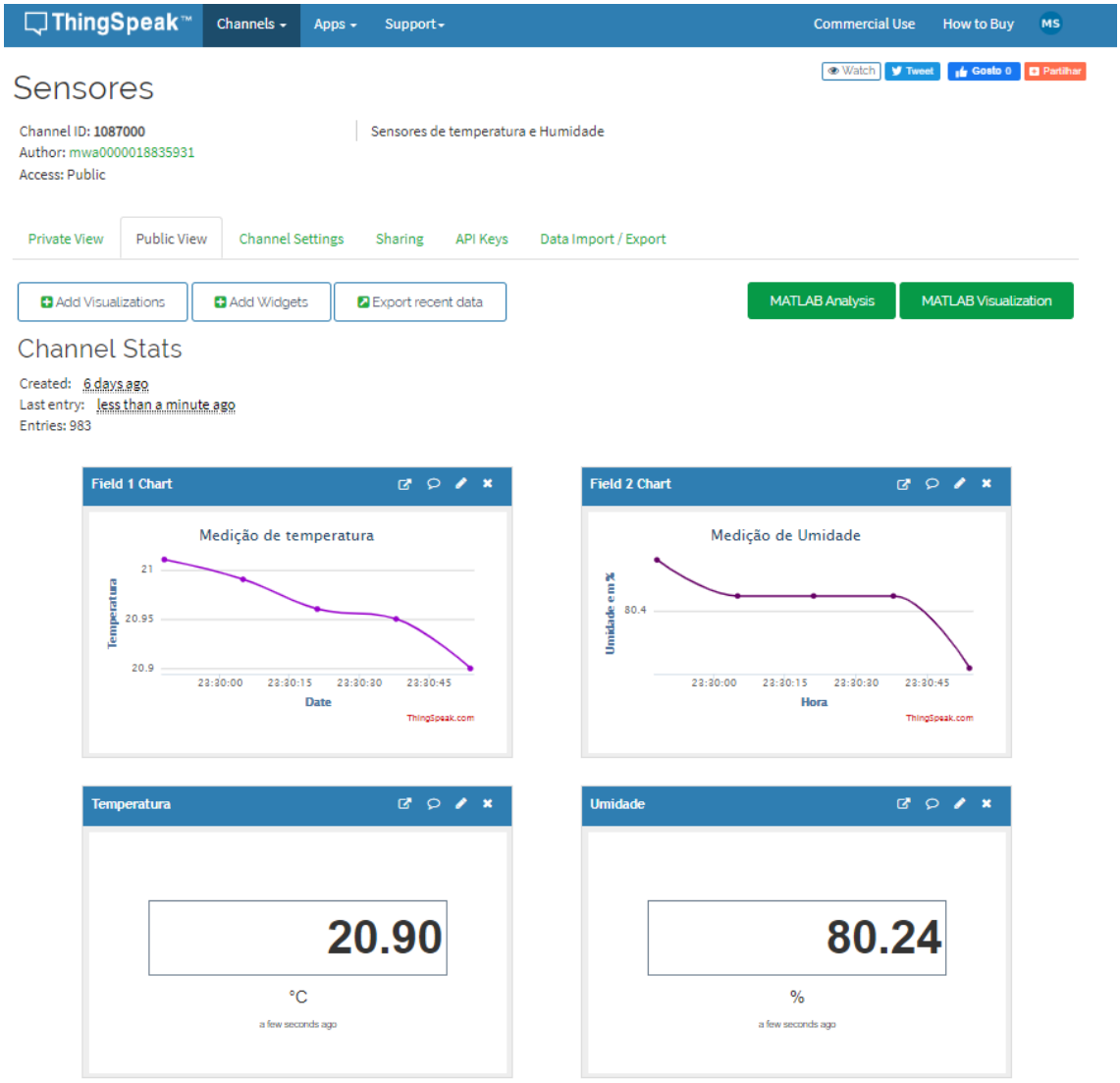

Figura 20. Representação gráfica dos dados medidos

Além disso, temos a opção de representa a leitura de forma, nesse caso, usando um widget de um medidor.

