```
/*****************************************************************************************************************
* 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/>.                                                                         *
*                                                                                                                     *
* © Copyright 2019-2020 Filippo Pardini - filippo@robotica.eng.br - http://blog.robotica.eng.br/                      *
*****************************************************************************************************************/
/*
  Projeto CoMoSisFog V3.1.3 (03/05/2020)
  * Projeto para Controle e Monitoramento de um Sistema Fotovoltaico off grid para iluminacao 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 equalizadores). 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 (~ 550W) em 220Vca
  * através de disjuntores 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 paineis 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, uma fotocelula, 3 sensores INA226,
  * um Xbee, um botão NA e um micro SD. 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, potencia 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, otimizando-as.

  Parte 2 - Laboratorio (na residência)
```

```
44
45    ESP32 remoto - controlador
46    * Esta parte utiliza uma placa DOIT ESP32 DEVKIT V1, uma eeprom At24c256 - 32KB, um Nanoshield ADC circuitar, um
47    * display LCD03 com keypad, um buzzer, dois botões NA, um Xbee, um PC/keyboard/mouse, uma fonte 5Vdc com bateria
48    * ion-litio 3S e um celular. Ela atua como servidor AP para comunicação com o celular (o qual envia comandos
49    * para acionar os grupos de refletores) e como entrada de configurações, solicitações e comandos, via PC, para a
50    * Parte 1 via Xbee. Além disso, faz o display, no LCD03, dos dados enviados pela parte 1, comanda os grupos de
51    * refletores pelo keypad e envia os dados para a nuvem via MQTT (cloudMQTT) e ThingSpeak (para analise com MATLAB).
52  */
53
54  //*************************************************************************************************************************
55  //****************************************** Este é o sketch da Parte 2 ***************************************************
56  //*************************************************************************************************************************
57
58  // Bibliotecas
59  #include <DEF_N1.h>                                                 // Definições
60  #include <WiFi.h>                                                   // WiFi
61  #include <DNSServer.h>                                              // DNS
62  #include <WebServer.h>                                              // Server
63  #include <Wire.h>                                                   // I2C
64  #include <LCD03_N1.h>                                               // LCD03
65  #include <elapsedMillis.h>                                          // Intervalos de tempo
66  #include <Nanoshield_ADC.h>                                         // Nanoshield ADC ADS1115
67  #include <WiFi.h>                                                   // WiFi
68  #include <PubSubClient.h>                                           // MQTT
69  #include "ThingSpeak.h"                                             // ThingSpeak
70  #include <ArduinoJson.h>                                            // Json
71
72  //_____________________________________________________________________________________________________________________
73
74  #define i2c_addr_eeprom 0x50                                        // Endereco I2C da eeprom At24LC256
75
76  template <class T> int eeWrite(int ee, const T& value)              // Classe genérica para gravação da eeprom
77  {                                                                   // ee - posição na eeprom, value - estrutura de dados
78    const byte* p = (const byte*)(const void*)&value;                 // Pointer para a estrutura
79    int i;                                                            // Contador
80    Wire.beginTransmission(i2c_addr_eeprom);                          // Inicia transmissão I2C
81    Wire.write((int)(ee >> 8)); // MSB                                // Envia MSB da posição
82    Wire.write((int)(ee & 0xFF)); // LSB                             // Envia LSB da posição
83    for (i = 0; i < sizeof(value); i++)                               // Para todos os bytes da estrutura
84    Wire.write(*p++);                                                 // Envia o byte
85    Wire.endTransmission();                                           // Encerra transmissão I2C
86    return i;                                                         // Retorna o número de bytes gravados
```

```
87  }
88
89  template <class T> int eeRead(int ee, T& value)                // Classe genérica para leitura da eeprom
90  {                                                              // ee - posição na eeprom, value - estrutura de dados
91    byte* p = (byte*)(void*)&value;                              // Pointer para a estrutura
92    int i;                                                       // Contador
93    Wire.beginTransmission(i2c_addr_eeprom);                     // Inicia transmissão I2C
94    Wire.write((int)(ee >> 8)); // MSB                           // Envia MSB da posição
95    Wire.write((int)(ee & 0xFF)); // LSB                        // Envia LSB da posição
96    Wire.endTransmission();                                      // Encerra transmissão I2C
97    Wire.requestFrom(i2c_addr_eeprom,sizeof(value));             // Solicita à eeprom o número de bytes da estrutura
98    for (i = 0; i < sizeof(value); i++)                          // Para esse número de bytes
99    if(Wire.available())                                         // Se tem dados no buffer
100   *p++ = Wire.read();                                          // Lê o byte para a estrutura
101   return i;                                                    // Retorna o número de bytes lidos
102 }
103
104 struct eeprom                                                  // Estrutura de dados na eeprom
105 {
106   unsigned long valido;                                        // Código para validar dados
107   unsigned long inter1;                                        // Intervalo de tempo para "elapsedMillis timeElapsed1;"
108   unsigned long inter2;                                        // Intervalo de tempo para "elapsedMillis timeElapsed2;"
109   unsigned int cod_mon;                                        // Código para imprimir ou não no monitor serial
110 } eeprom;
111
112 //_____________________________________________________________________________________________________________________
113
114 // WiFi
115 const char* ssid = ssid_ego;                                   // Nome da rede WiFi
116 const char* password = password_ego;                           // Senha da rede WiFi
117
118 //_____________________________________________________________________________________________________________________
119
120 // MQTT
121 const char* BROKER_MQTT = BROKER_MQTT_ego;                     // URL do broker MQTT
122 int BROKER_PORT = BROKER_PORT_ego;                             // Porta do Broker MQTT
123 #define ID_MQTT  ID_MQTT_ego                                   // User ID
124 #define PSW_MQTT PSW_MQTT_ego                                  // Password
125 #define tracer1 "comosisfog/tracer1"                           // Topico tracer1
126 #define tracer2 "comosisfog/tracer2"                           // Topico tracer2
127 #define inversor "comosisfog/inversor"                         // Topico inversor
128 #define etotin "comosisfog/etotin"                             // Topico energia total acumulada
129 #define etotout "comosisfog/etotout"                           // Topico energia total consumida
```

```
130  #define aviso1 "comosisfog/aviso1"                                    // Topico aviso1
131  #define aviso2 "comosisfog/aviso2"                                    // Topico aviso2
132
133  //_____________________________________________________________________________________________________________
134
135  // Variaveis e constantes
136  #define SDApin 21                                                     // I2C SDA
137  #define SCLpin 22                                                     // I2C SCL
138  #define RX1pin 16                                                     // Pino RX2
139  #define TX1pin 17                                                     // pino TX2
140  #define RX2pin 4                                                      // UART2 RX
141  #define TX2pin 2                                                      // UART2 TX
142  #define liga_desliga_tudo 26                                          // Pino de interrupt para ligar ou desligar tudo
143  #define keypadpin 25                                                  // Pino de interrupt para o keypad
144  #define buzzer 27                                                     // Pino para acionamento do buzzer
145
146  char buffer[100];                                                     // Buffer
147  char entrada[21];                                                     // Vetor para receber digitação (1 linha do LCD03)
148  const byte DNS_PORT = 53;                                             // DNS
149  const char *ssidsoftAP = ssidsoftAP_ego;                              // WiFi.softAP
150  const char *passwordsoftAP = passwordsoftAP_ego;                      // WiFi.softAP
151  unsigned long key = 2863311530;                                       // Código eeprom válida
152  volatile bool pedido_keypad;                                          // Variável para interrupt do keypad
153  volatile bool liga_desliga;                                           // Variável para o interrupt de liga/desliga tudo
154  long lastReconnectAttempt = 0;                                        //  Tempo última conexão MQTT
155  String versao = "V3.1.3";                                             // ************* Versão *********************************
156
157  unsigned int dia;
158  unsigned int mes;
159  unsigned int ano;
160  unsigned int horas;
161  unsigned int minutos;
162  unsigned int segundos;
163
164  unsigned int d1;
165  unsigned int m1;
166  unsigned int a1;
167  unsigned int d2;
168  unsigned int m2;
169  unsigned int a2;
170
171  float vina1,cina1,pina1,vina2,cina2,pina2,vina3,cina3,pina3,etracer1,etracer2,etotalin,einversor,etotalout,volt;
172
```

```
173  String tempo,str;
174
175  unsigned int k,tmp;
176  uint8_t cont1;
177  char sp;
178  bool ch;
179
180  //_____________________________________________________________________________________________________________
181
182  IPAddress local_ip(192, 168, 5, 1);                          // IP do servidor AP
183  IPAddress gateway(192, 168, 5, 1);                           // Gateway do servidor AP
184  IPAddress subnet(255, 255, 255, 0);                          // Subnet do servidor AP
185
186  //_____________________________________________________________________________________________________________
187
188  // Instancias
189  WiFiClient wificlient;                                       // Instancia WiFi MQTT
190  WiFiClient wifiTS;                                           // Instancia WiFi ThingSpeak
191  PubSubClient MQTT(BROKER_MQTT,BROKER_PORT,wificlient);       // Instancia MQTT passando o objeto wificlient
192  DNSServer dnsServer;                                         // Instancia DNSServer para nome de dominio
193  WebServer server(80);                                        // Instancia WebServer para serevidor AP
194  elapsedMillis timeElapsed1;                                  // Instancia elapsedMillis para eeprom.inter1
195  elapsedMillis timeElapsed2;                                  // Instancia elapsedMillis para eeprom.inter2
196  elapsedMillis timeElapsed;                                   // Instancia elapsedMillis generico
197  Nanoshield_ADC adc(0x4B);                                    // Instancia Nanoshield_ADC
198
199  //_____________________________________________________________________________________________________________
200
201  // ThingSpeak
202  unsigned long myChannelNumber = myChannelNumber_ego;         // ThingSpeak ID do canal
203  const char * myWriteAPIKey = myWriteAPIKey_ego;              // ThingSpeak chave de escrita
204  String myStatus = "";                                        // ThingSpeak para teste (11)
205
206  //_____________________________________________________________________________________________________________
207
208  // Funções
209  void conecta_MQTT(void);                                     // Conecta MQTT
210  bool reconnect();                                            // Reconecta MQTT
211  void conecta_wifi(unsigned char pr);                         // Conecta WiFi
212  bool recebe_status(void);                                    // Recebe os dados dos grupos e refletores
213  bool recebe_inas(void);                                      // Recebe dados das INAs
214  bool pede_p1(uint8_t sensor);                                // Pede dados à Parte 1
215  void envia_PC(unsigned int kk);                              // Envia dados ao PC
```

```
void mostra_LCD03(void);                                        // Mostra dados no LCD03
void limpa_buffer_serial(uint8_t ser,uint8_t cmp);              // Limpa buffer das seriais
bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde); // Aguarda caracteres nas seriais
bool recebido_P1(unsigned long tp);                             // ACK de P1
unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano); // Calcula número de dias
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
String SendHTML(void) ;                                         // Para o servidor AP
void handle_NotFound(void);                                     // Para o servidor AP
void handle_grupo0(void);                                       // Para o servidor AP
void handle_grupo1(void);                                       // Para o servidor AP
void handle_grupo2(void);                                       // Para o servidor AP
void handle_grupo3(void);                                       // Para o servidor AP
void handle_grupo4(void);                                       // Para o servidor AP
void handle_grupo5(void);                                       // Para o servidor AP
void handle_grupo6(void);                                       // Para o servidor AP
void handle_grupo7(void);                                       // Para o servidor AP
void handle_grupo8(void);                                       // Para o servidor AP
void handle_grupo9(void);                                       // Para o servidor AP
void handle_grupo10(void);                                      // Para o servidor AP
void handle_grupo11(void);                                      // Para o servidor AP
void handle_grupo12(void);                                      // Para o servidor AP
void handle_OnConnect(void);                                    // Para o servidor AP

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

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

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

  Serial.begin(115200);                                         // Monitor serial
  while (!Serial);

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

