Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

PROTÓTIPO DE SISTEMA EMBARCADO IOT DE COLETA DE TURBIDEZ DA ÁGUA UTILIZANDO O MICROCONTROLADOR ESP32 LORA

PROTOTYPE OF IOT EMBEDDED SYSTEM FOR DATA COLLECTION OF WATER TURBIDITY

Evandro de Assis Rosa[1] Ricardo Rall[2]

RESUMO

A escassez de água potável nos rios está impulsionando a criação de protótipos para monitorar a qualidade da água, essencial para a saúde pública, o desenvolvimento sustentável das comunidades e tomada de decisões por governos e autoridades, orientando políticas de gestão da água. Este trabalho objetivou a construção de um protótipo para coletar dados de turbidez de rios e lagos e enviá-los para uma plataforma online, permitindo visualização quase em tempo real. Utilizando microcontroladores ESP32 Lora em topologia ponto-a-ponto e protocolo Lora, os dados foram disponibilizados a um MQTT *Broker* via protocolo MQTT, *Message Queuing Telemetry Transport*, permitindo que dispositivos subscritos acessassem os dados. A coleta foi realizada em um lago na cidade de Botucatu durante uma hora, obtendo-se valor médio de turbidez de 274,92 NTU (*Nephelometric Turbidity Unit*). A tecnologia LoRa ofereceu uma boa combinação de alcance, baixo consumo de energia, penetração e custo efetivo, sendo atraente para comunicações sem fio em IoT e monitoramento remoto. Apesar de obter resultados coerentes, o protótipo apresentou dificuldades, especialmente na calibração do sensor de turbidez.

Palavras-chave: IoT. ESP32. LoRa, turbidez. sensores. MQTT.

ABSTRACT

The scarcity of drinking water in rivers is driving the development of prototypes to monitor water quality, essential for public health and the sustainable development of communities, data for decision-making by governments and authorities, guiding water management policies. This work aimed to build a prototype for modeling turbidity data from rivers and lakes and sending them to an online platform, allowing visualization in almost real time. Using ESP32 Lora microcontrollers in a point-to-point topology and Lora protocol, the data was made available to an MQTT Broker via the MQTT, *Message Queuing Telemetry Transport*, protocol, allowing subscriber devices to access the data. A collection was carried out in a lake in the city of Botucatu for one hour, obtaining an average turbidity value of 274.92 NTU (*Nephelometric Turbidity Unit*). LoRa technology offered a good combination of range, low power consumption, penetration and cost effectiveness, making it attractive for IoT wireless communications and remote monitoring. Despite obtaining consistent results, the prototype presented difficulties, especially in configuring the turbidity sensor.

Keywords: IoT, LoRa, turbidity, sensors, MQTT.

1 Graduado em Análise e Desenvolvimento de Sistema pela Faculdade de Tecnologia de Botucatu.
2 Professor Doutor de Ensino Superior pela Faculdade de Tecnologia de Botucatu. Avenida José Ítalo Bacchi s/n – jardim Aeroporto – Botucatu – SP CEP: 18606-855. e-mail: ricardo.rall@fatec.sp.gov.br

Recebido em dezembro de 2024 e aceito em março de 2025.

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

1 INTRODUÇÃO

Um dos grandes problemas que a humanidade enfrenta atualmente, e que é agravado com o passar dos anos, é a escassez de água potável. A água ocupa 71% da superfície do planeta. Desse total, mais de 97,0% correspondem as águas salgadas e 3,0% águas doces, distribuídos em 2,5% de água doce congeladas em geleiras e calotas polares (água em estado sólido) e somente 0,5% em água doce (Melo; Moura; Oliveira, 2022).

Para dimensionar o tamanho do problema de escassez atual, segundo Jaeger et. al., o acesso cada vez menor à água é um problema significante para até 2 bilhões de pessoas no mundo, impactando a produção de alimentos, desenvolvimento econômico e sistemas ecológicos, e a falta de água já afeta cerca de 40% da população mundial.

Nas últimas décadas a humanidade desenvolveu diversas ferramentas com potencial de ajudar a combater problemas socioeconômicos, possibilitando monitorar variáveis que influenciam na qualidade da água de rios, lagos e fontes, como por exemplo a acidez, turbidez e presença de substâncias consideradas poluentes. Esse monitoramento é feito há décadas por meio coleta manual de água em pontos específicos de rios, levando as amostras para análise em laboratório por meio de diversas técnicas (Wang *et al*, 2018). Com as grandezas quantificadas, é possível utilizar cálculos do índice de qualidade da água, IQA, que leva em conta fatores, como potencial hidrogeniônico (pH) e turbidez, para avaliar e gerir recursos hídricos, mostrando a evolução da qualidade ao longo do tempo. O IQA foi desenvolvido em 1970, nos Estados Unidos, pela National Sanitation Foundation, e introduzido no Brasil em 1975, através da Companhia Ambiental do Estado de São Paulo (Cetesb). Posteriormente adotado por outros estados, continua a ser o principal índice de aferição dos níveis de qualidade da água usado no país (Félix *et al*, 2023).

