```
/* TCC - Medidor de consumo elétrico - PROJETO 8
/*
   Testes de utilização do ESP32 com o sensor PZEM
*/

// --------------------- RECEIVER --------------------------

// --------------------- Bibliotecas utilizadas -------------------------

#include <WiFi.h>
#include  <WiFiClient.h>
#include  <BlynkSimpleEsp32.h>
#include <SPI.h> //responsável pela comunicação serial
#include <LoRa.h> //responsável pela comunicação com o Lora
#include "SSD1306.h" //responsável pela comunicação com o display
#include <Wire.h>  //responsável pela comunicação i2c
#include <LiquidCrystal_I2C.h> // Biblioteca do LCD
#include <HardwareSerial.h> // Hardware do PZEM

#define BLYNK_PRINT Serial

// Definição das variáveis do APP Blynk
  float voltage_blynk=0;
  float current_blynk=0;
  float power_blynk=0;
  float energy_blynk=0;
  float consumo_blynk=0;
  float custo_blynk=0;
  float falta_blynk=0;
  float limite = 0 ;

// Colocar o Auth Token disponível no Blynk App
/* Inserir o "Auth Token" obtido no Blynk App. Vá até "Project Settings"
(ícone parafuso),
* entre "" a seguir.
* É enviado ao e-mail cadastrado ou pode ser copiado para área
transferência SmartPhone.
* WiFi: Fazer o NodeMCU Lolin conectado na Rede,
* SSID e Senha devem ser escritos entre respectivas "" a seguir.
*/
  char auth[] = "ReTdIP_oDz5MDJ37-vQ3GLTrtRfV0DDM";

// Credenciais da Wifi Local
  char ssid[] = "blynk";          //colocar o ssid da rede
  char pass[] = "blynkteste";          //colocar a senha da sua rede
```

```
BLYNK_WRITE(V10)
{
  float pinValue = param.asFloat(); // assigning incoming vale from pin V1
to a variable
   Serial.print("Slider:");
   Serial.println(pinValue);
  // process received value
  limite = pinValue;
  Blynk.virtualWrite(V11, 0);
}
/*============================================================================
=======================================
* DEFIINIÇÃO DAS VARIÁVEIS DO LORA
*/
// Definição dos pinos
  #define SCK     5    // GPIO5  -- SX127x's SCK
  #define MISO    19   // GPIO19 -- SX127x's MISO
  #define MOSI    27   // GPIO27 -- SX127x's MOSI
  #define SS      18   // GPIO18 -- SX127x's CS
  #define RST     14   // GPIO14 -- SX127x's RESET
  #define DI00    26   // GPIO26 -- SX127x's IRQ(Interrupt Request)

// Frequência do radio - podemos utilizar ainda : 433E6, 868E6, 915E6
  #define BAND    915E6
  #define PABOOST true

// Parâmetros: address,SDA,SCL
  SSD1306 display(0x3c, 4, 15); //construtor do objeto que controlaremos o
display
  String rssi = "RSSI --";
  String packSize = "--";
  String packet ;

// --------------------- LCD ---------------------------

LiquidCrystal_I2C lcd(0x27,16,2); // set the LCD address to 0x3F for a 16
chars and 2 line display
// Entradas do LCD
#define SDA1 21
#define SCL1 22

/*============================================================================
=======================================
* SETUP, INICIALIZAÇÃO DAS VARIÁVEIS, FUNÇÕES E BIBLIOTECAS
*/
```

```
void setup() {
  Serial.begin(115200); // Inicialização da comunicação serial

//Setup OLED
  pinMode(16,OUTPUT); //RST do oled
  digitalWrite(16, LOW);    // Reseta o OLED
  delay(50);
  digitalWrite(16, HIGH); // Enquanto o OLED estiver ligado, GPIO16 deve
estar HIGH



  delay(1500);


// Setup LCD
    Serial.println("Inicialização LCD");
    lcd.begin();

// Indica no display que inicilizou corretamente.



