```
/*****************************************************************************************************************************
* This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General    *
* Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.  *
* This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied   *
* warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more  *
* details. You should have received a copy of the GNU Lesser General Public License along with this program.          *
* If not, see <http://www.gnu.org/licenses/> or <http://blog.robotica.eng.br/gnu-lesser-general-public-license/>.      *
*                                                                                                                     *
* © Copyright 2019-2021 Filippo Pardini - filippo@robotica.eng.br - http://blog.robotica.eng.br/                      *
*****************************************************************************************************************************/
/*
  Projeto CoMoSisFog V5.0.1 (15/08/2021)
  * Projeto para Controle e Monitoramento de um Sistema Fotovoltaico off grid para iluminação de um jardim

  * O sistema é constituído por 8 painéis “Canadian CS6K-270” de 270Wp cada. Eles estão interligados de forma a
  * estabelecer dois conjuntos iguais de 4 painéis. Cada conjunto tem cada 2 painéis interligados em série e por sua
  * vez interligados em paralelo. Cada conjunto alimenta um controlador “Controlador de Carga MPPT Epsolar Tracer-4210A
  * 40A 12/24V” através de um stringbox de proteção. Os 2 controladores fazem a carga de um banco de baterias
  * constituído de 6 baterias estacionarias “Heliar Freedom DF4100” de 240Ah cada. Estas baterias estão interligadas
  * de forma a obter um banco de 24V (dois conjuntos (cj1 e cj2) em série de três baterias em paralelo cada um,
  * balanceados por equalizador). Os controladores estão interligados ao banco de baterias através do stringbox de
  * proteção de forma invertida entre eles (um controlador alimenta o + do cj1 e o – do cj2 e o outro controlador
  * alimenta o – do cj1 e o + do cj2). Ao banco de baterias está ligado, via um disjuntor DC de 80A, um “Inversor
  * Senoidal Epsolar SHI2000-22 – 2000VA / 24Vcc / 220Vca” que alimenta os refletores LED do jardim (610W) em 220Vca
  * através de fusíveis de proteção. O controle e monitoramento deste sistema é feito através de duas Partes:
  * Parte 2 - um “controlador dos refletores” e Parte 1 - um “monitor dos parâmetros elétricos de carga e descarga do
  * banco de baterias”. O controlador e o monitor estão integrados no projeto CoMoSisFog que usa técnicas de IOT.

  Parte 1 - Quiosque (onde estão os painéis solares)

  ESP32 local - monitor
  * Esta parte utiliza uma placa DOIT ESP32 DEVKIT V1, display Oled, um RTC DS3231, uma eeprom 24C32 - 4K que vem com o
  * RTC DS3231, dois expansores I2C PCF8574, 3 placas de 4 relés com isolação ótica, 1 relé com isolação ótica,
  * uma fotocélula, 3 sensores INA226, um Xbee, um botão NA, um micro SD e um conversor de tensão 12Vdc/5Vdc.
  * Ela faz o monitoramento dos tres INA226 (INA1 - carga do banco de baterias via Tracer1, INA2 - carga do banco de
  * baterias via Tracer2, INA3 - descarga do banco de baterias via inversor) usando o RTC DS3231M, mostrando as
  * leituras atuais de corrente, voltagem, potência e energia armazenada e consumida no display Oled e enviando,
  * via Xbee, para o ESP32 da Parte 2. Grava no SD as voltagens máximas e mínimas das INAs e energia gerada e consumida
  * em 24h (para análise e calibração de ciclos liga/desliga). Em função desses dados, da fotocélula, dos dados
  * configurados pelo ESP32 remoto (Parte 2 via Xbee), controla e disponibiliza energia, em horários pré-configurados,
  * para os refletores, executa comandos e solicitações do ESP32 remoto e gerencia a energia fornecida pelas baterias.

  Parte 2 - Laboratório (na residência)

  ESP32 remoto - controlador
  * Esta parte utiliza uma placa DOIT ESP32 DEVKIT V1, uma eeprom At24c256 - 32KB, um Nanoshield ADC circuitar, um
  * display LCD03 com keypad, um buzzer, dois botões NA, um integrado MC1449 (debouncer), um Xbee, uma fonte 5Vdc
  * com bateria ion-litio 3S e uma micro-ventoinha 12Vdc. Ela atua como entrada de configurações, solicitações e
  * comandos, via keypad, para a parte 1 via Xbee. Faz o display, no LCD03, dos dados enviados pela parte 1 e comanda
  * os grupos de refletores pelo keypad.
