```
/*****************************************************************************************************
* 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 Filippo Pardini - filippo@robotica.eng.br - http://blog.robotica.eng.br/                           *
*****************************************************************************************************/
/*
  Projeto CoMoSisFog V3.1.0 (20/11/2019)
  * 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. Os controladores estão interligados ao banco de baterias através do stringbox
  * de proteção de forma invertida entre eles (um controlador alimenta o + da 1ª bateria e o - da última bateria e o
  * outro controlador alimenta o - da 1ª bateria e o + da última bateria). 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
```

```
37      * no display Oled e enviando, via Xbee, para o ESP32 da Parte 2. Grava no SD as voltagens máximas e mínimas das INAs
38      * e energia gerada e consumida em 24h (para análise e calibração de ciclos liga/desliga). Em função desses dados, da
39      * fotocélula, dos dados configurados pelo ESP32 remoto (Parte 2 via Xbee), controla e disponibiliza energia em
40      * horários pré-configurados para os refletores, executa comandos e solicitações do ESP32 remoto e gerencia a energia
41      * fornecida pelas baterias, otimizando-as.
42
43      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 <WiFi.h>                                       // WiFi
60  #include <DNSServer.h>                                  // DNS
61  #include <WebServer.h>                                  // Server
62  #include <Wire.h>                                       // I2C
63  #include <LCD03_N1.h>                                   // LCD03
64  #include <elapsedMillis.h>                              // Intervalos de tempo
65  #include <Nanoshield_ADC.h>                             // Nanoshield ADC ADS1115
66  #include <WiFi.h>                                       // WiFi
67  #include <PubSubClient.h>                               // MQTT
68  #include "ThingSpeak.h"                                 // ThingSpeak
69  #include <ArduinoJson.h>                                // Json
70
71  //_____________________________________________________________________________________________________________
72
```

```
#define i2c_addr_eeprom 0x50                                    // Endereco I2C da 24LC256

template <class T> int eeWrite(int ee, const T& value)          // Classe genérica para gravação da eeprom
{                                                               // ee - posição na eeprom, value - estrutura de dados
  const byte* p = (const byte*)(const void*)&value;             // Pointer para a estrutura
  int i;                                                        // Contador
  Wire.beginTransmission(i2c_addr_eeprom);                      // Inicia transmissão I2C
  Wire.write((int)(ee >> 8)); // MSB                            // Envia MSB da posição
  Wire.write((int)(ee & 0xFF)); // LSB                          // Envia LSB da posição
  for (i = 0; i < sizeof(value); i++)                           // Para todos os bytes da estrutura
  Wire.write(*p++);                                             // Envia o byte
  Wire.endTransmission();                                       // Encerra transmissão I2C
  return i;                                                     // Retorna o número de bytes gravados
}

template <class T> int eeRead(int ee, T& value)                 // Classe genérica para leitura da eeprom
{                                                               // ee - posição na eeprom, value - estrutura de dados
  byte* p = (byte*)(void*)&value;                               // Pointer para a estrutura
  int i;                                                        // Contador
  Wire.beginTransmission(i2c_addr_eeprom);                      // Inicia transmissão I2C
  Wire.write((int)(ee >> 8)); // MSB                            // Envia MSB da posição
  Wire.write((int)(ee & 0xFF)); // LSB                          // Envia LSB da posição
  Wire.endTransmission();                                       // Encerra transmissão I2C
  Wire.requestFrom(i2c_addr_eeprom,sizeof(value));              // Solicita à eeprom o número de bytes da estrutura
  for (i = 0; i < sizeof(value); i++)                           // Para esse número de bytes
  if(Wire.available())                                          // Se tem dados no buffer
  *p++ = Wire.read();                                           // Lê o byte para a estrutura
  return i;                                                     // Retorna o número de bytes lidos
}

struct eeprom                                                   // Estrutura de dados na eeprom
{
  unsigned long valido;                                         // Código para validar dados