Dentre as grandezas avaliadas, temos a turbidez que está relacionada à dificuldade que feixes luminosos encontram em atravessar um meio que possui partículas suspensas (De Oliveira Junior; De Lima Silva; Ferreira Prado, 2023). Por meio de sensores é possível confirmar o senso comum de que água suja não é boa para consumo.

Estes podem substituir a coleta de água de rios e lagos, principalmente quando acoplados a sistemas computacionais de baixo custo e consumo, como os sistemas embarcados, sendo que tais sistemas devem ser capazes de se comunicar com outros, para transmitir dados coletados e interessados possam realizar análises (Andrade *et al*, 2022).

Neste cenário a comunicação se faz por meio de redes sem fio. Entre as mais viáveis estão a comunicação por meio da tecnologia de WiFi, com alcance de algumas dezenas de

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

metros e por sinais de rádio LoRa (*Long Range*) que pode alcançar facilmente quilômetros de distância. A escolha do tipo de comunicação envolve uma análise de *tradeoff* de alcance de sinal x tamanho dos pacotes de rede. Caso queira muito alcance e baixo tamanho de pacotes, sinais de rádio são mais adequados e vice-versa (Boras *et al*, 2019).

Um sistema de coleta de dados de água de rios e lagos se beneficia mais de sinais de rádio do que *WiFi*, principalmente no cenário em que vários coletores são espalhados ao longo do rio e estes enviam sinal para um concentrador, formando uma topologia de rede tipo estrela. Isso se deve ao fato de que os pacotes de rede terão apenas alguns bytes, e o sinal precisa atingir grandes distâncias. Para se construir esse tipo de sistema, é possível testá-lo com topologia ponto a ponto e posteriormente criar a rede no formato estrela, evitando adquirir equipamentos sem certeza da viabilidade do sistema de coleta.

Os sinais enviados ao concentrador, podem aproveitar protocolos de rede como o MQTT, *Message Queue Transport Telemetry,* que costumam ser mais eficientes que HTTP e a eficiência relativa tende a com o número de dispositivos (Sasaki;Yokotani, 2018). Outra vantagem em utilizá-lo é a facilidade em disponibilizar o sinal para diversos interessados com custo baixo, pois os MQTT *Brokers*, responsáveis por mediar as publicações/subscrições dos tópicos, costumam ter opções baratas para poucos aparelhos conectados. Uma vez que o tópico que o concentrador publica as mensagens é divulgado, basta fazer a subscrição a ele e começar a acessar os dados (Spohn, 2023).

Dado o problema descrito acima, este trabalho teve como objetivo a construção de um protótipo capaz de coletar dados de turbidez de rios e lagos e enviá-los para uma plataforma de dados na internet, na qual será possível a visualização dos dados coletados em tempo próximo do real.

2 MATERIAL E MÉTODOS

O projeto foi iniciado com o levantamento dos materiais necessários para o desenvolvimento do sistema. Uma vez adquiridos, foi criada a arquitetura do sistema, para que fosse possível desenvolver cada uma das partes por vez. Cada parte desenvolvida foi testada e validada, para que pudesse ser incorporada ao sistema.

2.1 Materiais e softwares utilizados na construção do sistema

- 2 SoC (System on Chip) HELTEC ESP32 LoRa v2;
- 1 sensor de turbidez DFRobot SKU_SEN0189;

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

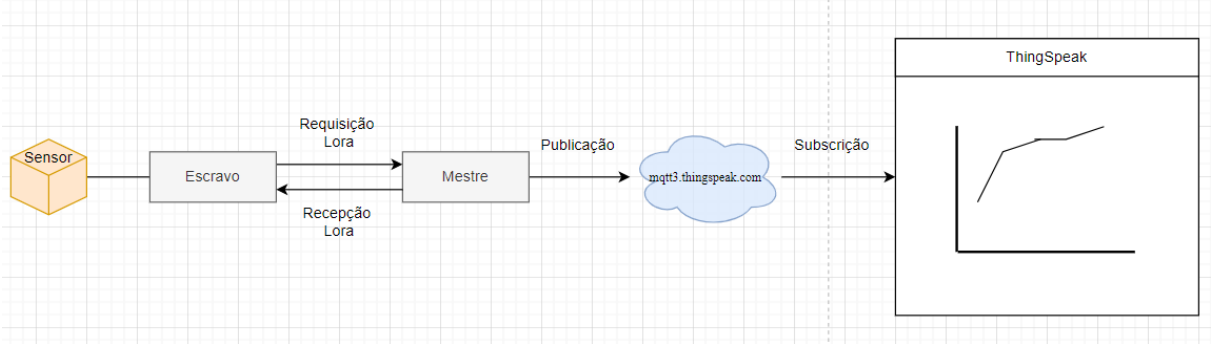
- 3 Protoboards de 400 pontos;
- Jumpers;
- Resistores de 10KΩ e 1KΩ;
- Cabos USB;
- Arduino UNO R3;
- Notebook com 8gb RAM, SSD 240GB, placa de vídeo 2GB e sistema operacional Ubuntu 20.04;
- 2 power banks com entrada USB;
- Smartphone com conexão 4G a internet
- Arduíno IDE 1.8.0;
- Plataforma IoT ThingSpeak;

2.2 Arquitetura proposta para o sistema

O sistema desenvolvido tem como seu coração os 2 ESP32 LoRa, que são os hardwares que recebem os códigos C da Arduíno IDE e controlam o funcionamento. Um deles atua como coletor dos dados do sensor de turbidez (*slave*), enquanto o outro faz a recepção (*master*/*gateway*). Ambos enviam sinais de rádio LoRa através de um chip que fica abaixo dos seus displays OLED e recebem sinais por meio das antenas.