*/

//*****************************************************************************************************************************
//************************************** Este é o sketch da Parte 2 ***********************************************************
//***************************************************************************************************************** Bibliotecas

// Bibliotecas
#include <Wire.h>                                       // I2C
#include <LCD03_N1.h>                                   // LCD03
#include <elapsedMillis.h>                              // Intervalos de tempo
#include <Nanoshield_ADC.h>                             // Nanoshield ADC ADS1115

//_____________________________________________________________________________________________________________ Variaveis e constantes

// Variaveis e constantes
#define SDApin 21                                       // I2C SDA
#define SCLpin 22                                       // I2C SCL
#define RX2pin 16                                       // Pino RX2 Serial2
#define TX2pin 17                                       // pino TX2 Serial2
#define liga_desliga_tudo 26                            // Pino de interrupt para ligar ou desligar tudo
#define keypadpin 25                                    // Pino de interrupt para o keypad
#define buzzer 27                                       // Pino para acionamento do buzzer
#define limpa_lcd clear_line(1);clear_line(2);clear_line(3);    // Limpa linhas 1.2.3 do LCD
#define timeout 40                                      // Timeout em segundos


char entrada[50];                                       // Vetor para receber digitação no keypad
unsigned long key = 2863311530;                         // Código eeprom válida
volatile bool pedido_keypad;                            // Variável para interrupt do keypad
volatile bool liga_desliga;                             // Variável para o interrupt de liga/desliga tudo
bool tudo_ligado;                                       // Variável de status
String versao = "V5.0.1";                                 // ************ Versão ************************** Versão

unsigned int dia;
unsigned int mes;
unsigned int ano;
unsigned int horas;
```

```
unsigned int minutos;
unsigned int segundos;

float vina1,cina1,pina1,vina2,cina2,pina2,vina3,cina3,pina3,etracer1,etracer2,ein,einversor,eout,volt;

String tempo,str,msg,tmp;

//______________________________________________________________________________________________________________________ Instâncias

elapsedMillis timeElapsed1;                                     // Instancia elapsedMillis - uniao.inter1
elapsedMillis timeElapsed;                                      // Instancia elapsedMillis generico
elapsedMillis espera;                                           // Instancia elapsedMillis espera resposta
Nanoshield_ADC adc(0x4B);                                       // Instancia Nanoshield_ADC

//______________________________________________________________________________________________________________________ union

union
{
  byte byteArray[8];                                            // Vetor unido à estrutura para gravação e leitura
                                                                // da eeprom
  struct                                                        // Estrutura de variáveis para processamento
  {
    unsigned long valido;                                       // Código que confirma eeprom válida
    unsigned long inter1;                                       // Duração do primeiro ciclo em milissegundos
  };
}uniao;

//______________________________________________________________________________________________________________________ Funções

void limpa_vetor(char *vet);                                    // Limpa o vetor de char vet
void limpa_buffer_serial2(void);                                // Limpa buffer da Serial2
void recebe_mostra_vcpe_INAs(void);                             // Mostra dados INAs
bool aguarda_e_mostra(bool ml,int pto);                         // Aguarda P1 e mostra
void IRAM_ATTR isr1();                                          // Rotina de interrupt para botão de pedido do keypad
void IRAM_ATTR isr2();                                          // Rotina de interrupt para botão de toggle tudo
void toca_buzzer(unsigned int seg);                             // Rotina para tocar o buzzer seg segundos
bool Serial2_print_flush_wait_P1(String msg_in,int pto);        // Função para imprimir em Serial2 com
                                                                // flush e aguardar retorno de P1 com timeout
void write_eeprom(unsigned int addr, byte data);                // Grava um byte eeprom
byte read_eeprom(unsigned int addr);                            // Lê um byte eeprom
int grava_estrutura_eeprom(void);                               // Grava estrutura eeprom
int le_estrutura_eeprom(void);                                  // Lê estrutura eeprom
bool aguarda_dados_serial2(int esp,int ponto);                  // Aguarda dados Serial2
bool decodifica_comando_recebido(String ing,int *codcom,int *np,int *param); // Parsing do comando recebido
void retorno_tela_lcd03(void);                                  // Display da tela de retorno

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//_________________________________________________Processa uma vez ____________________________________________________ Setup

void setup()
{
  int m,addr,nDevices,error;

  pinMode(buzzer,OUTPUT);                                       // Pino para acionar o buzzer
  digitalWrite(buzzer,LOW);                                     // Inicialmente desativado

  Serial.begin(115200);                                         // Define velocidade
  while (!Serial);                                              // aguarda

  Wire.begin(SDApin,SCLpin);                                    // Inicia I2C

  // ___________________________________________________________________________________________________________________ varredura I2C

  Serial.println("\nVarredura I2C");
  Serial.println("\nProcurando...");
  nDevices = 0;

  for(addr = 8;addr <= 128;addr++)
  {
    Wire.beginTransmission(addr);
    error = Wire.endTransmission();
    if (error == 0)
    {
      switch (addr)
      {
        case 0x50:
          Serial.print("\nEncontrada eeprom At24c256 no endereço 0x50");
          nDevices++;
          break;
        case 0x4B:
          Serial.print("\nEncontrado Nanoshield_ADC no endereço 0x4B");
          nDevices++;
          break;
        case 0x63:
          Serial.print("\nEncontrado LCD03 no endereço 0xC6");
```

```
          nDevices++;
          break;
        default:
          Serial.print("\nEndereço desconhecido 0x");
          Serial.print(addr,HEX);
      }
    }
  }
  if (nDevices == 3) Serial.println("\nTodos os equipamentos I2C presentes");
  else Serial.println("\nEquipamentos I2C faltando");

