UNIVERSIDADE PRESBITERIANA MACKENZIE
Faculdade de Computação e Informática

SmartIrrigate: Automatização do processo de regar as plantas.

Guilherme Heck Lara, Dhouglas Nogueira Saraiva, Wilian França Costa

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

guihecklara@gmail.com, dhouglas.saraiva@hotmail.com,
wilian.costa@mackenzie.br

Abstract. *This article aims to describe the project "SmartIrrigate", which has the function of measuring soil moisture and irrigating plants in an automated way, using a NodeMCU ESP8266 module together with a Hygrometer Sensor that measures soil moisture, activating when a Water Pump is necessary through a Relay that will feed the soil with water with a hose. It will be controlled via the MQTT protocol, being able to access a dashboard and check the soil moisture value at the moment and irrigate with necessary.*

Resumo. *Este artigo tem como objetivo descrever o projeto "SmartIrrigate", que tem a função de medir a umidade do solo e irrigar as plantas de maneira automatizada, utilizando um módulo NodeMCU ESP8266 juntamente com uma Sensor de Higrômetro que mede a umidade do solo, acionando quando necessário uma Bomba de Água através de um Relé que irá alimentar com uma mangueira o solo com água. Será controlado via protocolo MQTT, podendo acessar uma dashboard e verificar qual é o valor da umidade de solo no momento e irrigando se necessário.*

1. Introdução

A internet das coisas, vem mudando muito como a sociedade vê as tarefas e obrigações rotineiras, trazendo sempre soluções que são recorrentes no dia a dia, e tentando automatizar e facilitar a execução das mesmas. O problema em questão, é a criação de plantas, que é uma tarefa bem difícil e repetitiva, e a má gestão da mesma pode até matar a planta ou diminuir a produção dos mesmos.

O projeto tem como o objetivo facilitar a criação de plantas e vegetais de forma simples e automática, o sistema se baseará em regar as plantas quando um certo nível de umidade do solo for atingido, o sistema liberará água para irrigar a planta, tornando o processo de irrigação automático e simples.

2. Materiais e Métodos

- NodeMCU ESP8266

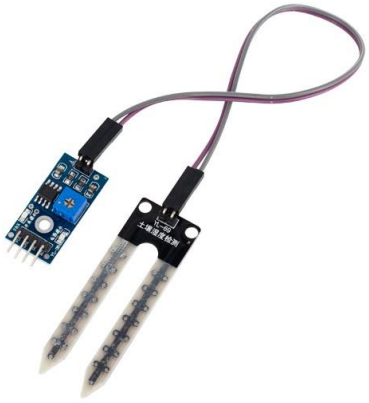
Figura 1 – Módulo Wifi NodeMCU ESP8266

Fonte: https://www.filipeflop.com/produto/modulo-wifi-esp8266-nodemcu-esp-12/

a. O módulo Wifi ESP8266 NodeMCU é uma placa de desenvolvimento muito utilizada e decidimos utilizar por conta do WiFi já inserido dentro deste componente, assim como todos os pinos necessários para o funcionamento. Um grande ponto também é a facilidade do conector USB e um regulador de tensão 3.3.V [1] , onde após serem efetuadas as configurações, é possível efetuar a implementação na IDE do Arduino.

- Sensor de Umidade

Figura 2 - Sensor de Umidade do Solo Higrômetro



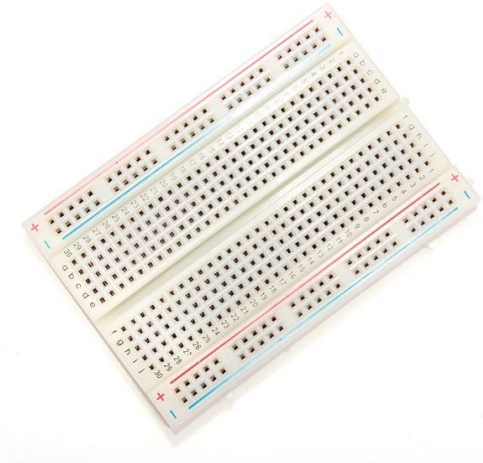
Fonte: https://www.filipeflop.com/produto/sensor-de-umidade-do-solo-higrometro/

a. O Sensor de Umidade do Solo Higrômetro é utilizado para identificar a condição do solo, utilizamos ela em nosso projeto para detectar se o solo está ou não na condição recomendada para o tipo de planta. O sensor é formado por duas hastes de eletrodos [2] que conduz uma corrente pelo solo, sendo possível identificar a qualidade. A outra parte do sensor é onde escolhemos se queremos utilizar a saída

digital (D0) ou a saída analógica (A0) para identificar qual o tipo de solo ideal para a planta utilizada.

