ANEXO 2 - CÓDIGO EMBARCADO NO DISPOSITIVO COLETOR DE DADOS DOS SENSORES

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

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

// Bibliotecas para o Sensor
#include <OneWire.h>
#include <EEPROM.h>

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

// Definir a frequência de operação
#define BAND 866E6

// Pinos do OLED
#define OLED_SDA 21
#define OLED_SCL 22
#define OLED_RST 23
#define SCREEN_WIDTH 128  // Largura do display OLED
#define SCREEN_HEIGHT 64  // Altura do display OLED

// Definição do sensor de temperatura
int DS18S20_Pin = 2;
OneWire ds(DS18S20_Pin);

// Definição para sensor de turbidez
float ntu;
float voltaget;

// Definição para sensor de pH
#define PH_PIN A4
float voltage;
float phValue;
```

```
float acidVoltage = 1900;
float neutralVoltage = 1450;
float slope, intercept;

// Definição para sensor de OD (Oxigênio Dissolvido)
#define DO_PIN A7
#define VREF 5000    // Tensão de referência (mv)
#define ADC_RES 4096 // Resolução do ADC
#define TWO_POINT_CALIBRATION 0
#define READ_TEMP (25) // Temperatura atual da água ℃ ou função de sensor de temperatura
#define CAL1_V (1200) // Tensão de calibração 1 (mv)
#define CAL1_T (25)   // Temperatura de calibração 1 (℃)
#define CAL2_V (1300) // Tensão de calibração 2 (mv)
#define CAL2_T (15)   // Temperatura de calibração 2 (℃)

// Tabela de DO (Oxigênio Dissolvido)
const uint16_t DO_Table[41] = {
    14460, 14220, 13820, 13440, 13090, 12740, 12420, 12110, 11810, 11530,
    11260, 11010, 10770, 10530, 10300, 10080, 9860, 9660, 9460, 9270,
    9080, 8900, 8730, 8570, 8410, 8250, 8110, 7960, 7820, 7690,
    7560, 7430, 7300, 7180, 7070, 6950, 6840, 6730, 6630, 6530, 6410};

uint8_t Temperaturet;
uint16_t ADC_Raw;
uint16_t ADC_Voltage;
uint16_t DO;
float OD;

// Função para leitura de OD
int16_t readDO(uint32_t voltage_mv, uint8_t temperature_c) {
  #if TWO_POINT_CALIBRATION == 0
    uint16_t V_saturation = (uint32_t)CAL1_V + (uint32_t)35 * temperature_c - (uint32_t)CAL1_T * 35;
    return (voltage_mv * DO_Table[temperature_c] / V_saturation);
  #else
    uint16_t V_saturation = (int16_t)((int8_t)temperature_c - CAL2_T) * ((uint16_t)CAL1_V - CAL2_V) / ((uint8_t)CAL1_T - CAL2_T) + CAL2_V;
    return (voltage_mv * DO_Table[temperature_c] / V_saturation);
  #endif
}

// Contador de pacotes
int readingID = 0;
int counter = 0;
String LoRaMessage = "";
```

```
float temperature = 0;

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

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

  // Inicializar 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 continuar, entrar em loop infinito
  }
  display.clearDisplay();
  display.setTextColor(WHITE);
  display.setTextSize(1);
  display.setCursor(0, 0);
  display.print("LORA TRANSMISSOR");
}

// Inicializar o módulo LoRa
void startLoRA() {
  // Configurar pinos SPI para o LoRa
  SPI.begin(SCK, MISO, MOSI, SS);
  // Configurar o transceptor LoRa
  LoRa.setPins(SS, RST, DIO0);

  while (!LoRa.begin(BAND) && counter < 10) {
    Serial.print(".");
    counter++;
    delay(500);
  }
  if (counter == 10) {
    // Incrementar o ID de leitura a cada nova tentativa
    readingID++;
    Serial.println("Falha ao iniciar LoRa!");
  }
  Serial.println("Inicialização LoRa OK!");
  display.setCursor(0, 10);
```

```
  display.clearDisplay();
  display.print("Inicialização OK!");
  display.display();
  delay(2000);
}

void loop() {
  /// TEMPERATURA ///
  float temperature = getTemp();
  Serial.println(temperature);

  /// PH ///
  slope = (7.0 - 4.0) / ((neutralVoltage - 1500) / 3.0 - (acidVoltage - 1500) / 3.0);
  intercept = 7.0 - slope * (neutralVoltage - 1500) / 3.0;
  static unsigned long timepoint = millis();
  if (millis() - timepoint > 1000U) {  // Intervalo de tempo: 1s
    timepoint = millis();
    voltage = analogRead(PH_PIN) / 4095.0 * 3300;  // Ler a tensão
    phValue = slope * (voltage - 1500) / 3.0 + intercept;  // Converter a tensão para
pH com compensação de temperatura
    Serial.print("\t pH: ");
    Serial.println(phValue, 2);
  }

  /// OD (Oxigênio Dissolvido) ///
  Temperaturet = (uint8_t)READ_TEMP;
  ADC_Raw = analogRead(DO_PIN);
  ADC_Voltage = uint32_t(VREF) * ADC_Raw / ADC_RES;
  Serial.println("OD:\t" + String(readDO(ADC_Voltage, Temperaturet) / 1000));

  /// TURBIDEZ ///
  int sensorValue = analogRead(A5);  // Ler o valor no pino analógico
  voltaget = sensorValue * (3.3 / 1453.7);  // Converter para tensão
  if (voltaget < 2.5) {
    ntu = 3000;
  } else if (voltaget > 4.2) {
    ntu = 0;
  } else {
    ntu = (-1120.4 * (voltaget * voltaget)) + (5742.3 * voltaget) - 4352.9;
  }
  delay(100);

  ///// LoRa: Display e Envio /////
  LoRaMessage = String(readingID) + "/" + String(temperature) + "&" + String(ntu) +
"$" + String(phValue) + "%" + String((readDO(ADC_Voltage, Temperaturet)) / 100);
  // Enviar pacote LoRa para o receptor
  LoRa.beginPacket();
```

```
  LoRa.print(LoRaMessage);
  LoRa.endPacket();

  // Atualizar o display
  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(String(readDO(ADC_Voltage, Temperaturet) / 100));
  display.display();

  Serial.print("Enviando pacote: ");
  Serial.println(readingID);
  readingID++;
  delay(1000);
}

// Função para obter a temperatura do sensor DS18S20
float getTemp() {
  byte data[12];
  byte addr[8];

  if (!ds.search(addr)) {
    // Sem mais sensores na cadeia, reiniciar a busca
    ds.reset_search();
```

```
    return -1000;
  }

  if (OneWire::crc8(addr, 7) != addr[7]) {
    Serial.println("CRC inválido!");
    return -1000;
  }

  if (addr[0] != 0x10 && addr[0] != 0x28) {
    Serial.print("Dispositivo não reconhecido");
    return -1000;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44, 1);  // Iniciar conversão com alimentação parasita

  byte present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);  // Ler Scratchpad

  for (int i = 0; i < 9; i++) {  // Necessário 9 bytes
    data[i] = ds.read();
  }

  ds.reset_search();

  byte MSB = data[1];
  byte LSB = data[0];

  float tempRead = ((MSB << 8) | LSB);  // Usar complemento de dois
  float TemperatureSum = tempRead / 16;

  return TemperatureSum;
}

void setup() {
  // Inicializar o monitor serial
  Serial.begin(115200);
  startOLED();
  startLoRA();
}
```