  unsigned long inter1;                                         // Intervalo de tempo para "elapsedMillis timeElapsed1;"
  unsigned long inter2;                                         // Intervalo de tempo para "elapsedMillis timeElapsed2;"
  unsigned int cod_mon;                                         // Código para imprimir ou não no monitor serial
```

```
109  } eeprom;
110
111  //_____________________________________________________________________________________________________________________
112
113  // WiFi
114  const char* ssid = "Pardini";                                    // Nome da rede WiFi
115  const char* password = "sitiolacicala";                          // Senha da rede WiFi
116
117  //_____________________________________________________________________________________________________________________
118
119  // MQTT
120  const char* BROKER_MQTT = "m16.cloudmqtt.com";                   // URL do broker MQTT
121  int BROKER_PORT = 17962;                                         // Porta do Broker MQTT
122  #define ID_MQTT  "lrffixsc"                                      // User ID
123  #define PSW_MQTT "JUp-9gCLKD2X"                                  // Password
124  #define tracer1 "comosisfog/tracer1"                             // Topico tracer1
125  #define tracer2 "comosisfog/tracer2"                             // Topico tracer2
126  #define inversor "comosisfog/inversor"                           // Topico inversor
127  #define etotin "comosisfog/etotin"                               // Topico energia total acumulada
128  #define etotout "comosisfog/etotout"                             // Topico energia total consumida
129  #define aviso1 "comosisfog/aviso1"                               // Topico aviso1
130  #define aviso2 "comosisfog/aviso2"                               // Topico aviso2
131
132  //_____________________________________________________________________________________________________________________
133
134  // Variaveis e constantes
135  #define SDApin 21                                                // I2C SDA
136  #define SCLpin 22                                                // I2C SCL
137  #define RX1pin 16                                                // Pino RX2
138  #define TX1pin 17                                                // pino TX2
139  #define RX2pin 4                                                 // UART2 RX
140  #define TX2pin 2                                                 // UART2 TX
141  #define liga_desliga_tudo 26                                     // Pino de interrupt para ligar ou desligar tudo
142  #define keypadpin 25                                             // Pino de interrupt para o keypad
143  #define buzzer 27                                                // Pino para acionamento do buzzer
144
```

```
145  char buffer[100];                                                  // Buffer
146  char entrada[21];                                                  // Vetor para receber digitação (1 linha do LCD03)
147  const byte DNS_PORT = 53;                                          // DNS
148  const char *ssidsoftAP = "CoMoSisFog";                             // WiFi.softAP
149  const char *passwordsoftAP = "villaadriana";                       // WiFi.softAP
150  unsigned long key = 2863311530;                                    // Código eeprom válida
151  volatile bool pedido_keypad;                                       // Variável para interrupt do keypad
152  volatile bool liga_desliga;                                        // Variável para o interrupt de liga/desliga tudo
153  long lastReconnectAttempt = 0;                                     //  Tempo última conexão MQTT
154  String versao = "V3.1.0";                                          // ************* Versão *********************************
155
156  unsigned int dia;
157  unsigned int mes;
158  unsigned int ano;
159  unsigned int horas;
160  unsigned int minutos;
161  unsigned int segundos;
162
163  unsigned int d1;
164  unsigned int m1;
165  unsigned int a1;
166  unsigned int d2;
167  unsigned int m2;
168  unsigned int a2;
169
170  float vina1,cina1,pina1,vina2,cina2,pina2,vina3,cina3,pina3,etracer1,etracer2,etotalin,einversor,etotalout,volt;
171
172  String tempo,str;
173
174  unsigned int k;
175  uint8_t cont1;
176  char sp;
177
178  //_____________________________________________________________________________________________________________________
179
180  IPAddress local_ip(192, 168, 5, 1);                                // IP do servidor AP
```

```
181  IPAddress gateway(192, 168, 5, 1);                                  // Gateway do servidor AP
182  IPAddress subnet(255, 255, 255, 0);                                 // Subnet do servidor AP
183
184  //_____________________________________________________________________________________________________________
185
186  // Instancias
187  WiFiClient wificlient;                                              // Instancia WiFi MQTT