Uma vez que os dados chegam ao *master*, este tenta enviar os dados para a plataforma ThingSpeak por meio do protocolo MQTT (publish). O sinal é publicado no *Broker* MQTT da plataforma (mqtt3.thingspeak.com), e o canal responsável pela construção dos gráficos faz subscrição no tópico, conforme ilustra a Figura 1.

Figura 1 - Arquitetura do sistema de coleta e visualização da turbidez da água.



Fonte: Proprio Autor. 2023.

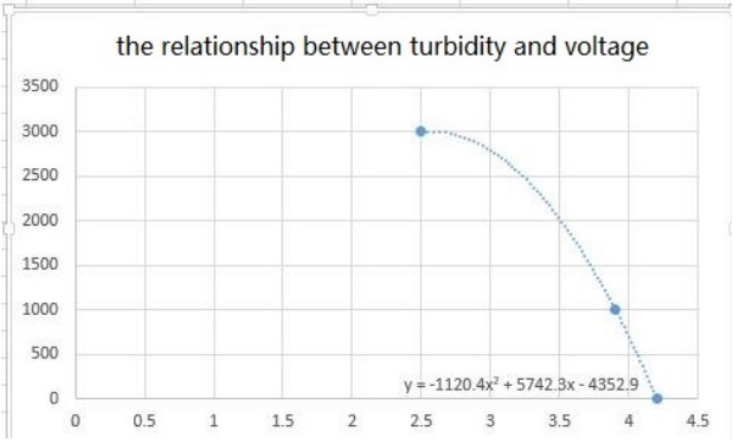
Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

2.3 Calibração do sensor de turbidez

A turbidez da água é uma grandeza associada à dificuldade de a luz atravessar um meio que tem partículas suspensas e é quantificada em Unidades Nefelométricas de Turbidez (NTU). No caso do sensor utilizado, a turbidez é medida através da dificuldade de um feixe de luz infravermelha atravessar o meio. A resistência do sensor infravermelho varia com a quantidade de luz que chega até ele, sendo que a tensão no resistor é capturada e convertida para NTUs por meio da fórmula Turbidez(x) = -1120.4 * x^2 + 5742.3 * x - 4352.9

O sensor da DFRobot é calibrado segundo o gráfico de turbidez em função da voltagem (Figura 2). A calibração é realizada levando em consideração o valor da turbidez nos extremos da parábola, de maneira que a voltagem em meio com turbidez < 0.5 NTU é 4.2V e quando a turbidez é ~3000 NTU (turbidez máxima) a tensão é de 2.5V.

Figura 2 - Curva da turbidez em função da voltagem para o sensor da DFRobot.



Fonte: DFRobot Wiki (2024).

Esse ajuste é feito por um *trimmer*, um parafuso que se encontra na placa do sensor, e que é protegido por um invólucro de plástico, como pode ser visto na Figura 3A. O *trimmer* é extremamente sensível, o que dificulta muito a calibração. A tensão no sensor foi medida com um Arduíno Uno R3, uma vez que os pinos de propósito geral I/O, conhecidos pela sigla GPIO, do ESP32 LoRa não aceitam tensões superiores a 3.3V. O processo consiste em tirar a tampa do sensor, girar o trimmer, fechar a tampa, mergulhar o sensor no líquido a ser medido, medir a tensão com auxílio do Arduino UNO e obter os valores através do seu Serial Monitor (Figura 3B).

Foram utilizados dois líquidos para testar os pontos extremos, água pura e café concentrado. A água pura apresentava valores bastante próximos do valor máximo de Tensão, no qual a turbidez se aproxima de 0 NTU. Já no caso do café solúvel concentrado, os valores

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

de Turbidez mal chegavam a 2000 NTU e, portanto, não ofereciam valores adequados para a calibração do sensor.

Figura 3 - Calibração do sensor por meio do trimmer e código da calibração

```
const int PinoTurbidez = A0;
float Tensao = 0

void setup()
{
  pinMode(PinoTurbidez, INPUT);
  Serial.begin(9600);
}

void loop()
{
  float Leitura = analogRead(PinoTurbidez);
  Tensao = Leitura * 5 / 1023;
  Serial.print("Tensao: ");
  Serial.print(Tensao);
  Serial.println("V");
  delay(2000);
}
```

Fonte: Proprio Autor. 2023.