```
259
260   LCD03_begin(0xC6,30);                                    // Inicia o LCD03 - tespera = 30 segundos
261   delay(3000);
262
263   str = "\nCoMoSisFog " + versao +  " parte 2";
264   Serial.print(str);                                       // Identificação
265   Serial.print("\nI2C ESP32 - SDA 21 SCL 22 inicializado");
266   Serial.print("\nNanoshield_ADC inicializado");
267   Serial.println("\nLCD03 inicializado");
268
269   hide_cursor;                                             // Esconde cursor
270   //backlight_on;                                          // Luz de fundo
271   clear_screen;                                            // Limpa tela
272
273   str = " CoMoSisFog " + versao;
274   display_string_lc(&str,1,1);                             // Versão do sistema
275   display_texto_lc(" Filippo Pardini",2,1);                // Autor
276
277   volt = adc.readVoltage(0);                               // Lê a voltagem no ADC
278   volt *= 5.87359;                                         // Escala em função do divisor de tensão
279   str = " Bat " + String(volt,2) + "V";                    // Monta String para display no LCD03
280   display_string_lc(&str,4,1);                             // Faz o display
281   if (volt <= 9.6)                                         // Verifica se tem que carregar a bateria
282   {
283     display_texto("  *RECARGA*");                          // Aviso para recarregar baterias - display
284     toca_buzzer(5);                                        // Aviso para recarregar baterias - buzzer
285   }
286
287   toca_buzzer(5);
288
289   conecta_wifi(1);                                         // Conecta WiFi
290
291   ThingSpeak.begin(wifiTS);                                // Inicia ThingSpeak
292
293   Serial.print("\nThingSpeak inicializado");
294
295   WiFi.softAP(ssidsoftAP, passwordsoftAP, 13, false, 1);   // Parametros de acesso ao servidor AP
296   delay(100);
297   WiFi.softAPConfig(local_ip, gateway, subnet);            // Parametros de acesso ao servidor AP
298
299   dnsServer.setTTL(300);                                   // Servidor DNS
300   dnsServer.setErrorReplyCode(DNSReplyCode::ServerFailure);// Servidor DNS
301   dnsServer.start(DNS_PORT, "comosisfog.net", local_ip);   // Servidor DNS
```