- Protoboard 400 Pontos

Figura 3 – Protoboard para montagem de circuitos eletrônicos



Fonte: https://www.filipeflop.com/produto/protoboard-400-pontos/

a. Este Protoboard é uma excelente ferramenta para a montagem de circuitos eletrônicos, sendo uma maneira rápida, fácil e prática para montar seus projetos. Foi utilizada para montar os componentes com a placa ESP8266 para o funcionamento de maneira adequada.

- Módulo Relé

Figura 4 – Módulo Relé 5V 1 Canal

Fonte: https://www.filipeflop.com/produto/modulo-rele-5v-1-canal/

a. O Módulo Relé 5V de 1 Canal, funciona como um interruptor, sendo usado como controlador de dispositivos, em nosso caso, ele irá ligar e desligar a Mini Bomba de Água.

- Jumpers

Figura 5 – Kit de Jumpers

Fonte: https://www.filipeflop.com/produto/kit-jumpers-10cm-x120-unidades/

a. Foram utilizados Jumpers (macho-macho, fêmea-fêmea e macho-fêmea) para conectar o sensor, atuadores e o NodeMCU, podendo definir as portas de cada das determinada ligação.

- Mini Bomba de Água

Figura 6 – Mini Bomba de água submersível vertical

Fonte: https://www.filipeflop.com/produto/bomba-dagua-submersiva-3-6v-100l-h/

a. A Mini Bomba de água é utilizada para enviar o fluxo de água para uma mangueira fazendo a irrigação do solo quando for detectado que a condição do solo não é a ideal no momento. Os seus fios são ligados no Relé que controla o acionamento da bomba e outro é alimentado na energia 5V.

- Mangueira Cristal

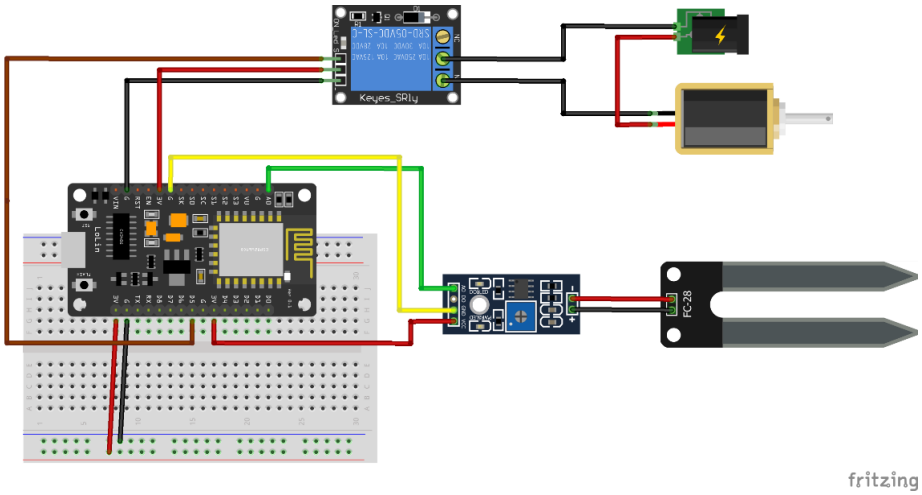
Figura 7 – Mangueira Cristal 3/8”x3mm

Fonte: https://www.indupropil.com.br/mangueira-cristal-atoxica-38-x-3mm.html

a. Mangueira transparente para ser conectada junto a Bomba de Água e alimentar o solo para umidecer.