  Serial2.begin(9600, SERIAL_8N1, RX2pin, TX2pin);              // Serial2 para Xbee - ESP32 - RX 16 TX 17
  Serial2.setTimeout(10000);                                    // Define time out
  Serial.print("\nSerial para Xbee inicializada");

  adc.begin();                                                  // Inicia o Nanoshield_ADC
  adc.setGain(GAIN_ONE);                                        // Define o ganho do ADC
  adc.setSampleRate(860);                                       // Define amostragem do ADC

  LCD03_begin(0xC6,30);                                         // Inicia o LCD03 - tespera = 30 segundos
  delay(3000);
  Serial.println("\nLCD03 inicializado");

  str = "\nCoMoSisFog " + versao +  " parte 2";
  Serial.print(str);                                            // Identificação
  Serial.print("\nI2C ESP32 - SDA 21 SCL 22 inicializado");
  Serial.print("\nNanoshield_ADC inicializado");

  hide_cursor;                                                  // Esconde cursor
  backlight_on;                                                 // Luz de fundo
  clear_screen;                                                 // Limpa tela

  str = " CoMoSisFog " + versao;
  display_string_lc(&str,1,1);                                  // Versão do sistema
  display_texto_lc(" Filippo Pardini",2,1);                     // Autor

  // Le eeprom ____________________________________________________________________________________________________ eeprom
  m = le_estrutura_eeprom();
  delay(30);

  // Se eeprom não válida define eeprom default
  if (uniao.valido != key)
  {
    uniao.valido = key;                                         // Valida a eeprom
    uniao.inter1 = 60000;                                       // 1 minuto

    m = grava_estrutura_eeprom();                               // Grava eeprom
    delay (30);

    Serial.print("\n\neeprom default");
    Serial.print("\nBytes gravados na eeprom - ");
    Serial.print(m);
  }
  else  Serial.print("\neeprom valida");                        // Eeprom valida

  Serial.print("\nBytes na eeprom - ");
  Serial.print(m);
  Serial.print("\nvalido = ");
  Serial.print(uniao.valido);
  Serial.print("\ninter1 = ");
  Serial.print(uniao.inter1);

  //_____________________________________________________________________________________________________ Carga da bateria

  // Verifica carga da bateria
  volt = adc.readVoltage(0);                                    // Lê a voltagem no ADC
  volt *= 5.87359;                                              // Escala em função do divisor de tensão
  str = " Bat " + String(volt,2) + "V";                         // Monta String para display no LCD03
  clear_line(4);
  display_string_lc(&str,4,1);                                  // Faz o display na linha 4
  if (volt <= 10.5)                                             // Verifica se tem que carregar a bateria
  {
    display_texto("  *RECARGA*");                               // Aviso para recarregar baterias - display
    toca_buzzer(5);                                             // Aviso para recarregar baterias - buzzer
  }

//_____________________________________________________________________________________________________ Interrupts

  pinMode(keypadpin, INPUT_PULLUP);                             // Pino a ser acionado quando for teclar no keypad
  attachInterrupt(digitalPinToInterrupt(keypadpin),isr1,FALLING); // Define interrupt no pino keypadpin
  pinMode(liga_desliga_tudo, INPUT_PULLUP);                     // Pino a ser acionado quando for toggle tudo
  attachInterrupt(digitalPinToInterrupt(liga_desliga_tudo),isr2,FALLING); // Define interrupt no pino liga_desliga_tudo

//_____________________________________________________________________________________________________ Inicialização variaveis

  timeElapsed1 = 0;                                             // Inicializa contagem de tempo 1 - elapsedMillis
  pedido_keypad = false;                                        // Botão keypad
```

```
  liga_desliga = false;                                         // Botão liga/desliga tudo
  tudo_ligado = false;
  vina1=cina1=pina1=vina2=cina2=pina2=vina3=cina3=pina3=etracer1=etracer2=ein=einversor=eout=volt= 0.0;

//______________________________________________________________________________________________________________________ Data e hora

  // Pede data e hora a P1
  delay(3000);
  Serial2.print(String("<*1*1>"));
  Serial2.flush();
  if (aguarda_dados_serial2(timeout,0))                         // Aguarda resposta                                        ponto 0
  {
    tempo = Serial2.readStringUntil('>') + String('>');
    tempo = tempo.substring(1,tempo.indexOf('>'));
    tempo = String(" ") + tempo;
    limpa_lcd;
    display_string_lc(&tempo,1,1);
  }
  else;                                                         // Timeout P1

//______________________________________________________________________________________________________________________

  Serial.print("\nIniciando...");
}

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//______________________________________________ Processa Continuamente ________________________________________________ Loop

void loop()
{
  int i,k,codigo,numpar,parametros[20];

  //____________________________________________________________________________________________________________________ Mensagem de P1

  if ((Serial2.available() > 0) && (Serial2.peek() == '<'))
  {
    msg = Serial2.readStringUntil('>');
    msg += String('>');
    Serial.print("\n");
    Serial.print(msg);
    limpa_lcd;

    tmp = String(" ") + msg;
    display_string_lc(&tmp,1,1);
    retorno_tela_lcd03();
  }

  //____________________________________________________________________________________________________________________ liga/desliga tudo

  // Verifica se foi apertado o botão liga/desliga tudo no comando
  if (liga_desliga)                                             // Toggle tudo
  {
    if (!tudo_ligado)
    {
      tmp = String("<*10>");
      if (Serial2_print_flush_wait_P1(tmp,1))                   // Envia comando liga tudo a P1 e aguarda resposta       ponto 1
      {
        tudo_ligado = !tudo_ligado;                             // Ligado
        liga_desliga = false;
      }
      else;                                                     // Timeout resposta P1
    }
    else
    {
      tmp = String("<*11>");
      if (Serial2_print_flush_wait_P1(tmp,2))                   // Envia comando desliga tudo a P1 e aguarda resposta    ponto 2
      {
        tudo_ligado = !tudo_ligado;                             // Desligado
        liga_desliga = false;
      }
      else;                                                     // Timeout resposta P1
    }
  }

  //____________________________________________________________________________________________________________________ comandos keypad

  // Verifica se veio solicitação do keypad para teclar comando e lê o comando.