  delay(1000);

// Setup LoRa

  SPI.begin(SCK,MISO,MOSI,SS); //inicia a comunicação serial com o Lora
  LoRa.setPins(SS,RST,DI00); //configura os pinos que serão utlizados pela
biblioteca (deve ser chamado antes do LoRa.begin)

// Debug Lora
  Serial.println("Iniciando LoRa ...");            //Acompanhamento no seria



  delay(1000);

// Inicializa o Lora com a frequência específica.
  if (!LoRa.begin(BAND)) // PABOOST não funciona
  {
     Serial.println("Inicialização do LoRa falhou !");
//Acompanhamento no serial
    while (1);
  }
```

```
  delay(1000);

// Indica no display que inicilizou corretamente.
  Serial.println("LoRa iniciado com sucesso!");            //Acompanhamento
no serial



  lcd.clear();
  lcd.setCursor(0,0);
  lcd.print("LoRa Iniciado");
  delay(1000);

//LoRa.onReceive(cbk);
  LoRa.receive(); // Habilita o Lora para receber dados

//NTP - Contagem do tempo
//  timerun = millis();

// Início do APP Blynk
  Serial.println("Iniciando Blink...");



  delay(1000);
  Blynk.begin(auth, ssid, pass); //Faz a conexão com o Blynk
  Serial.println("Blink Iniciado!");
  lcd.setCursor(0,1);
  lcd.print("Blynk Iniciado");



  delay(500);
// Início do APP Blynk
  Blynk.begin(auth, ssid, pass); //Faz a conexão com o Blynk
// Inicialização do Blynk
  Blynk.run();
  Serial.println("Blink run...");



  delay(1000);

}

/*=========================================================================
```

```
=======================================
* LAÇO DE EXECUÇÃO DAS ROTINAS
*/

void loop() {

// Inicialização do Blynk
  Blynk.run();


//parsePacket: checa se um pacote foi recebido
//retorno: tamanho do pacote em bytes. Se retornar 0 (ZERO) nenhum pacote
foi recebido
  int packetSize = LoRa.parsePacket();
//caso tenha recebido pacote chama a função para configurar os dados que
serão mostrados em tela
  if (packetSize) {
    cbk(packetSize);
  }
}

//função responsável por recuperar o conteúdo do pacote recebido
//parametro: tamanho do pacote (bytes)
void cbk(int packetSize) {
  packet ="";
  packSize = String(packetSize,DEC); //transforma o tamanho do pacote em
String para imprimirmos
  for (int i = 0; i < packetSize; i++) {
    packet += (char) LoRa.read(); //recupera o dado recebido e concatena na
variável "packet"
  }
  rssi = "RSSI=  " + String(LoRa.packetRssi(), DEC)+ "dB"; //configura a
String de Intensidade de Sinal (RSSI)
  //mostrar dados em tela
  blynk();
}

// ------------------------------------ BLYNK
---------------------------------------
void blynk (){
  Serial.print("Numeric Input Blynk:"); Serial.println(limite);

  int tamanho = packet.length(); // Indica o tamanho total do pacote
recebido/posição do final
   // stringaAProcurar.indexOf(stringProcurada)
  // Verifica qual é o parametro/ o número resultante indica a posição que
```

```
a variável foi achada
  // Se o retorno for -1 que dizer que não foi encontrado
  int teste = packet.indexOf("(V)");
  Blynk.virtualWrite(V9, 0); // apenas preparação pra futuro evento de
falta de energia
  if (teste > -1)
      { // String procurada encontrada
        // Procura 2 posições anteriores ao valor do parametro pois o valor
sempre estará após um : e um espaço (2 posições)
        int doispontos = packet.indexOf(": ");
        // Monta o valor que existe duas posições após o : até a posição ao
final do pacote
        String voltage_blynk = packet.substring(doispontos + 2, tamanho);
        Serial.println("Tensão recebida");
        voltage_blynk.toFloat();
        Serial.println(voltage_blynk);
        // Envia o valor para o Blynk
        Blynk.virtualWrite(V1, voltage_blynk);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("rssi: ");lcd.print(rssi);
        lcd.setCursor(0,1);
        lcd.print("V: ");lcd.print(voltage_blynk);
      }
  teste = packet.indexOf("(A)");
  if (teste > -1)
      {
          int doispontos = packet.indexOf(": ");
          String current_blynk = packet.substring(doispontos + 2, tamanho);