2.4 Instalação do board Heltec ESP 32 LoRa v2 no Arduino IDE e bibliotecas

Para iniciar a codificação no ESP32, inicialmente deve-se instalar o board Heltec WiFi LoRa 32(v2), do Boards Manager da Arduino IDE. Se o board não for instalado, a Arduino IDE é incapaz de reconhecer o SoC, impossibilitando o desenvolvimento.

Além do Board, também são necessárias as bibliotecas PubSubClient.h, para a comunicação MQTT com o *Broker*, SPI.h para utilização do display OLED e ativação do LoRa, além da biblioteca LoRa.h que tem as funções de envio e recepção dos pacotes LoRa.

2.5 Setup do Master e Slave

Para que sejam possíveis as comunicações *master*/*slave* e *gateway*/*broker*, devem ser preparadas as configurações iniciais dos ESP32 para habilitar o LoRa, MQTT e WiFi. A configuração no *slave*/*master* é idêntica e feita através da função setupLora(), onde a comunicação SPI é iniciada para habilitar o LoRa, como pode ser visto na Figura 4.

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

Figura 4 - Função setupLora() que inicia o sinal LoRa nos ESP32.

```
/*Configurações iniciais do LoRa*/
void setupLoRa()
{
  /*Inicializa a comunicação*/
  SPI.begin(SCK, MISO, MOSI, SS);
  LoRa.setPins(SS, RST, DI00);

  /*Inicializa o LoRa*/
  if (!LoRa.begin(BAND))
  {
    /*Se não conseguiu inicializar, mostra uma mensagem no display*/
    display.clearDisplay();
    display.setCursor(0, OLED_LINE3);
    display.println("Reinicie o Lora...");
    display.display();
    while (1);
  }

  /*Ativa o crc*/
  LoRa.enableCrc();

  /*Ativa o recebimento de pacotes*/
  LoRa.receive();
}
```

Fonte: Proprio Autor. 2023.

No *master*, também são necessárias as inicializações da conexão WiFi e MQTT, que são feitas pelas funções init_wifi() e init_MQTT(), como mostra a Figura 5.

Figura 5 - Funções de inicialização do WiFi e MQTT.

```
/* Função: inicaliza conexão wi-fi
 * Parâmetros: nenhum
 * Retorno: nenhum
 */
void init_wifi(void)
{
    connect_wifi();
}

/* Função: Configura endereço do broker e porta para conexão com broker MQTT
 * Parâmetros: nenhum
 * Retorno: nenhum
 */
void init_MQTT(void)
{
    MQTT.setServer(broker_mqtt, broker_port);
}
```

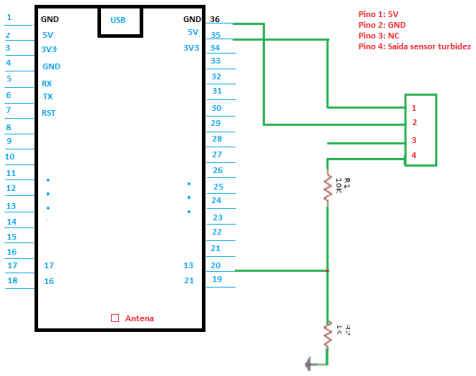
Fonte: Proprio Autor. 2023.

2.6 Leitura das tensões pelo Slave

Como dito anteriormente, os GPIOs do ESP32 LoRa aceitam uma tensão máxima de 3.3V e, por isso, não podem receber diretamente o sinal do sensor de turbidez. Para contornar esse problema, foi utilizado um divisor de tensão para que o sinal no GPIO de leitura seja 1/11 do original, utilizando resistores de 10KΩ e 1KΩ. Uma vez montado, o esquema elétrico da conexão do sensor com o *slave* está disposto na Figura 6.

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

Figura 6 - Esquema elétrico do *slave* com o sensor de turbidez conectado.



Fonte: Proprio Autor. 2023.

Na montagem deve-se tomar cuidado na escolha do GPIO, já que vários dos pinos do ESP32 LoRa tem outras finalidades, como a conexão com o chip LoRa, com o display OLED, etc. O GPIO 13 foi escolhido por não estar associado a outras funcionalidades, evitando sobrecarga no pino escolhido.

Com a montagem concluída, a função readData() (Figura 7) determina o valor em NTU da turbidez com fator de correção da tensão. Os GPIOs do ESP32 LoRa têm 12 bits, portanto tem valores de 0 até 4095 (o que explica as diferenças na fórmula apresentada).

Alguns refinamentos foram realizados para atenuar o ruído, por média de 100 leituras de tensão, assim obtendo um valor mais próximo do real e com tensão máxima em 4.2V.