  // Formato *c[*p1*...pn]# onde c => comando e [*p1*...pn] => parâmetros (opcional)
  // *0# => Mostra refletores e grupos ligados
  // *1*n => n=1 => mostra data e hora; n=2 => mostra status da fotocélula; n=3 => mostra energias eeprom;
  //    n=4 => Mostra energias 24h; n=5 => zera energias na eeprom e correntes
  // *2*n*v# => ciclos em P2; n=1 => inter1; v => minutos
  // *3*dia*mes*ano*hora*minuto# => acerta relogio; ano => 4 digitos; hora => 24h
  // *4*n# => solicita dados; n=1 => INAS e energias até o momento; n=2 => status grupos
  // *5*n*v# => ciclos em P1; n=1 => inter1; n=2 => inter2; v => minutos
```

```
  // *6*n*r[*r...*r]# => configuração de grupos; n => grupo; r => refletor; r¹ = 0 => limpa grupo
  // *7*n*a*b*hl*ml*hd*md# => configuração de grupo; a=0 => inativo; a=1 => ativo; b=1 => fotocélula; b=2 => comando(P2,
  //    celular); b=3 => programado => hl,ml,hd,md (hora e minuto liga, hora e minuto desliga) só existem neste caso.
  //    Este comando pode ser usado de 3 formas: *7*n*a# => só ativa/desativa grupo; *7*n*a*b# => ativa/desativa e
  //    declara tipo; *7*n*a*b*hl*ml*hd*md# => completo
  // *8*g1[*g2...*gn]# => toggle de grupos só do tipo 2 (comandado)
  //_____________________________________________________________________________________________________________
  // Via interrupt (botão vermelho) direto para P1:
  // <*10> => Liga todos os refletores (interno)
  // <*11> => Desliga todos os refletores (interno)
  //_____________________________________________________________________________________________________________
  // Solicita a P1 status dos refletores e grupos (interno)
  // <*12*n> => Status L/D, n=1 => refletores; n=2 => grupos
  //_____________________________________________________________________________________________________________

  if (pedido_keypad)                                        // Verifica se tem interrupt de teclado
  {                                                         // Sim
    limpa_lcd;
    display_texto_lc(" Digite comando",1,1);                // Display linha 1
    if(entrada_teclado(entrada))                            // Recebe digitação. A leitura é encerrada com #
    {
      Serial.print("\nDigitado: ");
      Serial.print(entrada);
      str = String(' ') + String(entrada);
      limpa_lcd;
      display_string_lc(&str,1,1);
      str = String('<') + String(entrada).substring(0,String(entrada).indexOf('#')) + String('>');
      if (decodifica_comando_recebido(str,&codigo,&numpar,parametros))
      {
        switch (codigo)
        {
        //_____________________________________________________________________________________________________________

          case 0:                                           // Status de refletores e grupos

            // Refletores
            Serial2.print("<*12*1>");
            Serial2.flush();
            if (aguarda_dados_serial2(timeout,3))           // Aguarda resposta de P1                               ponto 3
            {
              tmp = Serial2.readStringUntil('>') + String('>');
              tmp = String("RF ") + tmp.substring(1,tmp.indexOf('>'));
              Serial.print(tmp);
              tmp = String(' ') + tmp;
              limpa_lcd;
              display_string_lc(&tmp,1,1);
              delay(4000);
            }
            else;                                           // Timeout P1
            limpa_buffer_serial2();                         // Limpa buffer da Serial2

            // Grupos
            Serial2.print("<*12*2>");
            Serial2.flush();
            if (aguarda_dados_serial2(timeout,4))           // Aguarda resposta de P1                               ponto 4
            {
              tmp = Serial2.readStringUntil('>') + String('>');
              tmp = String("GR ") + tmp.substring(1,tmp.indexOf('>'));
              Serial.print(tmp);
              tmp = String(' ') + tmp;
              limpa_lcd;
              display_string_lc(&tmp,1,1);
              retorno_tela_lcd03();
            }
            else;                                           // Timeout P1
            limpa_buffer_serial2();                         // Limpa buffer da Serial2
            break;

        //_____________________________________________________________________________________________________________

          case 1:
            if ((parametros[1] == 3) || (parametros[1] == 4))   // Mostra energias eeprom ou energias 24h
            {
              Serial2.print(str);
              Serial2.flush();
              aguarda_e_mostra(true,5);                     // Aguarda resposta                                     ponto 5
            }
            else if ((parametros[1] == 1) || (parametros[1] == 2) || (parametros[1] == 5))
            {
              if (Serial2_print_flush_wait_P1(str,6));      // Envia comando a P1 e aguarda resposta                ponto 6
              else;                                         // Timeout resposta P1
            }
            else                                            // Parametro invalido
            {
              Serial.print("\nParametro invalido");
              clear_line(2);
              display_texto_lc(" Parametro invalido",2,1);
```

```
436           }
437           break;
438 