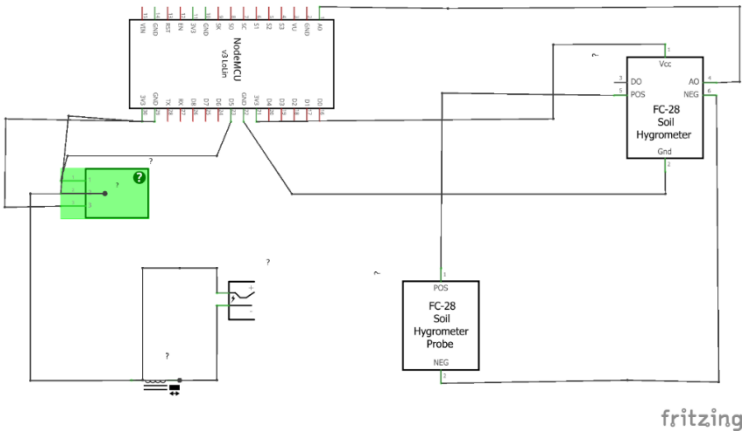
Utilizamos a ferramenta Fritzing para desenvolver um protótipo para a montagem do circuito do projeto, onde utilizamos para a montagem: NodeMCU ESP8266, Protoboard de 400 Pontos, Jumpers, Sensor de Umidade de Solo, Módulo Relé, Bomba de Água ligado a mangueira e energia 5V.

Figura 8 – Protótipo realizado no Fritzing



Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/images/ProjetoSmartIrrigate_Circuito.png

Figura 9 – Esquemático realizado no Fritzing



Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/images/ProjetoSmartIrrigate_Esquem%C3%A1tico.png

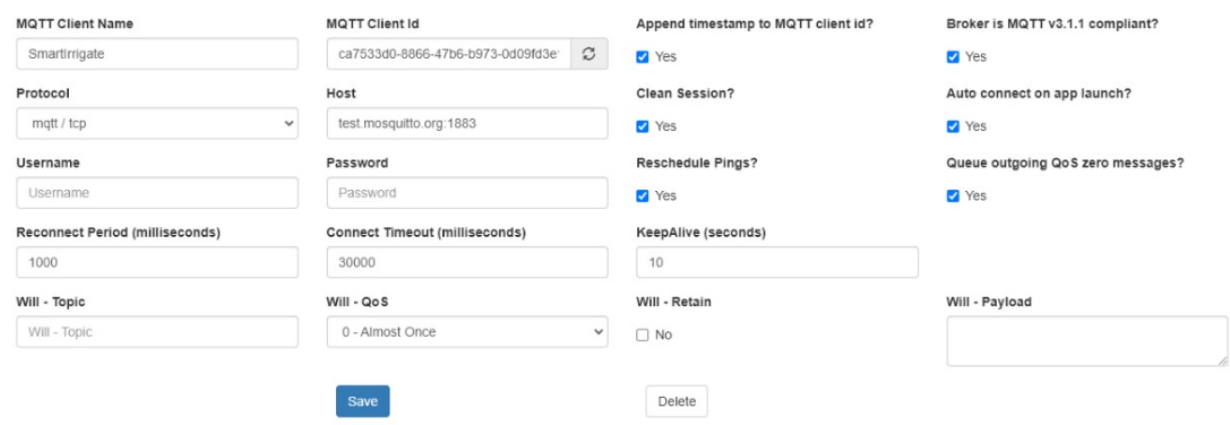
A implementação do sistema foi realizada no Arduino IDE, onde fizemos as configurações necessárias para conectar a placa NodeMCU ESP8266 e instalando todas as bibliotecas para funcionamento do projeto. Essas bibliotecas foram necessárias para fazer a conexão do Módulo NodeMCU com a internet WiFi e a comunicação do sistema com o protocolo MQTT: utilizamos o ESP8266Wifi [3] e PubSubClient [4].

Na parte de comunicação com o MQTT, utilizamos o broker test.mosquitto.org [5] apontando para a porta 1883. Optamos por um broker público por conta de ser simples a execução dos testes. Na implementação do sistema criamos o seguinte tópico:

- topico_sensor_umidade: neste tópico, fazemos a inscrição no ESP8266 que busca e retorna o valor atual da umidade do solo, sendo mostrado na dashboard do broker.