188  WiFiClient wifiTS;                                                  // Instancia WiFi ThingSpeak
189  PubSubClient MQTT(BROKER_MQTT,BROKER_PORT,wificlient);              // Instancia MQTT passando o objeto wificlient
190  DNSServer dnsServer;                                                // Instancia DNSServer para nome de dominio
191  WebServer server(80);                                               // Instancia WebServer para serevidor AP
192  elapsedMillis timeElapsed1;                                         // Instancia elapsedMillis para eeprom.inter1
193  elapsedMillis timeElapsed2;                                         // Instancia elapsedMillis para eeprom.inter2
194  elapsedMillis timeElapsed;                                          // Instancia elapsedMillis generico
195  Nanoshield_ADC adc(0x4B);                                           // Instancia Nanoshield_ADC
196
197  //_____________________________________________________________________________________________________________
198
199  // ThingSpeak
200  unsigned long myChannelNumber = 726200;                             // ThingSpeak ID do canal
201  const char * myWriteAPIKey = "MPVSYRDHBVLOCHJ8";                    // ThingSpeak chave de escrita
202  String myStatus = "";                                               // ThingSpeak para teste
203
204  //_____________________________________________________________________________________________________________
205
206  // Funções
207  void conecta_MQTT(void);                                            // Conecta MQTT
208  bool reconnect();                                                   // Reconecta MQTT
209  void conecta_wifi(unsigned char pr);                                // Conecta WiFi
210  bool recebe_status(void);                                           // Recebe os dados dos grupos e refletores
211  bool recebe_inas(void);                                             // Recebe dados das INAs
212  bool pede_p1(uint8_t sensor);                                       // Pede dados à Parte 1
213  void envia_PC(unsigned int kk);                                     // Envia dados ao PC
214  void mostra_LCD03(void);                                            // Mostra dados no LCD03
215  void limpa_buffer_serial(uint8_t ser,uint8_t cmp);                  // Limpa buffer das seriais
216  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()
{
  Serial.begin(115200);                                         // Monitor serial
  while (!Serial);
  str = "\nCoMoSisFog " + versao +  " parte 2";
  Serial.print(str);                                            // Identificação

  Wire.begin(SDApin,SCLpin);                                    // I2C ESP32 - SDA 21 SCL 22

```

```
253     adc.begin();                                                    // Inicia o Nanoshield_ADC
254     adc.setGain(GAIN_ONE);                                          // Define o ganho do ADC
255     adc.setSampleRate(860);                                         // Define amostragem do ADC
256
257     LCD03_begin(0xC6,10);                                           // Inicia o LCD03
258     Serial.println("LCD03 inicializado");
259
260     backlight_on;                                                   // Luz de fundo
261     clear_screen;                                                   // Limpa
        tela
262     hide_cursor;                                                    // Esconde cursor
263     str = " CoMoSisFog " + versao;
264     display_string_lc(&str,1,1);                                    // Versão do sistema
265     display_texto_lc(" Filippo Pardini",2,1);                       // Autor
266     delay (1000);
267
268     conecta_wifi(1);                                                // Conecta WiFi
269
270     ThingSpeak.begin(wifiTS);                                       // Inicia ThingSpeak
271
272     WiFi.softAP(ssidsoftAP, passwordsoftAP, 13, false, 1);          // Parametros de acesso ao servidor AP
273     delay(100);
274     WiFi.softAPConfig(local_ip, gateway, subnet);                   // Parametros de acesso ao servidor AP
275
276     dnsServer.setTTL(300);                                          // Servidor DNS
277     dnsServer.setErrorReplyCode(DNSReplyCode::ServerFailure);       // Servidor DNS
278     dnsServer.start(DNS_PORT, "comosisfog.net", local_ip);          // Servidor DNS
279
280     Serial2.begin(115200, SERIAL_8N1, RX1pin, TX1pin);              // Serial1 para Xbee - ESP32 - RX 16 TX 17
281     Serial2.setTimeout(10000);                                      // Time out
282     Serial.print("\nSerial para Xbee inicializada");
283
284     Serial1.begin(115200, SERIAL_8N1, RX2pin, TX2pin);              // Serial2 para PC - ESP32 - RX 4 TX 2
285     Serial1.setTimeout(10000);                                      // Time out
286     Serial.print("\nSerial para PC inicializada");