439         //_____________________________________________________________________________________________________
440 
441           case 2:                                           // Ciclo em P2
442             switch (parametros[1])
443             {
444               case 1:                                       // inter1
445                 uniao.inter1 = parametros[2] * 60000;
446                 tmp = String("inter1 = ") + String(uniao.inter1);
447                 Serial.print("\n");
448                 Serial.print(tmp);
449                 tmp = String(' ') + tmp;
450                 limpa_lcd;
451                 display_string_lc(&tmp,1,1);
452                 retorno_tela_lcd03();
453 
454                 i = grava_estrutura_eeprom();               // Grava eeprom
455                 delay (30);
456 
457                 Serial.print("\neeprom gravada");
458                 Serial.print("\nBytes gravados na eeprom - ");
459                 Serial.print(i);
460                 break;
461               default:
462                 Serial.println("\nParametro invalido ");
463                 clear_line(2);
464                 display_texto_lc(" Parametro invalido",2,1);
465                 retorno_tela_lcd03();
466             }
467             break;
468 
469         //_____________________________________________________________________________________________________
470 
471           case 3:                                           // Acerta relogio
472             if (Serial2_print_flush_wait_P1(str,7));        // Envia comando a P1 e aguarda resposta          ponto 7
473             else;                                           // Timeout resposta P1
474             break;
475 
476         //_____________________________________________________________________________________________________
477 
478           case 4:                                           // Solicita dados
479             switch (parametros[1])
480             {
481               case 1:                                       // Dados INAs
482                 Serial2.print(str);
483                 Serial2.flush();
484                 recebe_mostra_vcpe_INAs();
485                 break;
486               case 2:                                       // Status grupos
487                 Serial2.print(str);
488                 Serial2.flush();
489                 aguarda_e_mostra(true,8);                   // Aguarda resposta                               ponto 8
490                 break;
491               default:
492                 Serial.println("\nParametro invalido");
493                 clear_line(2);
494                 display_texto_lc(" Parametro invalido",2,1);
495                 retorno_tela_lcd03();
496             }
497             break;
498 
499         //_____________________________________________________________________________________________________
500 
501           case 5:                                           // Ciclos em P1
502             if ((parametros[1] == 1) || (parametros[1] == 2))
503             {
504               if (Serial2_print_flush_wait_P1(str,9));      // Envia comando a P1 e aguarda resposta          ponto 9
505               else;                                         // Timeout resposta P1
506             }
507             else
508             {
509               Serial.print("\nParametro invalido");
510               clear_line(2);
511               display_texto_lc(" Parametro invalido",2,1);
512               retorno_tela_lcd03();
513             }
514             break;
515 
516         //_____________________________________________________________________________________________________
517 
518           case 6:                                           // Configuração de grupo - refletores
519             if (parametros[1] <= 12)
520             {
521               if (Serial2_print_flush_wait_P1(str,10));     // Envia comando a P1 e aguarda resposta          ponto 10
522               else;                                         // Timeout resposta P1
```

```
            }
            else
            {
              Serial.print("\nParametro invalido");
              clear_line(2);
              display_texto_lc(" Parametro invalido",2,1);
              retorno_tela_lcd03();
            }
            break;

        //______________________________________________________________________________________________________________

          case 7:                                               // Configuração de grupo - tipo
            if (parametros[1] <= 12)
            {
              if (Serial2_print_flush_wait_P1(str,11));         // Envia comando a P1 e aguarda resposta                 ponto 11
              else;                                             // Timeout resposta P1
            }
            else
            {
              Serial.print("\nParametro invalido");
              clear_line(2);
              display_texto_lc(" Parametro invalido",2,1);
              retorno_tela_lcd03();
            }
            break;

        //______________________________________________________________________________________________________________

          case 8:                                               // Toggle de grupos só do tipo 2 - comandado
            for (i=1;i<=numpar;i++)