```
302
303     Serial2.begin(115200, SERIAL_8N1, RX1pin, TX1pin);          // Serial1 para Xbee - ESP32 - RX 16 TX 17
304     Serial2.setTimeout(10000);                                  // Time out
305     Serial.print("\nSerial para Xbee inicializada");
306
307     Serial1.begin(115200, SERIAL_8N1, RX2pin, TX2pin);          // Serial2 para PC - ESP32 - RX 4 TX 2
308     Serial1.setTimeout(10000);                                  // Time out
309     Serial.print("\nSerial para PC inicializada");
310
311     // Atribuição dos handles para o servidor AP
312
313     server.on("/", HTTP_GET, handle_OnConnect);
314     server.on("/grupo0", HTTP_GET, handle_grupo0);
315     server.on("/grupo1", HTTP_GET, handle_grupo1);
316     server.on("/grupo2", HTTP_GET, handle_grupo2);
317     server.on("/grupo3", HTTP_GET, handle_grupo3);
318     server.on("/grupo4", HTTP_GET, handle_grupo4);
319     server.on("/grupo5", HTTP_GET, handle_grupo5);
320     server.on("/grupo6", HTTP_GET, handle_grupo6);
321     server.on("/grupo7", HTTP_GET, handle_grupo7);
322     server.on("/grupo8", HTTP_GET, handle_grupo8);
323     server.on("/grupo9", HTTP_GET, handle_grupo9);
324     server.on("/grupo10", HTTP_GET, handle_grupo10);
325     server.on("/grupo11", HTTP_GET, handle_grupo11);
326     server.on("/grupo12", HTTP_GET, handle_grupo12);
327     server.onNotFound(handle_NotFound);
328
329     server.begin();                                             // Inicializa servidor AP
330     Serial.print("\nweb server e DNS inicializados");
331
332   //_____________________________________________________________________________________________________________________
333
334   // Força eeprom invalida para inicializar default
335   // Depois lembrar de retirar este codigo
336     //eeprom.valido = 0;
337     //eeprom.inter1 = 0;
338     //eeprom.inter2 = 0;
339     //eeprom.cod_mon = 0;
340     //eeWrite(0,eeprom);
341     //delay (30);
342   //_____________________________________________________________________________________________________________________
343
344     eeRead(0,eeprom);                                           // Le eeprom
```

```
  delay (30);

  if (eeprom.valido != key)                                          // Eeprom invalida, força reinicializacao default
  {
    eeprom.inter1 = 60000;                                           // Default - 1 minuto
    eeprom.inter2 = 300000;                                          // Default - 5 minutos
    eeprom.cod_mon = 1;                                              // Habilita monitor serial
    eeprom.valido = key;                                             // Valida eeprom

    eeWrite(0,eeprom);                                               // Grava eeprom
    delay (30);
    Serial.print("\neeprom default");
  }

  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

  timeElapsed1 = 0;
  timeElapsed2 = 0;
  timeElapsed = 0;
  cont1 = 0;
  pedido_keypad = false;
  liga_desliga = false;
}

//_______________________________________________________________________________________________________________________

void loop()
{
  conecta_MQTT();                                                    // Conecta com o Broker MQTT
  MQTT.loop();

  dnsServer.processNextRequest();                                    // DNS
  server.handleClient();

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//_______________________________________________ Processa Continuamente ________________________________________________

  if (Serial2.available() > 10)                                      // *** Atenção - verificar se 10 é OK ***
```

```
388     {
389       // Verifica se P1 enviou algum texto <....>
390       if (Serial2.read() == '<')                              // Verifica se a P1 enviou mensagem - formato - <texto>
391       {
392         str = " " + Serial2.readStringUntil('>');
393         clear_line(1);
394         clear_line(2);
395         clear_line(3);
396         display_string_lc(&str,1,1);                          // Display no LCD03
397         toca_buzzer(5);                                       // Toca buzzer
398         while (read_keypad() != '#')                          // Espera para ler. Repete até digitar #
399         {
400           delay(3000);                                        // Aguarda
401           toca_buzzer(5);                                     // Toca buzzer
402         }
403       }
404     }
405
406     //_____________________________________________________________________________________________________________
407
408     if (Serial1.available() >= 4)                             // Pc enviou algo?*** Atenção - verificar se 4 é OK ***
409     {
410       // Verifica se o PC enviou alguma coisa: comando, coniguração p1, coniguração p2, configuração dos grupos de
411       // refletores, configuração de seus modos ou solicitação de dados. Ou a Parte 1 enviou mensagem.
412       // Indicadores válidos:
413       // '!' - Comando toggle dos grupos de refletores - formato - !(g) - onde g é o número do grupo.  g = 0 => tudo
414       // '$' - Solicitação de dados - formato - $(i) - onde i=1 data-hora, i=2 dados INAs e energias, i=3 status dos
415       //       grupos e refletores
416       // '@' - Solicitação de dados - formato - @(di,df) - onde di - data de início, df - data de fim, ambos no formato
417       //       dia,mes,ano. Ex: @(1,8,2019/31,8,2019) vai ler o SD e enviar os dados relativos a esse período
418       // '%' - Configuração de ciclos em p2 - formato - %n(v) - onde n=1 eeprom.inter1, n=2 eeprom.inter2,
419       //       n=3 eepro.cod_mon, v = 0 ou 1 ou valor em minutos
420       // '#' - Configuração de ciclos em p1 - formato - #n(v) - onde n=1 inter1, n=2 inter2,
421       //       n=3 cod_mon, v = 0 ou 1 ou valor em minutos
422       // '&' - Configuração dos grupos de refletores - formato - &n(r,...,r) - onde n é o número do grupo
423       //       e r o número do refletor
424       // '*' - Configuração dos horarios liga/desliga dos grupos e modo de operação de cada grupo
425       //       Modo de operação: ativo -> sim(1)/não(0), automatico -> fotocelula, programado -> hora liga/desliga,
426       //       manual -> comando de toggle via p2,PC ou celular - formato - *n(a,b,c,d) - onde n é o número do grupo
427       //       a=0 -> inativo, a=1 => ativo, b=1 -> automático(fotocelula), b=2 -> manual(comando PC,p2,celular),
428       //       b=3 -> programado (c e d só existem neste caso), c -> hora liga (formato hh:mm hh -> hora formato 24,
429       //       mm -> minutos), d -> hora desliga (idem hora liga)
430
```