287
```

```
288     // Atribuição dos handles para o servidor AP
289
290     server.on("/", HTTP_GET, handle_OnConnect);
291     server.on("/grupo0", HTTP_GET, handle_grupo0);
292     server.on("/grupo1", HTTP_GET, handle_grupo1);
293     server.on("/grupo2", HTTP_GET, handle_grupo2);
294     server.on("/grupo3", HTTP_GET, handle_grupo3);
295     server.on("/grupo4", HTTP_GET, handle_grupo4);
296     server.on("/grupo5", HTTP_GET, handle_grupo5);
297     server.on("/grupo6", HTTP_GET, handle_grupo6);
298     server.on("/grupo7", HTTP_GET, handle_grupo7);
299     server.on("/grupo8", HTTP_GET, handle_grupo8);
300     server.on("/grupo9", HTTP_GET, handle_grupo9);
301     server.on("/grupo10", HTTP_GET, handle_grupo10);
302     server.on("/grupo11", HTTP_GET, handle_grupo11);
303     server.on("/grupo12", HTTP_GET, handle_grupo12);
304     server.onNotFound(handle_NotFound);
305
306     server.begin();                                          // Inicializa servidor AP
307     Serial.print("\nweb server e DNS inicializados");
308
309   //_____________________________________________________________________________________________________________
310
311   // Força eeprom invalida para inicializar default
312   // Depois lembrar de retirar este codigo
313     //eeprom.valido = 0;
314     //eeprom.inter1 = 0;
315     //eeprom.inter2 = 0;
316     //eeprom.cod_mon = 0;
317     //eeWrite(0,eeprom);
318     //delay (30);
319   //_____________________________________________________________________________________________________________
320
321     eeRead(0,eeprom);                                        // Le eeprom
322     delay (30);
323
```

```
324     if (eeprom.valido != key)                                       // Eeprom invalida, força reinicializacao default
325     {
326       eeprom.inter1 = 60000;                                        // Default - 1 minuto
327       eeprom.inter2 = 300000;                                       // Default - 5 minutos
328       eeprom.cod_mon = 1;                                           // Habilita monitor serial
329       eeprom.valido = key;                                          // Valida eeprom
330
331       eeWrite(0,eeprom);                                            // Grava eeprom
332       delay (30);
333       Serial.print("\neeprom default");
334     }
335
336     pinMode(keypadpin, INPUT_PULLUP);                               // Pino a ser acionado quando for teclar no keypad
337     attachInterrupt(digitalPinToInterrupt(keypadpin),isr1,FALLING); // Define interrupt no pino keypadpin
338     pinMode(liga_desliga_tudo, INPUT_PULLUP);                       // Pino a ser acionado quando for toggle tudo
339     attachInterrupt(digitalPinToInterrupt(liga_desliga_tudo),isr2,FALLING); // Define interrupt no pino liga_desliga_tudo
340     pinMode(buzzer, OUTPUT);                                        // Pino para acionar o buzzer
341     digitalWrite(buzzer, LOW);                                      // Inicialmente desativado
342
343     timeElapsed1 = 0;
344     timeElapsed2 = 0;
345     timeElapsed = 0;
346     cont1 = 0;
347     pedido_keypad = false;
348     liga_desliga = false;
349   }
350
351   //_____________________________________________________________________________________________________________________
352
353   void loop()
354   {
355     conecta_MQTT();                                                 // Conecta com o Broker MQTT
356     MQTT.loop();
357
358     dnsServer.processNextRequest();                                 // DNS
359     server.handleClient();
```

```

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//______________________________________________Processa Continuamente _________________________________________________

  if (Serial2.available() > 10)
  {
    // Verifica se P1 enviou algum texto <....>
    if (Serial2.read() == '<')                                  // Verifica se a P1 enviou mensagem - formato - <texto>
    {
      str = " " + Serial2.readStringUntil('>');
      clear_screen;
      display_string_lc(&str,1,1);                              // Display no LCD03
      toca_buzzer(3);                                           // Toca buzzer
      while (read_keypad() != '#')                              // Repete atá digitar #
      {
        delay(3000);                                            // Aguarda
        toca_buzzer(3);                                         // Toca buzzer
      }
    }
  }

  //____________________________________________________________________________________________________________________

  if (Serial1.available() >= 4)                                 // Pc enviou algo?
  {
    // Verifica se o PC enviou alguma coisa: comando, coniguração p1, coniguração p2, configuração dos grupos de
    // refletores, configuração de seus modos ou solicitação de dados. Ou a Parte 1 enviou mensagem.