Figura 7 - Função readData() que retorna a turbidez em NTUs.

```
/*Função que vai ler os dados do sensor (no caso aqui é um contador)*/
String readData()
{
  double Voltagem = 0.0;
  double Turbidez = 0.0;

  /*Tirando a média da voltagem para diminuir o ruído.*/
  for(int i = 0; i < TOTAL_LEITURAS; i++)
  {
    Voltagem += analogRead(SENSOR_TURBIDEZ) * TENSAO_MAX * CORRECAO_SINAL / RESOLUCAO_ADC;
  }

  Voltagem = Voltagem / TOTAL_LEITURAS;
  /*Correção para turbidez anular em água pura*/

  if(Voltagem > 4.2)
  {
    Voltagem = 4.2;
  }

  Turbidez = (-1) * (1120.4) * Voltagem * Voltagem + 5742.3 * Voltagem + (-1)*4352.9;

  Serial.println(Turbidez);
  return String(Turbidez);
}
```

Fonte: Proprio Autor. 2023.

2.7 Coleta de dados

O sensor foi colocado no lago do Parque Municipal em Botucatu, com o *slave*/*gateway* recebendo alimentação de 5V via power bank e conexão com a internet através de celular com

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

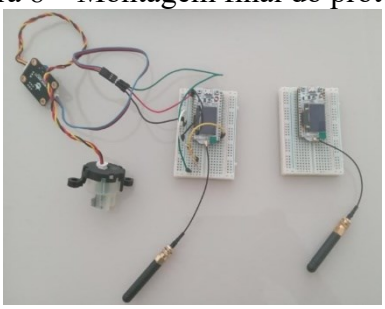
4G. Os dados de turbidez foram coletados por 1h, com intervalo de 36 segundos entre cada envio para a plataforma ThingSpeak, através da qual foram obtidas as visualizações.

3 RESULTADOS E DISCUSSÃO

3.1 Montagem e comunicação master/slave

Para iniciar a codificação no ESP32, inicialmente deve-se instalar o *board* Heltec WiFi LoRa 32(v2), do Boards Manager da Arduino IDE. Se o *board* não for instalado, a Arduino IDE é incapaz de reconhecer o SoC, impossibilitando o desenvolvimento.

Figura 8 – Montagem final do protótipo



Fonte: Proprio Autor. 2023.

A comunicação é iniciada por requisição de envio de dados do sensor que o master faz para o *slave*. A função send() do master é utilizada na requisição a cada 5 segundos como na Figura 9.

Figura 9 - Função send() do master que requisita dados para o slave.

```
282 void loop(){
283
284   /*
285     O GATEWAY pede para o Slave mandar os dados somente se
286     passar um tempo maior que o intervalo determinado (5 segundos)
287     OBS: A função millis() pega o tempo desde que o Lora ligou, então
288     o lastSendTime sempre recebe + 5 segundos a cada requisição de dados.
289   */
290   if (millis() - lastSendTime > INTERVAL)
291   {
292     /*Marcamos o tempo que ocorreu o último envio*/
293     lastSendTime = millis();
294     /*Envia o pacote para informar ao Slave que o GATEWAY quer receber dados*/
295     send();
296   }
297
```

Fonte: Proprio Autor. 2023.

Ao receber a requisição, o *slave* tenta ler o pacote e caso consiga, inicia o armazenamento dos dados enviados em uma *string*. Caso a *string* recebida seja igual a getdata,

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

o *slave* lê os dados do sensor, armazena-os em uma *string*, cria o pacote LoRa e envia os dados de turbidez para o *master* no formato setdata=”dados” (Figura 10).

Figura 10 - Código que retorna a medida de turbidez para o *master*.

```
// Informa o Slave que a que o GATEWAY quer dados
const String GETDATA = "getdata";

// String de retorno para o GATEWAY que contém os dados dos sensores
const String SETDATA = "setdata=";
/*Programa principal*/
void loop()
{
  /*Tenta ler o pacote*/
  int packetSize = LoRa.parsePacket();

  /*Verifica se o pacote possui a quantidade de caracteres que esperamos*/
  if (packetSize == GETDATA.length())
  {
    String received = "";

    /*Armazena os dados do pacote em uma string*/
    while(LoRa.available()){
      received += (char) LoRa.read();
    }

  if(received.equals(GETDATA)){
      /*Simula a leitura dos dados*/
      String dados = readData();
      Serial.println("Criando pacote para envio");

      /*Cria o pacote para envio*/
      LoRa.beginPacket();
      LoRa.print(SETDATA + dados);
      Serial.print("string");
      Serial.println(SETDATA + dados);

      /*Finaliza e envia o pacote*/
      LoRa.endPacket();

```

Fonte: Proprio Autor. 2023.

Ao receber dados do slave, o master avalia o tamanho da string recebida. Se for maior que 7 (tamanho de setdata=), então o slave conseguiu enviar os dados do sensor e ocorrerá uma tentativa de envio do pacote MQTT para o *Broker* (publicação dos dados), conforme ilustra a Figura 11.