```
431       sp = Serial1.peek();
432
433       switch (sp)
434       {
435         case '%':
436           ch = true;
437           if (!aguarda_serial(1,10000,5,1)) break;                // Serial1,10seg,5 caracteres,ponto de chamada 1
438           switch(Serial1.parseInt())                              // Configura ciclos em p2
439           {
440             case 1:                                               // Interval1_p2 em minutos
441               tmp = Serial1.parseInt();
442               eeprom.inter1 = tmp * 60000;                        // Passa para milisegundos
443               Serial1.print("\n%1(");
444               Serial1.print(tmp);
445               Serial1.print(")");
446               break;
447             case 2:                                               // Interval2_p2 em minutos
448               tmp = Serial1.parseInt();
449               eeprom.inter2 = tmp * 60000;                        // Passa para milisegundos
450               Serial1.print("\n%2(");
451               Serial1.print(tmp);
452               Serial1.print(")");
453               break;
454             case 3:                                               // eeprom.cod_mon
455               eeprom.cod_mon = Serial1.parseInt();
456               Serial1.print("\n%3(");
457               Serial1.print(eeprom.cod_mon);
458               Serial1.print(")");
459               break;
460             default:
461               Serial.println("\nComando % - parametro invalido ");
462               ch = false;
463               break;
464           }
465           if (ch)
466           {
467             eeWrite(0,eeprom);                                    // Grava eeprom
468             delay (30);
469             Serial.print("\neeprom gravada P2");
470             MQTT.publish(aviso2,"eeprom P2 configurada");         // Avisa no MQTT
471           }
472           break;
473         case '$':
```

```
474           if (!aguarda_serial(1,10000,4,2)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 2
475           k = Serial1.parseInt();
476           if (!pede_p1(k))
477           {
478             Serial.print("\nSem resposta de P1 - ponto de chamada 14"); // ponto de chamada 14
479           }
480           break;
481         case '!':
482           if (!aguarda_serial(1,10000,4,3)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 3
483           str = Serial1.readStringUntil(')') + ')';               // Lê comando
484           Serial2.print(str);                                     // Envia comando para P1
485           if (!recebido_P1(10000,10))                             // Serial2,10seg,1 caractere,ponto de chamada 10
486           {
487             Serial.print("\nSem resposta de P1 - ponto de chamada 15"); // ponto de chamada 15
488             break;
489           }
490           Serial.print("\nEnviou para P1: "); Serial.print(str);
491           break;
492         case '#':
493           if (!aguarda_serial(1,10000,5,4)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 4
494           str = Serial1.readStringUntil(')') + ')';               // Lê comando
495           Serial2.print(str);                                     // Envia comando para P1
496           if (!recebido_P1(10000,11))                             // Serial2,10seg,1 caractere,ponto de chamada 11
497           {
498             Serial.print("\nSem resposta de P1 - ponto de chamada 16"); // ponto de chamada 16
499             break;
500           }
501           Serial.print("\nEnviou para P1: "); Serial.print(str);
502           MQTT.publish(aviso1,"eeprom P1 configurada");           // Avisa no MQTT
503           break;
504         case '&':
505           if (!aguarda_serial(1,10000,5,5)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 5
506           str = Serial1.readStringUntil(')') + ')';               // Lê comando
507           Serial2.print(str);                                     // Envia comando para P1
508           if (!recebido_P1(10000,12))                             // Serial2,10seg,1 caractere,ponto de chamada 12
509           {
510             Serial.print("\nSem resposta de P1 - ponto de chamada 17"); // ponto de chamada 17
511             break;
512           }
513           Serial.print("\nEnviou para P1: "); Serial.print(str);
514           break;
515         case '*':
516           if (!aguarda_serial(1,10000,6,6)) break;                // Serial1,10seg,6 caracteres,ponto de chamada 6
```

```
517           str = Serial1.readStringUntil(')') + ')';               // Lê comando
518           Serial2.print(str);                                     // Envia comando para P1
519           if (!recebido_P1(10000,13))                             // Serial2,10seg,1 caractere,ponto de chamada 13
520           {
521             Serial.print("\nSem resposta de P1 - ponto de chamada 18"); // ponto de chamada 18
522             break;
523           }
524           Serial.print("\nEnviou para P1: "); Serial.print(str);
525           break;
526         case '@':
527           d1 = Serial1.parseInt();
528           m1 = Serial1.parseInt();
529           a1 = Serial1.parseInt();
530           d2 = Serial1.parseInt();
531           m2 = Serial1.parseInt();
532           a2 = Serial1.parseInt();
533
534           str = String(dias_desde_01_01_2019(d1,m1,a1)) + ',' + String(dias_desde_01_01_2019(d2,m2,a2)); // Comando
535           Serial2.print(str);                                     // Envia comando para P1
536           Serial.print("\nEnviou para P1: "); Serial.print(str);
537           // Vamos aguardar a Parte 1 enviar o que foi pedido e mandar para o PC
538           if (!aguarda_serial(2,10000,40,8)) break;               // Serial2,10seg,40 caracteres,ponto de chamada 8
539           str = Serial2.readStringUntil('>');                     // Recebe de P1
540           Serial1.print("\n"+str);                                // Manda para o PC
541           Serial2.read();                                         // Acerta ponto de leitura
542           break;
543         default:
544           Serial.print("\nCodigo de comando ");
545           Serial.print(sp);
546           Serial.print(" invalido");
547           break;
548       }
549       limpa_buffer_serial(1,40);                                  // Limpa buffer
550     }
551
552     //_____________________________________________________________________________________________________________
553
554     // Verifica se foi apertado o botão toggle tudo
555     if (liga_desliga)                                             // Toggle tudo
556     {
557       Serial2.print("!(0)");                                      // Envia a p1 toggle tudo
558       Serial.print("\nEnviou pedido de toggle tudo a P1");
559       liga_desliga = false;
```

```
560     }
561
562     //_____________________________________________________________________________________________________________
563
564     // Verifica se veio solicitação do keypad para teclar comando de ligar/desligar grupo de refletores (códigos 1 a 12)
565     // Podemos utilizar os códigos > 12 para comandos de outra natureza.
566     if (pedido_keypad)
567     {
568       display_texto_lc(" Digite refletor + #",1,1);
569       clear_line(2);
570       clear_line(3);
571
572       if(entrada_teclado(entrada))                              // Recebe digitação. A leitura é encerrada
573       {                                                         // quando é digitado #
574         String este = String(entrada);
575         if (este.toInt() > 12)                                  // Outros comandos via keypad, programar aqui
576         {
577           // Comando > 12 , programe aqui se precisar retirando o comando abaixo
578           Serial.println("\nComando > 12 nao programado");
579           display_texto_lc(" Comando > 12",2,1);
580         }
581         else
582         {
583           display_string_lc(&este,2,1);                         // Display do que recebeu linha 2 coluna 1
584
585           Serial2.print("!(");
586           Serial2.print(entrada);                               // Envia a p1 o número do grupo a ser toggled
587           Serial2.print(')');
588
589           Serial.print("\nEnviou pedido de toggle a P1 - ");
590           Serial.print("!(");
591           Serial.print(entrada);                                // Envia a p1 o número do grupo a ser toggled
592           Serial.print(')');
593
594           display_texto_lc(" Enviou pedido a P1",3,1);
595         }
596       }
597       else
598       {
599         Serial.print("\nAlgo nao funcionou - comando keypad");
600         display_texto_lc(" Algo nao funcionou",2,1);
601       }
602       pedido_keypad = false;
```