    // Indicadores válidos:
    // '!' - Comando toggle dos grupos de refletores - formato - !(g) - onde g é o número do grupo.  g = 0 => tudo
    // '$' - Solicitação de dados - formato - $(i) - onde i=1 data-hora, i=2 dados INAs e energias, i=3 status dos
    //       grupos e refletores
    // '@' - Solicitação de dados - formato - @(di,df) - onde di - data de início, df - data de fim, ambos no formato
    //       dia,mes,ano. Ex: @(1,8,2019/31,8,2019) vai ler o SD e enviar os dados relativos a esse período
    // '%' - Configuração de ciclos em p2 - formato - %n(v) - onde n=1 eeprom.inter1, n=2 eeprom.inter2,
```

```
396       //        n=3 eepro.cod_mon, v = 0 ou 1 ou valor em minutos
397       // '#' - Configuração de ciclos em p1 - formato - #n(v) - onde n=1 inter1, n=2 inter2,
398       //        n=3 cod_mon, v = 0 ou 1 ou valor em minutos
399       // '&' - Configuração dos grupos de refletores - formato - &n(r,...,r) - onde n é o número do grupo
400       //        e r o número do refletor
401       // '*' - Configuração dos horarios liga/desliga dos grupos e modo de operação de cada grupo
402       //        Modo de operação: ativo -> sim(1)/não(0), automatico -> fotocelula, programado -> hora liga/desliga,
403       //        manual -> comando de toggle via p2,PC ou celular - formato - *n(a,b,c,d) - onde n é o número do grupo
404       //        a=0 -> inativo, a=1 => ativo, b=1 -> automático(fotocelula), b=2 -> manual(comando PC,p2,celular),
405       //        b=3 -> programado (c e d só existem neste caso), c -> hora liga (formato hh:mm hh -> hora formato 24,
406       //        mm -> minutos), d -> hora desliga (idem hora liga)
407
408       sp = Serial1.peek();
409
410       switch (sp)
411       {
412         case '%':
413           if (!aguarda_serial(1,1000,5,1)) break;                  // Serial1,1seg,5 caracteres,ponto de chamada 1
414           switch(Serial1.parseInt())                               // Configura ciclos em p2
415           {
416             case 1:                                                // Interval1_p2 em minutos
417               eeprom.inter1 = Serial1.parseInt() * 60000;          // Passa para milisegundos
418               break;
419             case 2:                                                // Interval2_p2 em minutos
420               eeprom.inter2 = Serial1.parseInt() * 60000;          // Passa para milisegundos
421               break;
422             case 3:                                                // eeprom.cod_mon
423               eeprom.cod_mon = Serial1.parseInt();
424               break;
425             default:
426               Serial.println("Codigo invalido comando %");
427               break;
428           }
429           eeWrite(0,eeprom);                                       // Grava eeprom
430           delay (30);
431           Serial.print("\neeprom gravada P2");
```

```
432           MQTT.publish(aviso2,"eeprom P2 configurada");           // Avisa no MQTT
433           break;
434         case '$':
435           if (!aguarda_serial(1,1000,4,2)) break;                 // Serial1,1seg,4 caracteres,ponto de chamada 2
436           k = Serial1.parseInt();
437           if (!pede_p1(k))
438           {
439             Serial.print("\nSem resposta de P1 - ponto de chamada 14"); // ponto de chamada 14
440           }
441           break;
442         case '!':
443           if (!aguarda_serial(1,100,4,3)) break;                  // Serial1,1seg,4 caracteres,ponto de chamada 3
444           str = Serial1.readStringUntil(')') + ')';               // Lê comando
445           Serial2.print(str);                                     // Envia comando para P1
446           if (!recebido_P1(1000,10))                              // Serial2,1seg,1 caractere,ponto de chamada 10
447           {
448             Serial.print("\nSem resposta de P1 - ponto de chamada 15"); // ponto de chamada 15
449             break;
450           }
451           Serial.print("\nEnviou para P1: "); Serial.print(str);
452           break;
453         case '#':
454           if (!aguarda_serial(1,1000,5,4)) break;                 // Serial1,1seg,4 caracteres,ponto de chamada 4
455           str = Serial1.readStringUntil(')') + ')';               // Lê comando
456           Serial2.print(str);                                     // Envia comando para P1
457           if (!recebido_P1(1000,11))                              // Serial2,1seg,1 caractere,ponto de chamada 11
458           {
459             Serial.print("\nSem resposta de P1 - ponto de chamada 16"); // ponto de chamada 16
460             break;
461           }
462           Serial.print("\nEnviou para P1: "); Serial.print(str);
463           MQTT.publish(aviso1,"eeprom P1 configurada");           // Avisa no MQTT
464           break;
465         case '&':
466           if (!aguarda_serial(1,1000,5,5)) break;                 // Serial1,1seg,4 caracteres,ponto de chamada 5
467           str = Serial1.readStringUntil(')') + ')';               // Lê comando
```

```
468           Serial2.print(str);                                    // Envia comando para P1
469           if (!recebido_P1(1000,12))                             // Serial2,1seg,1 caractere,ponto de chamada 12
470           {