            {
              str = String("<*8*") + String(parametros[i]) + String(">");
              Serial2.print(str);
              aguarda_e_mostra(false,12);                       // Aguarda resposta                                      ponto 12
            }
            break;

        //______________________________________________________________________________________________________________

          default:
          Serial.print("\nKeypad - comando invalido");
          limpa_lcd;
          display_texto_lc(" Comando invalido",2,1);
          retorno_tela_lcd03();
          limpa_vetor(entrada);
          limpa_buffer_serial2();
          pedido_keypad = false;
        }
        //______________________________________________________________________________________________________________

      }
      else                                                      // Deu erro em entrada_teclado
      {
        Serial.print("\nAlgo nao funcionou - decodifica comando");
        limpa_lcd;
        display_texto_lc(" Algo nao funcionou",2,1);
        retorno_tela_lcd03();
      }
    }
    else                                                        // Deu erro em entrada_teclado
    {
      Serial.print("\nAlgo nao funcionou - entrada keypad");
      limpa_lcd;
      display_texto_lc(" Algo nao funcionou",2,1);
      retorno_tela_lcd03();
    }
    limpa_vetor(entrada);
    limpa_buffer_serial2();
    pedido_keypad = false;
  }

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//_______________________________________ Ciclo de tempo timeElapsed1 - default 1 minuto _______________________________ timeElapsed1

  if (timeElapsed1 > uniao.inter1)                              // Executa a cada ciclo de duracao inter1
  {

    //__________________________________________________________________________________________________________________ Data e hora

    // Pede data e hora a P1
    Serial2.print(String("<*1*1>"));
    Serial2.flush();
    if (aguarda_dados_serial2(timeout,13))                      // Aguarda resposta                                      ponto 13
    {
```

```
      tempo = Serial2.readStringUntil('>') + String('>');
      tempo = tempo.substring(1,tempo.indexOf('>'));
      tempo = String(" ") + tempo;
      limpa_lcd;
      display_string_lc(&tempo,1,1);
    }
    else;                                                       // Timeout P1

    //__________________________________________________________________________________________________________________ Carga da bateria

    // Verifica carga da bateria
    volt = adc.readVoltage(0);                                  // Lê a voltagem no ADC
    volt *= 5.87359;                                            // Escala em função do divisor de tensão
    str = " Bat " + String(volt,2) + "V";                       // Monta String para display no LCD03
    clear_line(4);
    display_string_lc(&str,4,1);                                // Faz o display na linha 4
    if (volt <= 10.5)                                           // Verifica se tem que carregar a bateria
    {
      display_texto("  *RECARGA*");                             // Aviso para recarregar baterias - display
      toca_buzzer(5);                                           // Aviso para recarregar baterias - buzzer
    }

    //__________________________________________________________________________________________________________________

    timeElapsed1 = 0;                                           // Reinicia ciclo
  }
}

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//______________________________________________________Funções_________________________________________________________ isr1

void IRAM_ATTR isr1()                                           // Rotina para processamento do interrupt do keypad
{
  pedido_keypad = true;
}

//______________________________________________________________________________________________________________________ isr2

void IRAM_ATTR isr2()                                           // Rotina para processamento do interrupt de l/d tudo
{
  liga_desliga = true;
}

//______________________________________________________________________________________________________________________ limpa_vetor

void limpa_vetor(char *vet)
{
  int g;
  for (g=0;g<=20;g++)
  {
    vet[g] = ' ';
  }
}

//______________________________________________________________________________________________________________________ limpa_buffer_serial2

void limpa_buffer_serial2(void)
{
  while (Serial2.available()) Serial2.read();
}

//______________________________________________________________________________________________________________________ toca_buzzer

void toca_buzzer(unsigned int seg)
{
  // Toca o buzzer por seg segundos
  long intervalo = (long)(seg * 1000);

  timeElapsed = 0;                                              // Inicia contagem de tempo
  while(timeElapsed < intervalo)                                // Executa enquanto não passar o intervalo
  {
    digitalWrite(buzzer, HIGH);
    delay(500);
    digitalWrite(buzzer, LOW);
    delay(500);
  }
}

//______________________________________________________________________________________________________________________ Serial2_print_flush_wait_P1

// Envia comando a P1 e aguarda resposta