Figura 11 - Código no master que recebe os dados do *slave* e publica no *Broker* MQTT.

```
void loop(){

  /*
    O GATEWAY pede para o Slave mandar os dados somente se
    passar um tempo maior que o intervalo determinado (5 segundos)
    OBS: A função millis() pega o tempo desde que o Lora ligou, então
    o lastSendTime sempre recebe + 5 segundos a cada requisição de dados.
  */
  if (millis() - lastSendTime > INTERVAL)
  {
    /*Marcamos o tempo que ocorreu o último envio*/
    lastSendTime = millis();
    /*Envia o pacote para informar ao Slave que o GATEWAY quer receber dados*/
    send();
  }

```

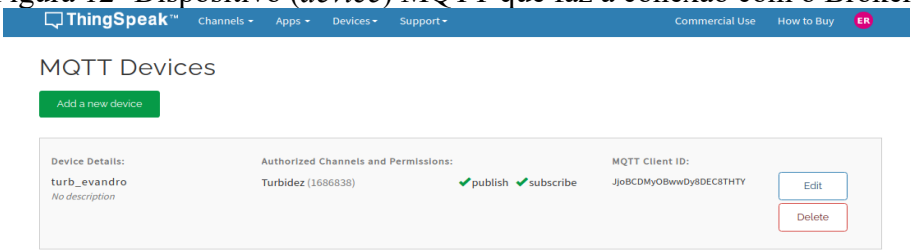
Fonte: Proprio Autor. 2023.

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

3.2 Envio dos dados do master para o Broker MQTT

A conexão com o Broker MQTT depende da criação de um MQTT *Device* no ThingSpeak, que fornecerá um Client ID, UserName e Password que são necessários para a criação da conexão com a plataforma, conforme mostra a Figura 12.

Figura 12- Dispositivo (*device*) MQTT que faz a conexão com o Broker.



Fonte: Proprio Autor. 2023.

As credenciais do MQTT *Device* são utilizadas para a criação da conexão MQTT com o *Broker*, por meio do método connect do objeto MQTT, conforme mostra a Figura 13.

Figura 13 - Função que cria a conexão com o Broker MQTT.

```
/* Função: conecta-se ao broker MQTT (se não há conexão já ativa)
 * Parâmetros: nenhum
 * Retorno: nenhum
 */
void connect_MQTT(void)
{
    while (!MQTT.connected())
    {
        Serial.print("* Tentando se conectar ao seguinte broker MQTT: ");
        Serial.println(broker_mqtt);

        if (MQTT.connect(clientID, mqttUserName, mqttPass, "", 0, false,"" ,true))
        {
            Serial.println("Conexao ao broker MQTT feita com sucesso!");

            display.setCursor(0, OLED_LINE4);
            display.print("Conectado ao broker!");
            display.display();
        }
        else
        {
            display.setCursor(0, OLED_LINE4);
            display.print("Falha ao se conectar ao broker MQTT!");
            display.display();
        }
    }
}
```

Fonte: Proprio Autor. 2023.

Como visto anteriormente, os dados são enviados para o *Broker* por meio da função send_mqtt(char* payload), na qual o parâmetro payload contém a mensagem que será enviada. Além de fazer o envio dos dados, essa função também é responsável por reestabelecer a conexão com o *WiFi* e o *Broker*, para o caso de perda de conexão. O código da função send_mqtt(char* payload) pode ser visto na Figura 14. O tópico de publicação deve ser no formato

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

"channels/<channel_id>/publish/fields/ <field_thingspeak>", onde channel_id é o id do canal no ThingSpeak que produz a visualização dos dados.

Figura 14 - Função que envia os dados para o Broker MQTT do ThingSpeak.

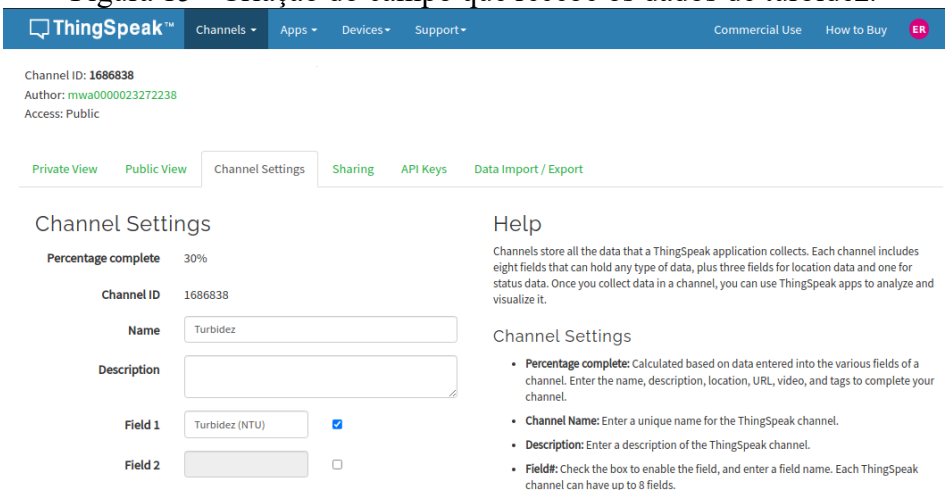
```
267 void send_mqtt(char* payload)
268 {
269     /* Verifica se a conexão wi-fi está ativa e,
270     em caso negativo, reconecta-se ao roteador */
271     verify_wifi_connection();
272     /* Verifica se a conexão ao broker MQTT está ativa e,
273     em caso negativo, reconecta-se ao broker */
274     connect_MQTT();
275
276     /* Envia a frase "Comunicacao MQTT via ESP32" via MQTT */
277     MQTT.publish(MQTT_PUB_TOPIC, payload);
278     display.setCursor(0, OLED_LINE5);
279     display.print("Dados: ");
280     display.print(payload);
281     display.display();
282 }
```

Fonte: Proprio Autor. 2023.

