## ANEXO 1 - CÓDIGO *GATEWAY*

```
// Importa a biblioteca Wi-Fi
#include <WiFi.h>

// Bibliotecas para LoRa
#include <SPI.h>
#include <LoRa.h>

// Bibliotecas para Display OLED
#include <Wire.h>
#include <Adafruit_GFX.h>
#include <Adafruit_SSD1306.h>

// Define os pinos usados pelo LoRa
#define SCK 5
#define MISO 19
#define MOSI 27
#define SS 18
#define RST 14
#define DIO0 26

#define BAND 915E6 // Banda de frequência utilizada para LoRa

// Pinos do OLED
#define OLED_SDA 4
#define OLED_SCL 15
#define OLED_RST 16
#define SCREEN_WIDTH 128 // Largura do display OLED, em pixels
#define SCREEN_HEIGHT 64 // Altura do display OLED, em pixels

// Substitua com suas credenciais de rede
String apiKey = "MQNN7IQTQN190QOY"; // Insira sua chave API de escrita do
ThingSpeak
const char *ssid = "TCCARTSARAH"; // Substitua pelo SSID e senha do seu Wi-Fi
const char *password = "tccarthursarah";
const char* server = "api.thingspeak.com";

WiFiClient client;

// Inicializa variáveis para obter e salvar dados do LoRa
int rssi;
String loRaMessage;
String temperature;
String ntu;
String phValue;
```

```
String OD;
String readingID;

Adafruit_SSD1306 display(SCREEN_WIDTH, SCREEN_HEIGHT, &Wire,
OLED_RST);

// Substitui o placeholder pelos valores do sensor
String processor(const String& var) {
  // Serial.println(var);
  if(var == "TEMPERATURE") {
    return temperature;
  }
  else if (var == "NTU") {
    return ntu;
  }
  else if (var == "PH") {
    return String(phValue);
  }
  else if (var == "OD") {
    return String(OD);
  }
  else if (var == "RRSI") {
    return String(rssi);
  }
  return String();
}

// Inicializa o display OLED
void startOLED() {
  // Reinicia o display OLED via software
  pinMode(OLED_RST, OUTPUT);
  digitalWrite(OLED_RST, LOW);
  delay(20);
  digitalWrite(OLED_RST, HIGH);

  // Inicializa o OLED
  Wire.begin(OLED_SDA, OLED_SCL);
  if(!display.begin(SSD1306_SWITCHCAPVCC, 0x3c, false, false)) { // Endereço
0x3C para 128x32
    Serial.println(F("Falha na alocação do SSD1306"));
    for(;;); // Não prossiga, loop infinito
  }
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0,0);
  display.print("ENVIO LORA");
```

```
}

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

  int counter = 0; // Inicializa o contador
  SPI.begin(SCK, MISO, MOSI, SS);

  // Configura o módulo transceptor LoRa
  LoRa.setPins(SS, RST, DIO0); // Configura o módulo LoRa

  while (!LoRa.begin(BAND) && counter < 10) {
    Serial.print(".");
    counter++;
    delay(2000);
  }
  if (counter == 10) {
    // Incrementa o readingID em cada nova leitura
    Serial.println("Falha na inicialização do LoRa!");
  }
  Serial.println("Inicialização LoRa OK!");
  delay(2000);

  // Conecta à rede Wi-Fi com SSID e senha
  Serial.print("Conectando a ");
  Serial.println(ssid);
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(2000);
    Serial.print(".");
  }
  // Imprime o endereço IP local e inicia o servidor web
  Serial.println("");
  Serial.println("WiFi conectado.");
  Serial.println("Endereço IP: ");
  Serial.println(WiFi.localIP());
}

// Lê o pacote LoRa e obtém as leituras dos sensores
void loop() {
  int packetSize = LoRa.parsePacket();
  if (packetSize) {
    Serial.print("Pacote LoRa recebido: ");
    while (LoRa.available()) { // Lê o pacote
      String LoRaData = LoRa.readString();
      Serial.print("Temperatura:");
```

```
  Serial.print(temperature);
  Serial.print("Turbidez:");
  Serial.print(ntu);
  Serial.print("PH:");
  Serial.print(phValue);
  Serial.print("OD:");
  Serial.print(OD);
  Serial.print("RSSI:");
  Serial.println(rssi);

  // Extrai os dados do pacote LoRa
  int pos1 = LoRaData.indexOf('/');
  int pos2 = LoRaData.indexOf('&');
  int pos3 = LoRaData.indexOf('$');
  int pos4 = LoRaData.indexOf('%');
  readingID = LoRaData.substring(0, pos1);
  temperature = LoRaData.substring(pos1+1, pos2);
  ntu = LoRaData.substring(pos2+1, pos3);
  phValue = LoRaData.substring(pos3+1, pos4);
  OD = LoRaData.substring(pos4+1, LoRaData.length());
}

rssi = LoRa.packetRssi(); // Obtém o RSSI
Serial.print(" com RSSI ");
Serial.println(rssi);

// Atualiza o display OLED com os dados recebidos
display.clearDisplay();
display.setCursor(0, 0);
display.setTextSize(1);
display.print("Pacote Enviado!");

display.setCursor(0, 20);
display.print("Temperatura:");
display.setCursor(72, 20);
display.print(temperature);
display.display();

display.setCursor(0, 30);
display.print("Turbidez:");
display.setCursor(62, 30);
display.print(ntu);
display.display();

display.setCursor(0, 40);
display.print("PH:");
display.setCursor(42, 40);
```

```
    display.print(phValue);
    display.display();

    display.setCursor(0, 50);
    display.print("OD:");
    display.setCursor(42, 50);
    display.print(OD);
    display.display();

    display.setCursor(0,60);
    display.print("RSSI:");
    display.setCursor(52,60);
    display.print(rssi);
    display.display();

  }

  // Envia os dados para o ThingSpeak
  if (client.connect(server, 80)) {
    String postStr = apiKey;
    postStr += "&field1=";
    postStr += String(temperature);
    postStr += "&field2=";
    postStr += String(ntu);
    postStr += "&field3=";
    postStr += String(phValue);
    postStr += "&field4=";
    postStr += String(OD);
    postStr += "\r\n\r\n\r\n\r\n";

    client.print("POST /update HTTP/1.1\n");
    client.print("Host: api.thingspeak.com\n");
    client.print("Connection: close\n");
    client.print("X-THINGSPEAKAPIKEY: " + apiKey + "\n");
    client.print("Content-Type: application/x-www-form-urlencoded\n");
    client.print("Content-Length: ");
    client.print(postStr.length());
    client.print("\n\n");
    client.print(postStr);
  }
  //delay(30000); // Adicione um atraso se necessário
}
```