471             Serial.print("\nSem resposta de P1 - ponto de chamada 17"); // ponto de chamada 17
472             break;
473           }
474           Serial.print("\nEnviou para P1: "); Serial.print(str);
475           break;
476         case '*':
477           if (!aguarda_serial(1,1000,6,6)) break;                // Serial1,1seg,6 caracteres,ponto de chamada 6
478           str = Serial1.readStringUntil(')') + ')';              // Lê comando
479           Serial2.print(str);                                    // Envia comando para P1
480           if (!recebido_P1(1000,13))                             // Serial2,1seg,1 caractere,ponto de chamada 13
481           {
482             Serial.print("\nSem resposta de P1 - ponto de chamada 18"); // ponto de chamada 18
483             break;
484           }
485           Serial.print("\nEnviou para P1: "); Serial.print(str);
486           break;
487         case '@':
488           d1 = Serial1.parseInt();
489           m1 = Serial1.parseInt();
490           a1 = Serial1.parseInt();
491           d2 = Serial1.parseInt();
492           m2 = Serial1.parseInt();
493           a2 = Serial1.parseInt();
494
495           str = String(dias_desde_01_01_2019(d1,m1,a1)) + ',' + String(dias_desde_01_01_2019(d2,m2,a2)); // Comando
496           Serial2.print(str);                                    // Envia comando para P1
497           Serial.print("\nEnviou para P1: "); Serial.print(str);
498           // Vamos aguardar a Parte 1 enviar o que foi pedido e mandar para o PC
499           if (!aguarda_serial(2,1000,40,8)) break;               // Serial2,1seg,40 caracteres,ponto de chamada 8
500           str = Serial2.readStringUntil('>');                    // Recebe de P1
501           Serial1.print("\n"+str);                               // Manda para o PC
502           Serial2.read();                                        // Acerta ponto de leitura
503           break;
```

```
504         default:
505           Serial.print("\nCodigo invalido comando @");
506           break;
507       }
508       limpa_buffer_serial(1,40);                                  // Limpa buffer
509     }
510
511     //_____________________________________________________________________________________________________________________
512
513     // Verifica se foi apertado o botão toggle tudo
514     if (liga_desliga)                                             // Toggle tudo
515     {
516       Serial2.print("!(0)");                                      // Envia a p1 toggle tudo
517       Serial.print("\nEnviou pedido de toggle tudo a P1");
518       liga_desliga = false;
519     }
520
521     //_____________________________________________________________________________________________________________________
522
523     // Verifica se veio solicitação do keypad para teclar comando de ligar/desligar grupo de refletores (códigos 1 a 12)
524     // Podemos utilizar os códigos > 12 para comandos de outra natureza.
525     if (pedido_keypad)
526     {
527       if(entrada_teclado(entrada))                                // Recebe digitação. A leitura é encerrada
528       {                                                           // quando é digitado #
529         String este = String(entrada);
530         if (este.toInt() > 12)                                    // Outros comandos via keypad, programar aqui
531         {
532           // Comando > 12 , programe aqui se precisar retirando o comando abaixo
533           Serial.print("\nComando > 12 nao programado");
534         }
535         else
536         {
537           display_vectorchar_lc(entrada,1,1);                     // Display do que recebeu linha 1 coluna 1
538           Serial_print(entrada,1,5);                              // Imprime na serial o que recebeu a partir da posição 1
539                                                                   // por 5 posições
```

```

        Serial2.print("!(");
        Serial2.print(entrada);                                 // Envia a p1 o número do grupo a ser toggled
        Serial2.print(')');
        Serial.print("\nEnviou pedido de toggle a P1");
      }
    }
    else
    {
      Serial.print("\nAlgo nao funcionou - comando keypad");
    }
    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");
    }
    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);

    //_________________________________________________________________________________________________________________
```

```

      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
      }

      //__________________________________________________________________________________________________________________

      volt = adc.readVoltage(0);                                    // Lê a voltagem no ADC
      volt *= 5.87359;                                              // Escala em função do divisor de tensão

      str = " " + String(volt,2) + "V";                             // Monta String para display no LCD03
      display_string_lc(&str,4,1);                                  // Faz o display
      if (volt <= 9.6)                                              // Verifica se tem que carregar a bateria
      {
        display_texto("  *RECARREGAR*");                            // Aviso para recarregar baterias - display