3.3 Visualização dos dados no ThingSpeak

Para que os dados sejam visualizados no ThingSpeak, é necessária a criação do canal que produzirá as visualizações dos dados. Este se conecta ao MQTT *Device*, que faz a subscrição no tópico publicado. Além disso, no canal deve-se criar um campo (field1) que será utilizado no eixo y da visualização, conforme mostra a Figura 15.

Figura 15 - Criação do campo que recebe os dados de turbidez.

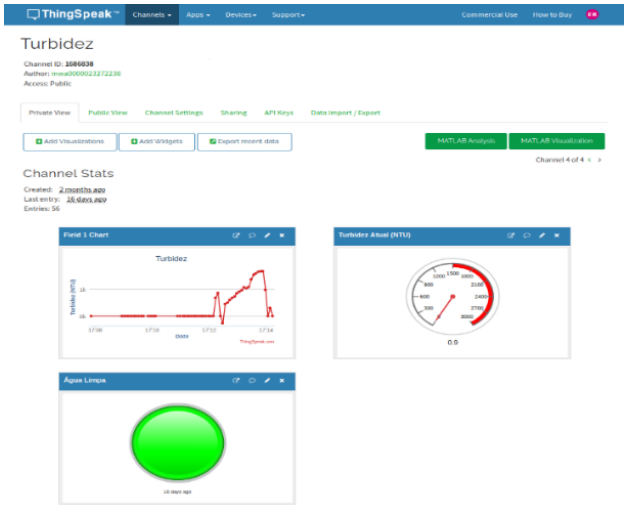


Fonte: Proprio Autor. 2023.

Com o canal criado, foram adicionadas visualizações e observadas as mudanças de turbidez ao longo do tempo. Entre as visualizações (Figura 16), existe um gráfico da turbidez em função do tempo e dois indicadores do valor de turbidez nos últimos 5 segundos. Para o indicador de água limpa, se a medida for superior a 50 NTU, a lâmpada verde é apagada.

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

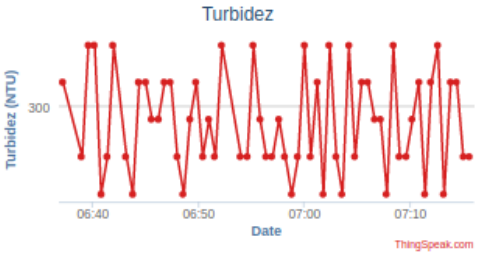
Figura 16 - Visualização dos dados de turbidez.



Fonte: Proprio Autor. 2023.

Com os dados da coleta de turbidez do lago, foi possível construir o gráfico da Figura 13, onde o valor médio de turbidez foi de 274,92 NTU. A turbidez da água para consumo deve ser entre 0 e 3 NTU (ALENCAR *et al*, 2020), a da água subterrânea típica é menor que 1,0 NTU e a turbidez para psicultura está entre 10 NTU e 40 NTU. Com esses valores (Figura 17), fica claro que a água de um lago qualquer não deveria ser utilizada para consumo. Para obter uma turbidez menor, é necessário fazer a filtragem da água, para separar as partículas sólidas e orgânicas do líquido, reduzindo a turbidez e aumentando a qualidade da água.

Figura 17 - Turbidez da água do lado coletada durante 1h.



Fonte: Proprio Autor. 2023.

4 CONCLUSÕES

A tecnologia LoRa oferece uma combinação de alcance estendido, baixo consumo de energia, boa penetração e custo efetivo, sendo atraente para aplicações de comunicação sem fio em IoT e monitoramento remoto. Embora tenha-se obtido resultados coerentes, o protótipo apresentou deficiências, principalmente na calibração do sensor de turbidez. Em valores muito

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

baixos de turbidez, onde a voltagem se aproxima de 4,2V, há grande variação nas medidas. Isso ocorre devido à variação forte da função quadrática utilizada para calcular a turbidez em valores próximos a 4,2V, que diminui conforme a voltagem diminui. Outro problema é a perda de calibração do sistema após períodos de inatividade.

Para obter medidas mais confiáveis, recomenda-se utilizar um sensor mais preciso e estável em futuras pesquisas. É possível ampliar a quantidade de ESP32 LoRa para obter dados de mais sensores, utilizando o mesmo código nos dispositivos e implementando um sistema de filas para que o *gateway* receba e envie mensagens para os dispositivos.

REFERÊNCIAS

ALENCAR, G. R. R. de *et al.* C. Controle de qualidade de Águas Minerais / Quality control of Mineral Waters. **Brazilian Journal of Health Review**, [S. l.], v. 3, n. 6, p. 16356–16368, 2020. DOI: 10.34119/bjhrv3n6-059. Disponível em: https://ojs.brazilianjournals.com .br/ojs/index.php/BJHR/article/view/19950. Acesso em: 16 maio. 2024.