bool Serial2_print_flush_wait_P1(String msg_in,int pto)
{
  Serial2.print(msg_in);
  Serial2.flush();
```

```
  if (aguarda_dados_serial2(timeout,pto))
  {
    msg = Serial2.readStringUntil('>') + String('>');
    msg = msg.substring(1,msg.indexOf('>'));
    Serial.print("\n");
    Serial.print(msg);
    limpa_lcd;
    msg = String(" ") + msg;
    display_string_lc(&msg,1,1);
    retorno_tela_lcd03();
  }
  else return false;
}

//_____________________________________________________________________________________________________________ aguarda_e_mostra

bool aguarda_e_mostra(bool ml,int pto)
{
  String grp;
  uint8_t il;

  il = 1;

  while(1)
  {
    if (aguarda_dados_serial2(timeout,pto))                    // Aguarda dados P1
    {
      if (Serial2.peek() == '#')
      {
        Serial2.read();
        retorno_tela_lcd03();
        return true;
      }
      else
      {
        grp = Serial2.readStringUntil('>') + String('>');
        grp = grp.substring(1,grp.indexOf('>'));
        Serial.print("\n");
        Serial.print(grp);
        grp = String(' ') + grp;
        if (il >= 4)
        {
          delay(2000);
          limpa_lcd;
          il = 1;
        }
        if (ml) display_string_lc(&grp,il++,1);
        else
        {
          limpa_lcd;
          display_string_lc(&grp,1,1);
        }
        delay(2000);
        Serial2.print('#');                                    // Avisa P1 OK
        Serial2.flush();
      }
    }
    else return false;                                         // Timeout P1
  }
}

//_____________________________________________________________________________________________________________ recebe_mostra_vcp_INA

void recebe_mostra_vcpe_INAs(void)
{
  // Recebe os dados das INAs de P1
  // nina=1 => dados INA1; nina=2 => dados INA2; nina=3 => dados INA3; nina=4 => dados de energia (ein,eout)
  // Formato: nina=1 => <v1,c1,p1> ; nina=2 => <v2,c2,p2> ; nina=3 => <v3,c3,p3> ; nina=4 => <ei,eo>
  // v - voltagem; c - corrente; p - potencia; ei - ein_24h; eo - eout_24h
  // Ex: nina=1 => <24.10,10.01,240.50>

  if (aguarda_dados_serial2(timeout,14))                       // Aguarda dados P1                                  ponto 14
  {
    vina1 = Serial2.parseFloat();
    cina1 = Serial2.parseFloat();
    pina1 = Serial2.parseFloat();
    Serial.print("\nvina1 = "); Serial.print(vina1,2);
    Serial.print("\ncina1 = "); Serial.print(cina1,2);
    Serial.print("\npina1 = "); Serial.print(pina1,2);
    limpa_lcd;
    delay(500);
    display_texto_lc(" vina1 = ",1,1);
    display_float(vina1,1);
    delay(500);
    display_texto_lc(" cina1 = ",2,1);
    display_float(cina1,1);
```

```
    delay(500);
    display_texto_lc(" pina1 = ",3,1);
    display_float(pina1,1);
    delay(2000);

    Serial2.print('#');                                     // Avisa P1 OK
    Serial2.flush();
  }
  else return;

  if (aguarda_dados_serial2(timeout,15))                    // Aguarda dados P1                                 ponto 15
  {
    vina2 = Serial2.parseFloat();
    cina2 = Serial2.parseFloat();
    pina2 = Serial2.parseFloat();
    Serial.print("\nvina2 = "); Serial.print(vina2,2);
    Serial.print("\ncina2 = "); Serial.print(cina2,2);
    Serial.print("\npina2 = "); Serial.print(pina2,2);
    limpa_lcd;
    delay(500);
    display_texto_lc(" vina2 = ",1,1);
    display_float(vina2,1);
    delay(500);
    display_texto_lc(" cina2 = ",2,1);
    display_float(cina2,1);
    delay(500);
    display_texto_lc(" pina2 = ",3,1);
    display_float(pina2,1);
    delay(2000);

    Serial2.print('#');                                     // Avisa P1 OK
    Serial2.flush();
  }
  else return;

  if (aguarda_dados_serial2(timeout,16))                    // Aguarda dados P1                                 ponto 16
  {
    vina3 = Serial2.parseFloat();
    cina3 = Serial2.parseFloat();
    pina3 = Serial2.parseFloat();
    Serial.print("\nvina3 = "); Serial.print(vina3,2);
    Serial.print("\ncina3 = "); Serial.print(cina3,2);
    Serial.print("\npina3 = "); Serial.print(pina3,2);
    limpa_lcd;
    delay(500);
    display_texto_lc(" vina3 = ",1,1);
    display_float(vina3,1);
    delay(500);
    display_texto_lc(" cina3 = ",2,1);
    display_float(cina3,1);
    delay(500);
    display_texto_lc(" pina3 = ",3,1);
    display_float(pina3,1);
    delay(2000);

    Serial2.print('#');                                     // Avisa P1 OK
    Serial2.flush();
  }
  else return;

  if (aguarda_dados_serial2(timeout,17))                    // Aguarda dados P1                                 ponto 17
  {
    ein = Serial2.parseFloat() / 1000.0;
    eout = Serial2.parseFloat() / 1000.0;
    Serial.print("\nein = "); Serial.print(ein,2); Serial.print("Kwh");
    Serial.print("\neout = "); Serial.print(eout,2); Serial.print("Kwh");
    limpa_lcd;
    delay(500);
    display_texto_lc(" ein = ",1,1);
    display_float(ein,2); display_texto(" Kwh");
    delay(500);
    display_texto_lc(" eout = ",2,1);
    display_float(eout,2); display_texto(" Kwh");
    retorno_tela_lcd03();
  }
  else return;
}

//_____________________________________________________________________________________________________________ write_eeprom

void write_eeprom(unsigned int addr, byte data)
{
  Wire.beginTransmission(0x50);                             // Endereço I2C da eeprom