```
  }

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

  if (timeElapsed1 > eeprom.inter1)                             // Executa a cada ciclo de duracao eeprom.inter1
  {
    Serial2.print("$(1)");                                      // Pede a P1 data e hora
    if (!aguarda_serial(2,1000,14,20))                          // Serial2,1seg,14 caracteres,ponto de chamada 20
    {
      Serial.print("\nP1 nao enviou data e hora");
    }
    else
    {
      dia = Serial2.parseInt();
      mes = Serial2.parseInt();
      ano = Serial2.parseInt();
      horas = Serial2.parseInt();
      minutos = Serial2.parseInt();
      segundos = Serial2.parseInt();
      tempo = String(dia) + '/' + String(mes) + '/' + String(ano) + ' ' + String(horas)
              + ':' + String(minutos) +  ':' + String(segundos);
    }

    //_______________________________________________ Dados INAs _______________________________________________________

    Serial2.print("$(2)");                                      // Vamos pedir os dados dos INA
    if (!recebe_inas())                                         // Recede os dados
    {
      Serial.print("\nP1 nao enviou dados das INAs");           // P1 não enviou dados  ponto de chamada 21
    }
    mostra_LCD03();                                             // Recede os dados e mostra no LCD03

    //___________________________________________________________________________________________________________________

    timeElapsed1 = 0;                                           // Reinicia ciclo
  }

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
```

```
646  //_____________________________________ Ciclo de tempo timeElapsed2 - default 5 minutos _____________________________________
647
648    if (timeElapsed2 > eeprom.inter2)                                  // Executa a cada ciclo de duracao eeprom.inter2
649    {
650
651      //_____________________________________________ Publicação MQTT ____________________________________________________
652
653      // Documentos para Json
654      StaticJsonDocument<100> doc1;
655      StaticJsonDocument<100> doc2;
656      StaticJsonDocument<100> doc3;
657      StaticJsonDocument<100> doc4;
658      StaticJsonDocument<100> doc5;
659
660      doc1["data"] = tempo;
661      JsonArray ina1 = doc1.createNestedArray("tracer1");
662      ina1.add(String(vina1, 2));
663      ina1.add(String(cina1, 2));
664      ina1.add(String(etracer1, 2));
665
666      serializeJson(doc1, buffer);
667
668      Serial.println(" ");
669      Serial.println(buffer);
670
671      MQTT.publish(tracer1, buffer);                                   // Publica MQTT tracer1
672
673      doc2["data"] = tempo;
674      JsonArray ina2 = doc2.createNestedArray("tracer2");
675      ina2.add(String(vina2, 2));
676      ina2.add(String(cina2, 2));
677      ina2.add(String(etracer2, 2));
678
679      serializeJson(doc2, buffer);
680
681      Serial.println(" ");
682      Serial.println(buffer);
683
684      MQTT.publish(tracer2, buffer);                                   // Publica MQTT tracer2
685
686      doc3["data"] = tempo;
687      JsonArray ina3 = doc3.createNestedArray("inversor");
688      ina3.add(String(vina3, 2));
```

```
689       ina3.add(String(cina3, 2));
690       ina3.add(String(einversor, 2));
691
692       serializeJson(doc3, buffer);
693
694       Serial.println(" ");
695       Serial.println(buffer);
696
697       MQTT.publish(inversor, buffer);                              // Publica MQTT inversor
698
699       doc4["data"] = tempo;
700       doc4["energia_total_in"] = String(etotalin, 2);
701       serializeJson(doc4, buffer);
702
703       Serial.println(" ");
704       Serial.println(buffer);
705
706       MQTT.publish(etotin, buffer);                                // Publica MQTT energia acumulada
707
708       doc5["data"] = tempo;
709       doc5["energia_total_out"] = String(einversor, 2);
710       serializeJson(doc5, buffer);
711
712       Serial.println(" ");
713       Serial.println(buffer);
714
715       MQTT.publish(etotout, buffer);                               // Publica MQTT energia consumida
716
717       //_____________________________________________ Publicação ThingSpeak ____________________________________________
718
719       // Define os valores para os campos do canal ThingSpeak
720       ThingSpeak.setField(1, vina1);
721       ThingSpeak.setField(2, cina1);
722       ThingSpeak.setField(3, vina2);
723       ThingSpeak.setField(4, cina2);
724       ThingSpeak.setField(5, vina3);
725       ThingSpeak.setField(6, cina3);
726       ThingSpeak.setField(7, etotalin);
727       ThingSpeak.setField(8, einversor);
728
729       // Define o status para o canal do ThingSpeak
730       if(cina1 > cina2)
731       {
```

```
732         myStatus = String("Corrente de carga tracer1 > tracer2");
733       }
734       else if(cina1 < cina2)
735       {
736         myStatus = String("Corrente de carga tracer2 > tracer1");
737       }
738       else if(einversor > etotalin)
739       {
740         myStatus = String("Energia consumida maior que energia acumulada");
741       }
742       else;
743
744       ThingSpeak.setStatus(myStatus);                                  // Grava o status
745
746       // Envia dados para o canal do ThingSpeak
747       int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
748       if(x == 200)
749       {
750         Serial.print("\nCanal atualizado com sucesso");
751       }
752       else
753       {
754         Serial.print("\nProblema fazendo o update do canal. Codigo de erro HTTP " + String(x));
755       }
756
757       //_____________________________________________________________________________________________________________
758
759       volt = adc.readVoltage(0);                                       // Lê a voltagem no ADC
760       volt *= 5.87359;                                                 // Escala em função do divisor de tensão
761
762       str = " Bat " + String(volt,2) + "V";                            // Monta String para display no LCD03
763       display_string_lc(&str,4,1);                                     // Faz o display
764       if (volt <= 9.6)                                                 // Verifica se tem que carregar a bateria
765       {
766         display_texto("  *RECARGA*");                                  // Aviso para recarregar baterias - display
767         toca_buzzer(5);                                                // Aviso para recarregar baterias - buzzer
768       }
769
770       //_____________________________________________________________________________________________________________
771
772       timeElapsed2 = 0;                                                // Reinicia ciclo
773     }
774   }
```

```

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@

void handle_OnConnect(void)
{
  Serial.print("\nCliente conectado");
  server.send(200, "text/html", SendHTML());
}