          Serial.println("Corrente recebida");
          current_blynk.toFloat();
          Serial.println(current_blynk);
          Blynk.virtualWrite(V2, current_blynk);
          if (current_blynk.toFloat() == 0){
            Serial.println("Possível falha no disjuntor");
            Blynk.virtualWrite(V15, 255); // Led de indicação da falha do
disjuntor
          }
          else {
            Blynk.virtualWrite(V15, 0);
          }
          lcd.clear();
          lcd.setCursor(0,0);
          lcd.print("rssi: ");lcd.print(rssi);
          lcd.setCursor(0,1);
          lcd.print("I: ");lcd.print(current_blynk);
```

```
    }
teste = packet.indexOf("(W)");
if (teste > -1)
    {
        int doispontos = packet.indexOf(": ");
        String power_blynk = packet.substring(doispontos + 2, tamanho);
        Serial.println("Potência recebida");
        power_blynk.toFloat();
        Serial.println(power_blynk);
        Blynk.virtualWrite(V3, power_blynk);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("rssi: ");lcd.print(rssi);
        lcd.setCursor(0,1);
        lcd.print("P: ");lcd.print(power_blynk);
    }
teste = packet.indexOf("(kWh)");
if (teste > -1)
    {
        int doispontos = packet.indexOf(": ");
        String energy_blynk = packet.substring(doispontos + 2, tamanho);
        Serial.println("Energia recebida");
        energy_blynk.toFloat();
        Serial.println(energy_blynk);
        Blynk.virtualWrite(V4, energy_blynk);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("rssi: ");lcd.print(rssi);
        lcd.setCursor(0,1);
        lcd.print("kWh: ");lcd.print(energy_blynk);
    }
teste = packet.indexOf("(kWh/mês)");
if (teste > -1)
    {
        int doispontos = packet.indexOf(": ");
        String consumo_blynk = packet.substring(doispontos + 2, tamanho);
        Serial.println("Consumo recebido");
        consumo_blynk.toFloat();
        Serial.println(consumo_blynk);
        Blynk.virtualWrite(V5, consumo_blynk);
        lcd.clear();
        lcd.setCursor(0,0);
        lcd.print("rssi: ");lcd.print(rssi);
        lcd.setCursor(0,1);
        lcd.print("kWh/mes: ");lcd.print(consumo_blynk);
    }
```

```
  teste = packet.indexOf("(R$/mês)");
  if (teste > -1)
      {
           int doispontos = packet.indexOf(": ");
           String custo_blynk = packet.substring(doispontos + 2, tamanho);
           Serial.println("Custo mensal recebido");
           custo_blynk.toFloat();
           Serial.println(custo_blynk);
           Blynk.virtualWrite(V6, custo_blynk);
           if (custo_blynk.toFloat() > limite)
           {
              Serial.println("Valor de consumo esperado excedido");
              Blynk.virtualWrite(V11, 255);
           }
           lcd.clear();
           lcd.setCursor(0,0);
           lcd.print("rssi: ");lcd.print(rssi);
           lcd.setCursor(0,1);
           lcd.print("Custo/mes: ");lcd.print(custo_blynk);
      }
  teste = packet.indexOf("Falta de energia");
  if (teste > -1)
      {
           int doispontos = packet.indexOf(": ");
           String falta_blynk = packet.substring(doispontos + 2, tamanho);
           Serial.println("Falta de energia detectada");
           Blynk.virtualWrite(V9, 255); //Enviar 255 pra setar o LED em NL
alto e 0 pra NL baixo
           Blynk.virtualWrite(V1, 0.0); // Atualizar o valor da tensão pra 0
           Blynk.virtualWrite(V2, 0.0); // Atualizar o valor da corrente pra
0
           lcd.clear();
           lcd.setCursor(0,0);
           lcd.print("rssi: ");lcd.print(rssi);
           lcd.setCursor(0,1);
           lcd.print("Falta de energia");
      }
}

// EVENTOR -- Colocar o Push Notification para notificação no celular
  // Utilizar o V6 = Custo mensal para definir o limite do consumo mensal
  // Utilizar o V9 = Falta de Energia para informar ao cliente uma possível
falta de energia no local

// TESTAR O WIDGET RTC
  //
```

https://github.com/blynkkk/blynk-library/blob/master/examples/Widgets/RTC/RTC.ino