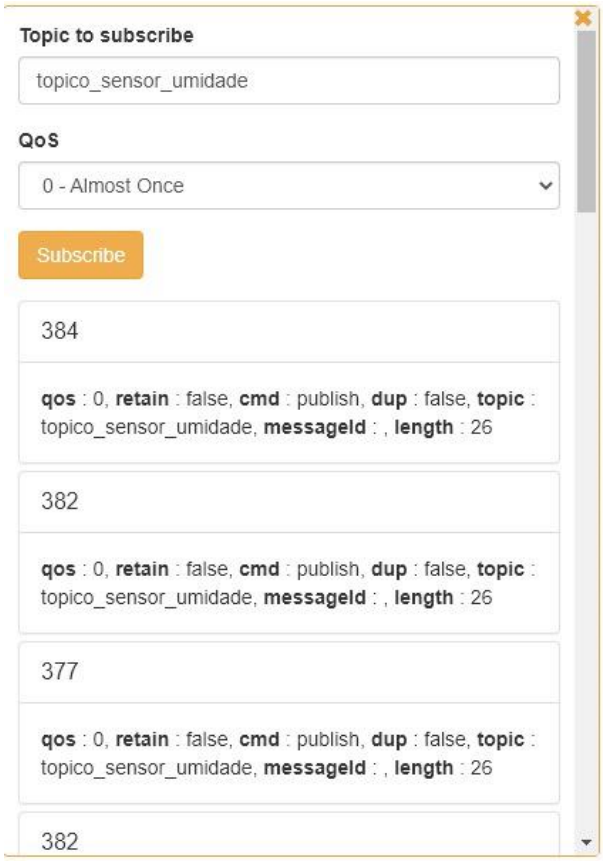
As imagens abaixo demonstram as chamadas feitas no topico_sensor_umidade, juntamente com as configurações do MQTT realizadas e o retorno do valor que está sendo coletado do solo.

Figura 10 – Configurações de chamada tópico MQTT



Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/images/ConfigChamadaMQTT.jpeg

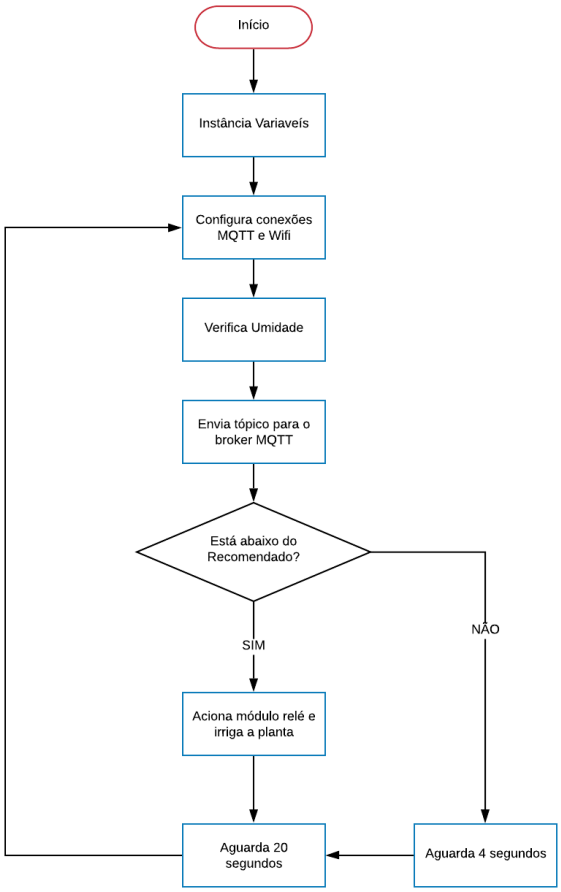
Figura 11 – Retorno do valor da umidade



Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/images/RetornoTopicoSensorUmidade.jpeg

O fluxograma abaixo mostra visualmente o fluxo do projeto, onde após a inicialização de variáveis e configurações de conexão, o sistema checa e responde de acordo com o resultado coletado no solo e devolvendo via protocolo MQTT para o topico_sensor_umidade.

Figura 12 – Fluxograma do sistema



Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/images/Fluxograma.png

No inicio do codigo começamos importando as bibliotecas necessárias para o funcionamento correto do sistema. Essas bibliotecas, são respectivamentes usadas para configurar o WiFi de conexão do ESP8266 e para configurar o broker MQTT.