//______________________________________________________________________________________________________________________

// HTTP request: Grupo 0
void handle_grupo0(void)
{
  Serial.print("\nGrupo 0");
  Serial2.print("!(0)");                                        // Grupo 0 contém todos os refletores liga/desliga
  server.send(200, "text/html", SendHTML());
}

//______________________________________________________________________________________________________________________

// HTTP request: Grupo 1
void handle_grupo1(void)
{
  Serial.print("\nGrupo 1");
  Serial2.print("!(1)");
  server.send(200, "text/html", SendHTML());
}

//______________________________________________________________________________________________________________________

// HTTP request: Grupo 2
void handle_grupo2(void)
{
  Serial.print("\nGrupo 2");
  Serial2.print("!(2)");
  server.send(200, "text/html", SendHTML());
}

//______________________________________________________________________________________________________________________

```

```
818  // HTTP request: Grupo 3
819  void handle_grupo3(void)
820  {
821    Serial.print("\nGrupo 3");
822    Serial2.print("!(3)");
823    server.send(200, "text/html", SendHTML());
824  }
825
826  //_____________________________________________________________________________________________________________
827
828  // HTTP request: Grupo 4
829  void handle_grupo4(void)
830  {
831    Serial.print("\nGrupo 4");
832    Serial2.print("!(4)");
833    server.send(200, "text/html", SendHTML());
834  }
835
836  //_____________________________________________________________________________________________________________
837
838  // HTTP request: Grupo 5
839  void handle_grupo5(void)
840  {
841    Serial.print("\nGrupo 5");
842    Serial2.print("!(5)");
843    server.send(200, "text/html", SendHTML());
844  }
845
846  //_____________________________________________________________________________________________________________
847
848  // HTTP request: Grupo 6
849  void handle_grupo6(void)
850  {
851    Serial.print("\nGrupo 6");
852    Serial2.print("!(6)");
853    server.send(200, "text/html", SendHTML());
854  }
855
856  //_____________________________________________________________________________________________________________
857
858  // HTTP request: Grupo 7
859  void handle_grupo7(void)
860  {
```

```
861     Serial.print("\nGrupo 7");
862     Serial2.print("!(7)");
863     server.send(200, "text/html", SendHTML());
864   }
865
866   //________________________________________________________________________________________________________________________
867
868   // HTTP request: Grupo 8
869   void handle_grupo8(void)
870   {
871     Serial.print("\nGrupo 8");
872     Serial2.print("!(8)");
873     server.send(200, "text/html", SendHTML());
874   }
875
876   //________________________________________________________________________________________________________________________
877
878   // HTTP request: Grupo 9
879   void handle_grupo9(void)
880   {
881     Serial.print("\nGrupo 9");
882     Serial2.print("!(9)");
883     server.send(200, "text/html", SendHTML());
884   }
885
886   //________________________________________________________________________________________________________________________
887
888   // HTTP request: Grupo 10
889   void handle_grupo10(void)
890   {
891     Serial.print("\nGrupo 10");
892     Serial2.print("!(10)");
893     server.send(200, "text/html", SendHTML());
894   }
895
896   //________________________________________________________________________________________________________________________
897
898   // HTTP request: Grupo 11
899   void handle_grupo11(void)
900   {
901     Serial.print("\nGrupo 11");
902     Serial2.print("!(11)");
903     server.send(200, "text/html", SendHTML());
```

```
904  }
905
906  //_____________________________________________________________________________________________________________________
907
908  // HTTP request: Grupo 12
909  void handle_grupo12(void)
910  {
911    Serial.print("\nGrupo 12");
912    Serial2.print("!(12)");
913    server.send(200, "text/html", SendHTML());
914  }
915
916  //_____________________________________________________________________________________________________________________
917
918  // HTTP request: other
919  void handle_NotFound(void)
920  {
921    Serial.print("\nPage not found");
922    server.send(404, "text/plain", "Not found");
923  }
924
925  //_____________________________________________________________________________________________________________________
926
927  String SendHTML(void)
928  {
929    String html = "<!DOCTYPE html>\n";
930    html += "<html>\n";
931    html += "<head>\n";
932    html += "<title>CoMoSisFog</title>\n";
933    html += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">\n";
934    html += "</head>\n";
935    html += "<body>\n";
936    html += "<div align=\"center\">\n";
937    html += "<h1>CoMoSisFog</h1>\n";
938    html += "<br>\n";
939    html += "<form method=\"GET\">\n";
940    html += "<input type=\"button\" value=\"Grupo 1\" onclick=\"window.location.href='/grupo1'\">\n";
941    html += "<br><br>\n";
942    html += "<input type=\"button\" value=\"Grupo 2\" onclick=\"window.location.href='/grupo2'\">\n";
943    html += "<br><br>\n";
944    html += "<input type=\"button\" value=\"Grupo 3\" onclick=\"window.location.href='/grupo3'\">\n";
945    html += "<br><br>\n";
946    html += "<input type=\"button\" value=\"Grupo 4\" onclick=\"window.location.href='/grupo4'\">\n";
```

```
947     html += "<br><br>\n";
948     html += "<input type=\"button\" value=\"Grupo 5\" onclick=\"window.location.href='/grupo5'\">\n";
949     html += "<br><br>\n";
950     html += "<input type=\"button\" value=\"Grupo 6\" onclick=\"window.location.href='/grupo6'\">\n";
951     html += "<br><br>\n";
952     html += "<input type=\"button\" value=\"Grupo 7\" onclick=\"window.location.href='/grupo7'\">\n";
953     html += "<br><br>\n";
954     html += "<input type=\"button\" value=\"Grupo 8\" onclick=\"window.location.href='/grupo8'\">\n";
955     html += "<br><br>\n";
956     html += "<input type=\"button\" value=\"Grupo 9\" onclick=\"window.location.href='/grupo9'\">\n";
957     html += "<br><br>\n";
958     html += "<input type=\"button\" value=\"Grupo 10\" onclick=\"window.location.href='/grupo10'\">\n";
959     html += "<br><br>\n";
960     html += "<input type=\"button\" value=\"Grupo 11\" onclick=\"window.location.href='/grupo11'\">\n";
961     html += "<br><br>\n";
962     html += "<input type=\"button\" value=\"Grupo 12\" onclick=\"window.location.href='/grupo12'\">\n";
963     html += "</form>\n";
964     html += "</div>\n";
965     html += "</body>\n";
966     html += "</html>\n";
967     return html;
968   }
969
970   //_____________________________________________________________________________________________________________
971
972   bool recebe_inas(void)
973   {
974     // Formato: ln(valor) onde l=v voltagem,l=c corrente,l=p potencia,l=e energia
975     // Para v,c,p -> n=1 ina1,n=2 ina2,n=3 ina3,valor ponto flutuante
976     // Para e -> n=i etotalin, n=o etotalout
977
978     if (!aguarda_serial(2,1000,1,9)) return false;                // Serial2,1seg,1 caractere,ponto de chamada 9
979     while (Serial2.available() > 0)                               // Aguarda chegar alguma coisa
980     {
981       switch (Serial2.read())
982       {
983         case 'v':
984           switch (Serial2.parseInt())
985           {
986             case 1:
987               vina1 = Serial2.parseFloat();
988               break;
989             case 2:
```