        toca_buzzer(3);                                             // Aviso para recarregar baterias - buzzer
      }

      //__________________________________________________________________________________________________________________

      timeElapsed1 = 0;                                             // Reinicia ciclo
    }

  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
  //___________________________________ Ciclo de tempo timeElapsed2 - default 5 minutos __________________________________

    if (timeElapsed2 > eeprom.inter2)                               // Executa a cada ciclo de duracao eeprom.inter2
    {
      //_______________________________________________ Dados INAs _______________________________________________________

      mostra_LCD03();                                               // Recede os dados e mostra no LCD03

```

```
612       //_____________________________________________ Publicação MQTT _____________________________________________
613
614       // Documentos para Json
615       StaticJsonDocument<100> doc1;
616       StaticJsonDocument<100> doc2;
617       StaticJsonDocument<100> doc3;
618       StaticJsonDocument<100> doc4;
619       StaticJsonDocument<100> doc5;
620
621       doc1["data"] = tempo;
622       JsonArray ina1 = doc1.createNestedArray("tracer1");
623       ina1.add(String(vina1, 2));
624       ina1.add(String(cina1, 2));
625       ina1.add(String(etracer1, 2));
626
627       serializeJson(doc1, buffer);
628
629       Serial.println(" ");
630       Serial.println(buffer);
631
632       MQTT.publish(tracer1, buffer);                          // Publica MQTT tracer1
633
634       doc2["data"] = tempo;
635       JsonArray ina2 = doc2.createNestedArray("tracer2");
636       ina2.add(String(vina2, 2));
637       ina2.add(String(cina2, 2));
638       ina2.add(String(etracer2, 2));
639
640       serializeJson(doc2, buffer);
641
642       Serial.println(" ");
643       Serial.println(buffer);
644
645       MQTT.publish(tracer2, buffer);                          // Publica MQTT tracer2
646
647       doc3["data"] = tempo;
```

```
648       JsonArray ina3 = doc3.createNestedArray("inversor");
649       ina3.add(String(vina3, 2));
650       ina3.add(String(cina3, 2));
651       ina3.add(String(einversor, 2));
652
653       serializeJson(doc3, buffer);
654
655       Serial.println(" ");
656       Serial.println(buffer);
657
658       MQTT.publish(inversor, buffer);                          // Publica MQTT inversor
659
660       doc4["data"] = tempo;
661       doc4["energia_total_in"] = String(etotalin, 2);
662       serializeJson(doc4, buffer);
663
664       Serial.println(" ");
665       Serial.println(buffer);
666
667       MQTT.publish(etotin, buffer);                            // Publica MQTT energia acumulada
668
669       doc5["data"] = tempo;
670       doc5["energia_total_out"] = String(einversor, 2);
671       serializeJson(doc5, buffer);
672
673       Serial.println(" ");
674       Serial.println(buffer);
675
676       MQTT.publish(etotout, buffer);                           // Publica MQTT energia
          consumida
677
678       //_____________________________________________ Publicação ThingSpeak _____________________________________________
679
680       // Define os valores para os campos do canal ThingSpeak
681       ThingSpeak.setField(1, vina1);
682       ThingSpeak.setField(2, cina1);
```

```
683       ThingSpeak.setField(3, vina2);
684       ThingSpeak.setField(4, cina2);
685       ThingSpeak.setField(5, vina3);
686       ThingSpeak.setField(6, cina3);
687       ThingSpeak.setField(7, etotalin);
688       ThingSpeak.setField(8, einversor);
689
690       // Define o status para o canal do ThingSpeak
691       if(cina1 > cina2)
692       {
693         myStatus = String("Corrente de carga tracer1 > tracer2");
694       }
695       else if(cina1 < cina2)
696       {
697         myStatus = String("Corrente de carga tracer2 > tracer1");
698       }
699       else if(einversor > etotalin)
700       {
701         myStatus = String("Energia consumida maior que energia acumulada");
702       }
703       else;
704
705       ThingSpeak.setStatus(myStatus);                          // Grava o status
706
707       // Envia dados para o canal do ThingSpeak
708       int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
709       if(x == 200)
710       {
711         Serial.print("\nCanal atualizado com sucesso");
712       }
713       else
714       {
715         Serial.print("\nProblema fazendo o update do canal. Codigo de erro HTTP " + String(x));
716       }
717
718       //_____________________________________________________________________________________________________________________
```

```

    timeElapsed2 = 0;                                           // Reinicia ciclo
  }
}

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

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());
}

//______________________________________________________________________________________________________________________
```

```
755
756   // HTTP request: Grupo 2
757   void handle_grupo2(void)
758   {
759     Serial.print("\nGrupo 2");
760     Serial2.print("!(2)");
761     server.send(200, "text/html", SendHTML());
762   }
763
764   //_____________________________________________________________________________________________________________