Figura 13 – Importando bibliotecas

```
// Importando as bibliotecas necessárias
# include <ESP8266WiFi.h>
# include <PubSubClient.h>
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

Logo após configuramos os dados de configuração para conectar com o MQTT, onde primeiramente definimos um ID e o nome do tópico onde iremos dar Subscribe. Depois configuramos a url de conexão com broker público e a porta que usaremos, em nosso caso escolhemos a porta descriptografada.

Figura 14 – Declarando variaveis do tópico MQTT e broker

```
// Define configs do MQTT
# define ID_MQTT "smart_irrigate_mqtt"
# define TOPICO_PUBLISH_UMIDADE "topico_sensor_umidade"

// Endereço e a porta do broker
const char * BROKER = "test.mosquitto.org";
int PORTA = 1883;
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

Em seguida definimos os pinos que usaremos os nossos componentes do circuito, sendo eles:

- Módulo Relé (pino 2) que será usado para controlar o acionamento da bomba de água
- Sensor de Solo Higrômetro (pino 17) que será usado para identificar e retornar a umidade do solo no momento.

Figura 15 – Definindo pinos utilizados

```
// Definindo os pinos dos dispositivos
# define rele 2
# define sensor_umidade 17
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

Nesta etapa você irá declarar o nome e senha de sua rede WiFi para conexão com a internet e em baixo definimos o client para conexão com o MQTT.

Figura 16 – Declarando nome e senha do WiFi e Cliente

```
// Informações da rede WiFi
char * SSID = "";
char * PASSWORD = "";

WiFiClient espClient;
PubSubClient MQTT(espClient);
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

Por fim, instanciamos as últimas variáveis globais necessárias na lógico do loop(), um valor inteiro para a umidade e uma coleção de char para tráfego de mensagens.

Figura 17 – Declarando variáveis globais

```
//Criando as váriaveis globais
int umidade;
char * msg = "";
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

Criamos métodos para fazer e verificar as conexões, o primeiro é o ReconectMQTT que garante a conexão do client MQTT para a ESP822 com o broker definido. Ela é verificada no início do loop.

Figura 18 – Método que reconecta com o MQTT

```
// Função que se conecta ao MQTT
void ReconectMQTT(void) {
  if (!MQTT.connected()){
    while (!MQTT.connected()) {
      Serial.print("* Conectando-se ao MQTT: ");
      Serial.println(BROKER);
      if (MQTT.connect(ID_MQTT)) {
        Serial.println("Conectado com sucesso");
      } else {
        Serial1.println("Falha ao se conectar.");
        Serial.println("Reiniciando em 2s");
        delay(2000);
      }
    }
  }
}
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

Método que inicia o MQTT dentro do setup

Figura 19 – Método que inicia o MQTT

```
// Iniciando o MQTT
void StartMQTT(void) {
  MQTT.setServer(BROKER, PORTA);
}
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

O método ReconectWIFI garante a conexão com WiFi, ela é chamada no início do loop, juntamente com a função do ReconectMQTT.

Figura 20 – Método que reconecta com o WiFi

```
// Função que reconecta o serviço de WiFi
void ReconectWIFI(void) {
  if (WiFi.status() == WL_CONNECTED)
    return;
  WiFi.begin(SSID, PASSWORD);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }

  Serial.print("Conectado na rede ");
}
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

Na execução do setup, chamamos a função para iniciar o MQTT, além de definir o pinMode de nossos componentes do circuito.

Figura 21 – Função setup

```
void setup() {

  Serial.begin(115200);

  pinMode(sensor_umidade, INPUT);
  pinMode(rele, OUTPUT);

  StartMQTT();
}
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

A função ConsultaConexoes é chamada no início do loop para verificar as conexões com o WiFi e MQTT, para garantir o funcionamento do sistema de forma integra.

Figura 22 – Função que verifica se o WiFi e MQTT estão conectados