```
990               vina2 = Serial2.parseFloat();
991               break;
992             case 3:
993               vina3 = Serial2.parseFloat();
994               break;
995           }
996           Serial2.read();
997         break;
998         case 'c':
999           switch (Serial2.parseInt())
1000          {
1001            case 1:
1002              cina1 = Serial2.parseFloat();
1003              break;
1004            case 2:
1005              cina2 = Serial2.parseFloat();
1006              break;
1007            case 3:
1008              cina3 = Serial2.parseFloat();
1009              break;
1010          }
1011          Serial2.read();
1012        break;
1013        case 'p':
1014          switch (Serial2.parseInt())
1015          {
1016            case 1:
1017              pina1 = Serial2.parseFloat();
1018              break;
1019            case 2:
1020              pina2 = Serial2.parseFloat();
1021              break;
1022            case 3:
1023              pina3 = Serial2.parseFloat();
1024              break;
1025          }
1026          Serial2.read();
1027        break;
1028        case 'e':
1029          switch (Serial2.read())
1030          {
1031            case 'i':
1032              etotalin = Serial2.parseFloat();
```

```
1033              break;
1034            case 'o':
1035              etotalout = Serial2.parseFloat();
1036              break;
1037          }
1038        break;
1039      }
1040    }
1041    return true;
1042  }
1043
1044  //_____________________________________________________________________________________________________________________
1045
1046  bool recebe_status(void)
1047  {
1048    // Recebe os dados dos grupos e refletores
1049    str = "";
1050    if (!aguarda_serial(2,1000,1,19)) return false;              // Ponto de chamada 19
1051    while (Serial2.available() > 0)                              // Aguarda chegar alguma coisa
1052    {
1053      while (Serial2.peek() != '#') str += String(Serial2.read());
1054      Serial1.print(str.c_str());                                // Envia linha ao PC
1055      Serial2.read();                                            // Descarta #
1056    }
1057    return true;
1058  }
1059
1060  //_____________________________________________________________________________________________________________________
1061
1062  bool pede_p1(uint8_t sensor)
1063  {
1064    // Solicita dados a P1
1065    switch (sensor)
1066    {
1067        case 1:                                                  // Pede data e hora a p1
1068          Serial2.print("$(1)");
1069
1070          if (!aguarda_serial(2,1000,14,7))                      // Serial2,1seg,14 caracteres,ponto de chamada 7
1071          {
1072            Serial.print("\nP1 nao enviou data e hora");
1073            return false;
1074          }
1075          dia = Serial2.parseInt();
```

```
1076          mes = Serial2.parseInt();
1077          ano = Serial2.parseInt();
1078          horas = Serial2.parseInt();
1079          minutos = Serial2.parseInt();
1080          segundos = Serial2.parseInt();
1081          envia_PC(1);
1082          limpa_buffer_serial(2,10);
1083          return true;
1084        case 2:
1085          Serial2.print("$(2)");                                  // Vamos pedir os dados dos INA
1086          if (!recebe_inas())                                    // Recede os dados
1087          {
1088            Serial.print("\nP1 nao enviou dados das INAs");      // P1 não enviou dados
1089            return false;
1090          }
1091          envia_PC(2);
1092          return true;
1093        case 3:
1094          Serial2.print("$(3)");                                  // Pede o status dos grupos e refletores
1095          if (!recebe_status())                                  // Recede os dados
1096          {
1097            Serial.print("\nP1 nao enviou dados de status");     // P1 não enviou dados
1098            return false;
1099          }
1100          return true;
1101        default:
1102          Serial.print("\nCodigo invalido - pede_p1");
1103          return false;
1104      }
1105    }
1106
1107    //_____________________________________________________________________________________________________________________
1108
1109    void envia_PC(unsigned int kk)
1110    {
1111      // Envia dados ao PC
1112      switch (kk)
1113      {
1114        case 1:                                                  // Data e hora
1115          Serial1.print("\n");
1116          Serial1.print(dia,DEC);
1117          Serial1.print('/');
1118          Serial1.print(mes,DEC);
```

```
1119        Serial1.print('/');
1120        Serial1.print(ano,DEC);
1121        Serial1.print(' ');
1122        Serial1.print(horas,DEC);
1123        Serial1.print(':');
1124        Serial1.print(minutos,DEC);
1125        Serial1.print(':');
1126        Serial1.print(segundos,DEC);
1127        break;
1128      case 2:                                                    // Dados das inas
1129        Serial1.print("\nvina1 = ");
1130        Serial1.print(vina1,2);
1131        Serial1.print(" cina1 = ");
1132        Serial1.print(cina1,2);
1133        Serial1.print(" pina1 = ");
1134        Serial1.print(pina1,2);
1135        Serial1.print("\nvina2 = ");
1136        Serial1.print(vina2,2);
1137        Serial1.print(" cina2 = ");
1138        Serial1.print(cina2,2);
1139        Serial1.print(" pina2 = ");
1140        Serial1.print(pina2,2);
1141        Serial1.print("\nvina3 = ");
1142        Serial1.print(vina3,2);
1143        Serial1.print(" cina3 = ");
1144        Serial1.print(cina3,2);
1145        Serial1.print(" pina3 = ");
1146        Serial1.print(pina3,2);
1147        Serial1.print("\netotalin = ");
1148        Serial1.print(etotalin,2);
1149        Serial1.print(" etotalout = ");
1150        Serial1.print(etotalout,2);
1151        break;
1152      default:
1153        Serial.print("\nCodigo invalido - envia_PC");
1154        break;
1155    }
1156  }
1157
1158  //_____________________________________________________________________________________________________________________
1159
1160  void mostra_LCD03(void)
1161  {
```

```
1162    // Mostra dados no LCD03
1163    cont1++;
1164    switch (cont1)
1165    {
1166      case 1:
1167        clear_line(1);
1168        clear_line(2);
1169        clear_line(3);
1170        display_texto_lc(" vina1 = ",1,1);
1171        display_float(vina1);
1172        display_texto_lc(" cina1 = ",2,1);
1173        display_float(cina1);
1174        display_texto_lc(" pina1 = ",3,1);
1175        display_float(pina1);
1176        break;
1177      case 2:
1178        clear_line(1);
1179        clear_line(2);
1180        clear_line(3);
1181        display_texto_lc(" vina2 = ",1,1);
1182        display_float(vina2);
1183        display_texto_lc(" cina2 = ",2,1);
1184        display_float(cina2);
1185        display_texto_lc(" pina2 = ",3,1);
1186        display_float(pina2);
1187        break;
1188      case 3:
1189        clear_line(1);
1190        clear_line(2);
1191        clear_line(3);
1192        display_texto_lc(" vina3 = ",1,1);
1193        display_float(vina3);
1194        display_texto_lc(" cina3 = ",2,1);
1195        display_float(cina3);
1196        display_texto_lc(" pina3 = ",3,1);
1197        display_float(pina3);
1198        break;
1199      case 4:
1200        clear_line(1);
1201        clear_line(2);
1202        clear_line(3);
1203        display_texto_lc(" etotalin = ",1,1);
1204        display_float(etotalin);
```