  Wire.write((int)(addr >> 8));                             // Envia MSB do endereço na eeprom
  Wire.write((int)(addr & 0xFF));                           // Envia LSB do endereço na eeprom
  Wire.write(data);                                         // Envia byte de dados a ser gravado
  Wire.endTransmission();
```

```
  delay(6);                                          // Tempo necessário à gravação na eeprom (min 3,5mseg)
}

//_____________________________________________________________________________________________________________ read_eeprom

byte read_eeprom(unsigned int addr)
{
  byte rdata;

  Wire.beginTransmission(0x50);                      // Endereço I2C da eeprom
  Wire.write((int)(addr >> 8));                      // Envia MSB do endereço na eeprom
  Wire.write((int)(addr & 0xFF));                    // Envia LSB do endereço na eeprom
  Wire.endTransmission();
  Wire.requestFrom(0x50,1);                          // Solicita o byte
  while (!Wire.available());
  rdata = Wire.read();                               // Se disponivel, lê byte
  delay(6);
  return rdata;                                      // Retorna o byte lido ou 0xFF se não conseguiu ler
}

//_____________________________________________________________________________________________________ grava_estrutura_eeprom

int grava_estrutura_eeprom(void)
{
  int n;
  byte *p = uniao.byteArray;

  for (n=0;n<sizeof(uniao.byteArray);n++)
  {
    write_eeprom(n,*p++);
  }
  return n;
}

//_____________________________________________________________________________________________________ le_estrutura_eeprom

int le_estrutura_eeprom(void)
{
  int n;
  byte *p = uniao.byteArray;

  for (n=0;n<sizeof(uniao.byteArray);n++)
  {
    *p++ = read_eeprom(n);
  }
  return n;
}

//_____________________________________________________________________________________________________ aguarda_dados_serial2

bool aguarda_dados_serial2(int esp,int ponto)
{
  unsigned long tim;
  String to;

  tim = (unsigned long)esp * 1000;
  espera = 0;

  while ((Serial2.available() == 0) && (espera <= tim));
  if (Serial2.available() > 0) return true;
  else
  {
    to = String("Timeout - ponto ") + String(ponto);
    Serial.print("\n"); Serial.print(to);
    to = String(" ") + to;
    limpa_lcd;
    display_string_lc(&to,1,1);
    retorno_tela_lcd03();
    return false;
  }
}

//_____________________________________________________________________________________________________ decodifica_comando_recebido

bool decodifica_comando_recebido(String ing,int *codcom,int *np,int *param)
{
  // Decodifica um comando recebido no formato : <*c[*p1...*pn]>
  // onde:
  // c => comando (obrigatório); p1...pn => parametros (opcional); * => divisor (primeiro obrigatório,
  // outros se houver parametros); < e > => delimitadores (obrigatório)

  int ni,nf,n;  // ni => indice inicial; nf => indice final; n => número de parametros

  n = 0;

  if ((ing.indexOf('<') != -1) && (ing.indexOf('>') != -1) && (ing.indexOf('*') == 1)) // Garante <* e >
  {
```

```
    ing = ing.substring(0,ing.indexOf('>')) + String('>');      // Corta além do >
    ni = ing.indexOf('*') + 1;                                  // Indice inicio codigo de comando
    nf = ing.indexOf('*',ni);                                   // Indice fim codigo de comando
    if (nf == -1)                                               // Se não achou (-1), não tem parametros
    {
      nf = ing.indexOf('>',ni);                                 // Novo indice fim codigo de comando
      *codcom = ing.substring(ni,nf).toInt();                   // Codigo de comando sem parametros
      *np = n;                                                  // Numero de parametros => 0
      return true;                                              // Retorna verdadeiro
    }

    // Tem parametros
    *codcom = ing.substring(ni,nf).toInt();                     // Codigo de comando com parametros
    while(1)                                                    // Cicla
    {
      n++;                                                      // Conta parametros
      ni = nf + 1;                                              // Indice inicial parametro
      nf = ing.indexOf('*',ni);                                 // Indice final parametro
      if (nf == -1)                                             // Se -1 é ultimo parametro
      {
        nf = ing.indexOf('>',ni);                               // Novo indice final parametro
        param[n] = ing.substring(ni,nf).toInt();                // Parametro n
        *np = n;                                                // número de parametros
        return true;                                            // Retorna verdadeiro
      }
      else param[n] = ing.substring(ni,nf).toInt();             // Tem mais parametros
    }
  }
  else return false;                                            // Comando invalido, retorna falso
}

//_____________________________________________________________________________________________________________ retorno_tela_lcd03

void retorno_tela_lcd03(void)
{
  delay(3000);
  limpa_lcd;
  display_string_lc(&tempo,1,1);
}

//_____________________________________________________________________________________________________________
```