ANDRADE, G. *et al.* Abordagem iWater: uma Contribuição ao Monitoramento da Barragem Santa Bárbara no Cenário da IoT. In: WORKSHOP DE COMPUTAÇÃO APLICADA À GESTÃO DO MEIO AMBIENTE E RECURSOS NATURAIS (WCAMA), 13., 2022, Niterói. **Anais [...]**. Porto Alegre: Sociedade Brasileira de Computação, 2022 . p. 81-90. ISSN 2595-6124. DOI: https://doi.org/10.5753/wcama.2022.222865. Acesso em: 16 maio. 2024.

ÁVILA, F. G. *et al.*. A comparative study of water quality using two quality indices and a risk index in a drinking water distribution network. **Environmental Technology Reviews**, v. 11, p. 49–61, 2022. Disponível em: https://www.tandfonline.com/doi/epdf/10.10 80/21622515.2021.2013955?needAccess=true. Acesso em: 6 maio 2024.

BORAS, C.; *et al.* Using LoRa Technology for IoT Monitoring Systems, In: 10th International Conference on Networks of the Future (NoF), 2019, **Anais [...]**. Rome. IEEExplore, Rome, IEEE: 3 out 2019. p. 134-137. doi: 10.1109/NoF47743.2019.9014994. Acesso 2 maio 2024.

DE OLIVEIRA JUNIOR, M. A.; DE LIMA SILVA, N. C.; FERREIRA PRADO, M. Qualidade da água consumida em bebedouros no IFTM – campus Uberlândia. **Revista Meio Ambiente e Sustentabilidade**, [S. l.], v. 12, n. 25, p. 3–16, 2023. DOI: 10.22292/mas.v12i25.1115. Disponível em:https://www.revistasuninter.com/revistameioamb iente/index.php/meioAmbiente/article/view/1115. Acesso em: 16 maio. 2024.

DFRobot. **DFRobot Wiki**, 2024. DFRobot Open Source Hardware Eletronics and Kits. Disponível em: https://abrir.link/OXCRW. Acesso em: 22 maio 2024.

FÉLIX, E. A. *et al.* Índice de Qualidade da Água (IQA) Como Indicador de Vulnerabilidade Ambiental na Bacia Hidrográfica o Rio Cabaçal – MT. **Revista Geoaraguaia**, [S. l.], v. 13, n. Especial, p. 87–106, 2023. Disponível em: https://periodicoscientificos.ufmt.br/ojs/index.php/geo/article/vi ew/15276. Acesso em: 15 maio 2024.

Tekhne e Logos, Botucatu, SP, v.16, n.1, maio, 2025.
ISSN 2176 – 4808

JAEGER, W. K. *et al.* et. al.. Finding water scarcity amid abundance using human–natural system models. **Proceedings of the National Academy of Sciences**, v. 114, n.45, p. 11884–11889, 2017. https://doi.org/10.1073/pnas.1706847114. Acesso 4 maio 2024.

JUNG, M. S.; DA SILVA *et al.* A Fundamental Resource for Ensuring Sustainability. **Revista de Gestão Social e Ambiental**, São Paulo (SP), v. 17, n. 7, p. e03661, 2023. DOI: 10.24857/rgsa.v17n7-013. Disponível em: https://rgsa.openaccesspublications.org/rgsa/article/view/3661. Acesso em: 13 maio 2024.

MELO, J. de J.; MOURA DIA, M. J. .; OLIVEIRA, A. U. de . A água e sua proteção legal no brasil e acre: considerações sobre a legislações dos recursos hidrícos. Uáquiri - **Revista do Programa de Pós-Graduação em Geografia da Universidade Federal do Acre**, [S. l.], v. 4, n. 1, 2022. DOI: 10.29327/268458.4.1-5. Disponível em: https://periodicos.ufac.br/index.php/Uaquiri/article/ view/5059. Acesso em: 14 maio. 2024.

SASAKI, Y.; YOKOTANI, T. Performance evaluation of MQTT as a communication protocol for IoT and prototyping. **Advances in Technology Innovation**, v. 4, n. 1, p. 21-29, 2019. Disponível em: https://core.ac.uk/download/pdf/228834689.pdf. Acesso em: 22 maio 2024.

SPOHN, M. A.; GENERO, W. B. Análise experimental dos protocolos MQTT e MQTT-SN. **Revista Brasileira de Computação Aplicada**, [S. l.], v. 15, n. 1, p. 22-33, 2023. DOI: 10.5335/rbca.v15i1.13510. Disponível em:https://seer.upf.br/index.php/rbca/article/view/13510. Acesso em: 22 maio 2024.

WANG, Y. *et al.* Low-Cost Turbidity Sensor for Low-Power Wireless Monitoring of Fresh-Water Courses, **IEEE Sensors Journal**, v. 18, n. 11, p. 4689-4696, 1 Jun 2018. Acesso em: 2 maio 2024.