```
1205        display_texto_lc(" etotalout = ",2,1);
1206        display_float(etotalout);
1207        cont1 = 0;
1208        break;
1209      default:
1210        Serial.println("Contagem invalida");
1211        break;
1212    }
1213  }
1214
1215  //_____________________________________________________________________________________________________________
1216
1217  void IRAM_ATTR isr1()
1218  {
1219    // Rotina para processamento do interrupt do keypad
1220    pedido_keypad = true;
1221  }
1222
1223  //_____________________________________________________________________________________________________________
1224
1225  void IRAM_ATTR isr2()
1226  {
1227    // Rotina para processamento do interrupt de liga/desliga tudo
1228    liga_desliga = true;
1229  }
1230
1231  //_____________________________________________________________________________________________________________
1232
1233  void limpa_buffer_serial(uint8_t ser,uint8_t cmp)
1234  {
1235    // Limpa o buffer das seriais
1236    uint8_t i = 1;
1237
1238    while (i <= cmp)
1239    {
1240      if (ser == 1) {Serial1.read();i++;}
1241      else if (ser == 2) {Serial2.read();i++;}
1242      else return;
1243    }
1244    delay(1000);
1245  }
1246
1247  //_____________________________________________________________________________________________________________
```

```
1248
1249  bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde)
1250  {
1251    // Aguarda a serial enviar ncar caracteres no tempo intervalo mseg. Indica o ponto do programa que chamou - deonde
1252    timeElapsed = 0;                                                  // Inicia contagem de tempo
1253    while(timeElapsed < intervalo)                                    // Executa enquanto não passar o intervalo
1254    {
1255      if (ser == 1)                                                   // Serial1
1256      {
1257        if (Serial1.available() < ncar);
1258        else return true;
1259      }
1260      else if (ser == 2)                                              // Serial2
1261      {
1262        if (Serial2.available() < ncar);
1263        else return true;
1264      }
1265      else                                                            // Serial inválida
1266      {
1267        Serial.print("\nSerial ");
1268        Serial.print(ser);
1269        Serial.print(" invalida, chamada de ");
1270        Serial.print(deonde);
1271        toca_buzzer(5);
1272        return false;
1273      }
1274    }
1275    Serial.print("\nTime Out esperando Serial");                      // Time out
1276    Serial.print(ser);
1277    Serial.print(" chamada de ");
1278    Serial.print(deonde);
1279    toca_buzzer(5);
1280    return false;
1281  }
1282
1283  //_____________________________________________________________________________________________________________________
1284
1285  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano)
1286  {
1287    // Retorna o número de dias desde 01/01/2019
1288    unsigned int an,nd = 0;
1289    unsigned int dd[14] = {0,0,31,59,90,120,151,181,212,243,273,304,334,365};
1290
```

```
1291    for (an=2019;an<ano;an++)
1292    {
1293      if ((an % 4) == 0) nd += 366;                          // É ano bissexto (vale até 2100)
1294      else nd += 365;
1295    }
1296    if (((an % 4) == 0) && (mes > 2)) nd += (dd[mes] + dia + 1);  // É ano bissexto (vale até 2100)
1297    else nd += (dd[mes] + dia);
1298    return nd;
1299  }
1300
1301  //_____________________________________________________________________________________________________________
1302
1303  void toca_buzzer(unsigned int seg)
1304  {
1305    // Toca o buzzer por seg segundos
1306    long intervalo = (long)(seg * 1000);
1307
1308    timeElapsed = 0;                                         // Inicia contagem de tempo
1309    while(timeElapsed < intervalo)                           // Executa enquanto não passar o intervalo
1310    {
1311      digitalWrite(buzzer, HIGH);
1312      delay(500);
1313      digitalWrite(buzzer, LOW);
1314      delay(500);
1315    }
1316  }
1317
1318  //_____________________________________________________________________________________________________________
1319
1320  void conecta_MQTT(void)
1321  {
1322    // Conecta MQTT
1323    if (!MQTT.connected())                                   // Se não estiver conectado
1324    {
1325      long now = millis();
1326      if (now - lastReconnectAttempt > 5000)
1327      {
1328        lastReconnectAttempt = now;
1329        if (reconnect())                                     // Tentativa de reconexão
1330        {
1331          Serial.print("\nBroker MQTT conectado");
1332          lastReconnectAttempt = 0;
1333        }
```

```
1334      }
1335    }
1336  }
1337
1338  //_____________________________________________________________________________________________________________
1339
1340  bool reconnect()
1341  {
1342    // Reconecta MQTT
1343    if (MQTT.connect("quiosque",ID_MQTT,PSW_MQTT))
1344    {
1345      MQTT.publish(aviso1,"conectado");                          // Quando reconectado, publica aviso
1346    }
1347    return MQTT.connected();
1348  }
1349
1350  //_____________________________________________________________________________________________________________
1351
1352  void conecta_wifi(unsigned char pr)
1353  {
1354    // Conecta WiFi
1355    WiFi.begin(ssid, password);                                  // Inicia WiFi
1356    while (WiFi.status() != WL_CONNECTED)                        // Aguarda conexão
1357    {
1358      delay(500);
1359      Serial.println("Conectando WiFi..");
1360    }
1361    Serial.print("Connectado na rede ");                         // Conectado
1362    Serial.println(ssid);
1363    if (pr == 1)
1364    {
1365      IPAddress ip = WiFi.localIP();                             // Imprime o IP
1366      Serial.print("IP: ");
1367      Serial.println(ip);
1368      long rssi = WiFi.RSSI();                                   // Imprime o nível do sinal
1369      Serial.print("Nivel do sinal (RSSI): ");
1370      Serial.println(rssi);
1371    }
1372  }
1373
1374  //_____________________________________________________________________________________________________________
1375
1376  bool recebido_P1(unsigned long tp,uint8_t lg)
```

```
1377  {
1378    // Aguarda ACK de P1
1379    if (!aguarda_serial(2,tp,1,lg)) return false;              // Serial2,tp seg,1 caractere,ponto de chamada,time out
1380    else if (Serial2.read() != 0x6) return false;              // Não é ACK
1381    else return true;                                          // É ACK
1382  }
1383
1384  //_____________________________________________________________________________________________________________________
1385
```