```
void ConsultaConexoes(void) {
  ReconectWIFI();
  ReconectMQTT();
}
```

Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

Por fim temos o loop do sistema, começamos por verificar as conexões na função ConsultaConexoes, logo após fizemos um analogRead no nosso sensor de umidade, que retorna qual o valor do solo no momento, este valor pode ser de 0 a 1024, quando ele

estiver abaixo de 400 significa que o solo está seco e deve ser irrigado. Logo após coletas a umidade do solo, enviamos para o broker do MQTT, que mostrará em sua dashboard qual o valor do solo no momento, podendo saber se está irrigado ou não. Fazemos um IF ELSE simples para detectar se que relé deve ativar a bomba, caso a umidade esteja entre 0 e 399 a bomba será ativada, caso ela esteja entre 400 e 1024 a qualidade do solo está boa e a bomba não será acionada.

Figura 23 – Fluxo principal do sistema

```
void loop() {
  ConsultaConexoes();
  umidade = analogRead(sensor_umidade);
  sprintf(msg, "%d", umidade);


  Serial.println("Valor da umidade");
  Serial.println(umidade);

  MQTT.publish(TOPICO_PUBLISH_UMIDADE, msg);
  MQTT.loop();
  delay(10000);

  // Caso a umidade seja baixa
  if (umidade>=0 && umidade<400 ) {
    Serial.println("UMIDADE BAIXA");
    digitalWrite(rele, HIGH);
    delay (2000);
    digitalWrite(rele, LOW);
  }

  else if (umidade>=400 && umidade<1025 ) {
    Serial.println("UMIDADE BOA");
    digitalWrite(rele, HIGH);
    delay (2000);
    digitalWrite(rele, LOW);
  }