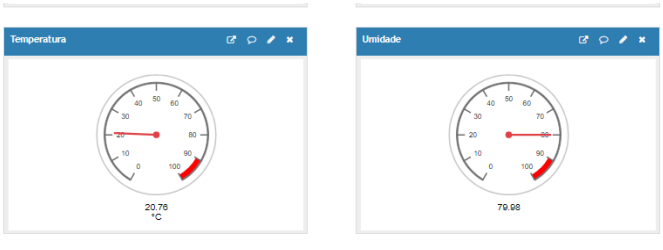

Figura 21. Widget adicional da aplicação

Além disso, a API nos disponibiliza um Código Iframe, caso, queiramos disponibilizar em algum site e também Podemos exporter no formato JSON, XML e CSV, de acordo com a nossa necessidade.

Figura 22. Formas de exportação de dados pela PI Thingspeak

Figura 23. Disponibilização de código HTML para utilizar em outros sites os resultados medidos pela aplicação

Porém, tivemos alguns problemas no desenvolvimento do circuito. De ínicio, queriamos utilizar o Arduino UNO com um um módulo ESP8266, porém, tivemos dificuldades ao implantar o MQTT, devido a conexão com a pinagem que não estava muito clara paranós. Por isso, optamos por utilizar a placa Wemos D1 por já ter o módulo aclopado. Outro problema que enfrentamos é devido a fragilidade do sensor de temperatura e umidade, conforme mostrado na figura 24. Por muitas vezes, a temperatura era apresentada no seu limite máximo, assim como a umidade.

Figura 24. Display LCD apresentação medições máximas suportadas pelo sensor de temperature e umidade HCD1080

Reparamos que tudo não passava de uma questão de encaixa na protboard, no fim, conseguimos adequar a montagem, e o problema aparentemente o problema parou de ocorrer.

Conclusões

Após a finalização do projeto, concluimos que conseguimos alcançar o objetivo inicial que queriamos, que era a implementação da comunicação MQTT no projeto, o uso da API que queriamos, além de conseguir representar as grandezas medidas em um display, fazendo um circuito até simples, economico, leve, e que entrega o que é proposto. Acreditamos que outras funcionalidades podem ser aplicadas nesse projeto, como a comunicação com mais um broker, a aquisição de outros atuadores e sensores, e também, a elaboração de um código até um pouco mais claro e coeso. A própria API do Thingspeak, possui integração com o Twitter, poderiamos criar um dispositivo, que além de aferir os dados e apresentar na API, poderia por exemplo, postar no próprio Twitter a temperatura atual e a umidade conforme ela for medida. Para maiores informações referente ao código, ao circuito, e a aplicação em si, podem ser encontrada no repositório do GitHub em https://github.com/MathSena/WemosHumidTemp
Também, possuimos um vídeo mostrando o funcionamento do projeto nio YouTube disponível em https://www.youtube.com/watch?v=uMxsGWlJOgg

Referências

Oliveira, Euller (2019), “Conhecendo a Wemos D1”

https://blogmasterwalkershop.com.br/embarcados/wemos/conhecendo-a-wemos-d1/

Texas instruments, “HDC1080 Low Power, High Accuracy Digital Humidity Sensor with Temperature Sensor (Rev. A) “

https://img.filipeflop.com/files/download/Datasheet_hdc1080.pdf

Rovai, Marcelo (2018) “IoT Feito Fácil: ESP-MicroPython-MQTT-Thingspeak”

https://mjrobot.org/2018/06/13/iot-feito-facil-esp-micropython-mqtt-thingspeak/

Arduino IDE

https://www.arduino.cc/en/main/software

GitHub, ClosedCube_HDC1080_Arduino

https://github.com/closedcube/ClosedCube_HDC1080_Arduino

GitHub, Arduino-LiquidCrystal-I2C-library

https://github.com/fdebrabander/Arduino-LiquidCrystal-I2C-library

Basílio, Shirley “Conhecendo o protocolo MQTT”

https://blogmasterwalkershop.com.br/outros/conhecendo-o-protocolo-mqtt/

Oliveira, Bruno (2017) “Dando uma breve análise no protocolo MQTT”

https://medium.com/internet-das-coisas/iot-05-dando-uma-breve-an%C3%A1lise-no-protocolo-mqtt-e404e977fbb6

MathWorks, Thingspeak

https://www.mathworks.com/help/thingspeak/

Tavares (2019), “Thingspeak. Enviando dados para a Internet”

https://cadernodelaboratorio.com.br/2019/03/09/thingspeak-enviando-dados-para-a-internet/

Canal do Projeto na Thingspeak

https://thingspeak.com/channels/1087000

GitHub do Projeto

https://github.com/MathSena/WemosHumidTemp

Video do Funcionamento do projeto WemosHumidTemp, YouTube

https://www.youtube.com/watch?v=uMxsGWlJOgg