  delay(10000);
}
```

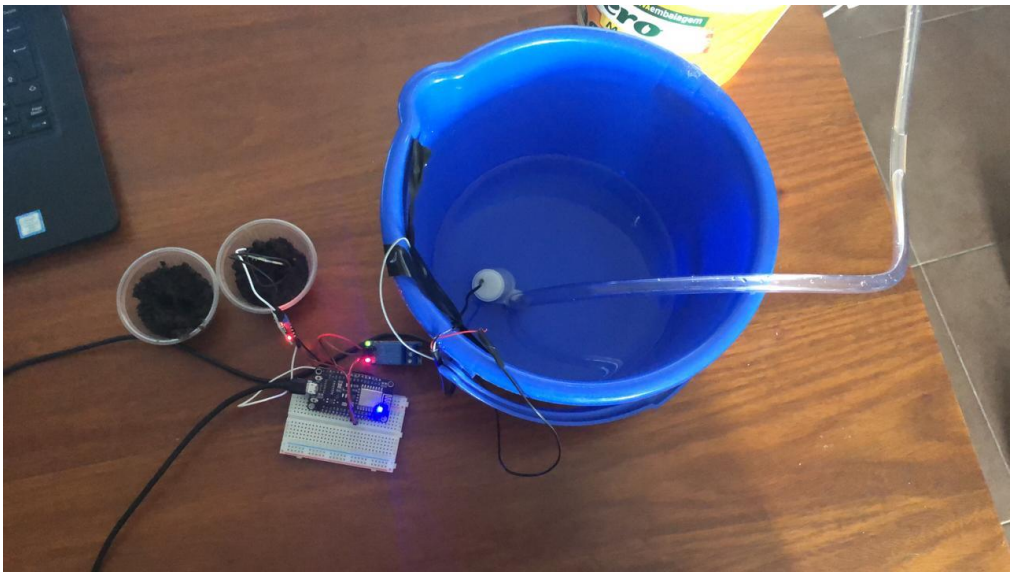
Fonte: https://github.com/dhougsaraiva/SmartIrrigate/blob/main/README.md

3. Resultados

O circuito e os dispositivos conseguem se comunicar com os protocolos definidos. O projeto é capaz de enviar a valor da umidade do solo para o seu usuário via protocolo MQTT. Utilizando aplicativos de celular como o MQTTool (iOS), MQTT Dash (Android) ou MQTT Box (Extensão Chrome) foi possível fazer os testes de conexão com o tópico e verificar os valores retornados do solo, podendo certificar que o projeto estava funcionando de maneira correta com o imaginado. Durante a execução do projeto a nossa maior dificuldade foi com os componentes utilizados, tivemos que efetuar algumas compras até acertar qual seria o conjunto de componentes ideal para o funcionamento correto do projeto.

Após realizar todo o entendimento, o projeto e a lógico dele são simples. Efetuando a compra de todos os componentes necessários você pode replicar facilmente esse projeto, a montagem do projeto deve ser de acordo com o protótipo de circuito e o vídeo de demonstração do projeto que pode ser acessado em https://youtu.be/Eccq4j8x5TI [6], para a alimentação do NodeMCU ESP8266 é necessário conectá-lo a um USB e para a Bomba de Água também é necessário conecta-la a uma energia 5V, em nosso caso usamos outro cabo USB. Para melhor entendimento também disponibilizamos um repositório com toda documentação e código disponíveis e necessárias para a execução deste projeto, que pode ser acessado em https://github.com/dhougsaraiva/SmartIrrigate [7].

Figura 24 – Projeto montado



Fonte: Guilherme Heck Lara, 2020

4. Conclusões

Nessa primeira versão conseguimos atender uma dor comum dos clientes que possuem plantas em casa, sendo uma alternativa de fácil de acesso e implementação e que irá contribuir muito para a saúde de suas plantas. Conseguimos executar todas as funções necessárias para o funcionamento do projeto, está tudo funcional e interagindo da maneira correta. Com certeza a grande vantagem desse projeto é a sua simplicidade, depois de ser bem compreendido tudo começa a fluir da maneira correta e se torna bem fácil o entendimento, e justamente por essa facilidade, é muito vantajoso um hardware simples colaborar tanto com um ecossistema e para o cotidiano do usuaário. Nossas maiores dificuldades na realização desse projeto foram problemas com os componentes, tivemos que efetuar algumas compras até acertar quais seriam os componentes ideais. Por engano, acabamos comprando um Relé de 12V ao invés do necessário que seria um de 5V, felizmente esse problema foi resolvido a tempo. Outro problema foi que compramos um Sensor de Solo que não vinha com o módulo necessário para leitura, então tivemos que acabar comprando outros depois. Depois de algumas tentativas de componentes, após realizar a montagem do circuito, tudo começou a fluir de maneira tranquila, sem dificuldades com a lógico da programação e também com as conexões WiFi e Broker MQTT. Para próximas versões

algumas melhorias poderiam ser feitas, como: Implementação para o cliente receber notificações sempre que sua planta for regada, assim como um relatório ou gráfico para visualizar de quanto em quanto tempo normalmente a planta está sendo regada, assim como a quantidade de água que está sendo gasta. Em questão de componentes poderia ter fios maiores para a água não ficar perto do circuito, correndo risco de acontecer algum acidente e estragar o projeto. Assim concluímos a execução do projeto com objetivo alcançado com sucesso e um sistema funcional e automatizado para irrigação de plantas.

5. Referências

[1] ELETROGATE, "NodeMCU – ESP12: Guia Completo – Introdução", Disponível em: https://blog.eletrogate.com/nodemcu-esp12-introducao-1/. Acesso em: 05/11/2020.

[2] VIDA DE SILICIO, "Sensor de Umidade do Solo – Higrômetro". Disponível em: https://www.vidadesilicio.com.br/sensor-umidade-solo-higrometro. Acesso em: 05/11/2020.

[3] FLIPEFLOP, "Tutorial Módulo Wireless ESP8266 com Arduino". Disponível em: https://www.filipeflop.com/blog/esp8266-arduino-tutorial/ . Acesso em: 06/11/2020.

[4] KNOLLEARY, "API Documentation Arduino Client for MQTT". Disponível em: https://pubsubclient.knolleary.net/api. Acesso em: 06/11/2020.

[5] ECLIPSE MOSQUITTO, "Documentação do protocolo MQTT". Disponível em: https://mosquitto.org/. Acesso em: 06/11/2020.

[6] SmartIrrigate – Projeto Objetos Inteligentes – Youtube. Direção e Produção: Dhouglas Saraiva e Guilherme Heck. São Paulo, 20/11/2020. Disponível em https://youtu.be/Eccq4j8x5TI.

[7] Microsoft, GitHub, 2020. Código fonte do projeto SmartIrrigate. Disponível em: https://github.com/dhougsaraiva/SmartIrrigate. Acesso em 19/11/2020.