765
766   // HTTP request: Grupo 3
767   void handle_grupo3(void)
768   {
769     Serial.print("\nGrupo 3");
770     Serial2.print("!(3)");
771     server.send(200, "text/html", SendHTML());
772   }
773
774   //_____________________________________________________________________________________________________________
775
776   // HTTP request: Grupo 4
777   void handle_grupo4(void)
778   {
779     Serial.print("\nGrupo 4");
780     Serial2.print("!(4)");
781     server.send(200, "text/html", SendHTML());
782   }
783
784   //_____________________________________________________________________________________________________________
785
786   // HTTP request: Grupo 5
787   void handle_grupo5(void)
788   {
789     Serial.print("\nGrupo 5");
790     Serial2.print("!(5)");
```

```
791     server.send(200, "text/html", SendHTML());
792   }
793
794   //_____________________________________________________________________________________________________________
795
796   // HTTP request: Grupo 6
797   void handle_grupo6(void)
798   {
799     Serial.print("\nGrupo 6");
800     Serial2.print("!(6)");
801     server.send(200, "text/html", SendHTML());
802   }
803
804   //_____________________________________________________________________________________________________________
805
806   // HTTP request: Grupo 7
807   void handle_grupo7(void)
808   {
809     Serial.print("\nGrupo 7");
810     Serial2.print("!(7)");
811     server.send(200, "text/html", SendHTML());
812   }
813
814   //_____________________________________________________________________________________________________________
815
816   // HTTP request: Grupo 8
817   void handle_grupo8(void)
818   {
819     Serial.print("\nGrupo 8");
820     Serial2.print("!(8)");
821     server.send(200, "text/html", SendHTML());
822   }
823
824   //_____________________________________________________________________________________________________________
825
826   // HTTP request: Grupo 9
```

```
827  void handle_grupo9(void)
828  {
829    Serial.print("\nGrupo 9");
830    Serial2.print("!(9)");
831    server.send(200, "text/html", SendHTML());
832  }
833
834  //_____________________________________________________________________________________________________________________
835
836  // HTTP request: Grupo 10
837  void handle_grupo10(void)
838  {
839    Serial.print("\nGrupo 10");
840    Serial2.print("!(10)");
841    server.send(200, "text/html", SendHTML());
842  }
843
844  //_____________________________________________________________________________________________________________________
845
846  // HTTP request: Grupo 11
847  void handle_grupo11(void)
848  {
849    Serial.print("\nGrupo 11");
850    Serial2.print("!(11)");
851    server.send(200, "text/html", SendHTML());
852  }
853
854  //_____________________________________________________________________________________________________________________
855
856  // HTTP request: Grupo 12
857  void handle_grupo12(void)
858  {
859    Serial.print("\nGrupo 12");
860    Serial2.print("!(12)");
861    server.send(200, "text/html", SendHTML());
862  }
```

```
863
864   //_____________________________________________________________________________________________________________
865
866   // HTTP request: other
867   void handle_NotFound(void)
868   {
869     Serial.print("\nPage not found");
870     server.send(404, "text/plain", "Not found");
871   }
872
873   //_____________________________________________________________________________________________________________
874
875   String SendHTML(void)
876   {
877     String html = "<!DOCTYPE html>\n";
878     html += "<html>\n";
879     html += "<head>\n";
880     html += "<title>CoMoSisFog</title>\n";
881     html += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">\n";
882     html += "</head>\n";
883     html += "<body>\n";
884     html += "<div align=\"center\">\n";
885     html += "<h1>CoMoSisFog</h1>\n";
886     html += "<br>\n";
887     html += "<form method=\"GET\">\n";
888     html += "<input type=\"button\" value=\"Grupo 1\" onclick=\"window.location.href='/grupo1'\">\n";
889     html += "<br><br>\n";
890     html += "<input type=\"button\" value=\"Grupo 2\" onclick=\"window.location.href='/grupo2'\">\n";
891     html += "<br><br>\n";
892     html += "<input type=\"button\" value=\"Grupo 3\" onclick=\"window.location.href='/grupo3'\">\n";
893     html += "<br><br>\n";
894     html += "<input type=\"button\" value=\"Grupo 4\" onclick=\"window.location.href='/grupo4'\">\n";
895     html += "<br><br>\n";
896     html += "<input type=\"button\" value=\"Grupo 5\" onclick=\"window.location.href='/grupo5'\">\n";
897     html += "<br><br>\n";
898     html += "<input type=\"button\" value=\"Grupo 6\" onclick=\"window.location.href='/grupo6'\">\n";
```

```
899     html += "<br><br>\n";
900     html += "<input type=\"button\" value=\"Grupo 7\" onclick=\"window.location.href='/grupo7'\">\n";
901     html += "<br><br>\n";
902     html += "<input type=\"button\" value=\"Grupo 8\" onclick=\"window.location.href='/grupo8'\">\n";
903     html += "<br><br>\n";
904     html += "<input type=\"button\" value=\"Grupo 9\" onclick=\"window.location.href='/grupo9'\">\n";
905     html += "<br><br>\n";
906     html += "<input type=\"button\" value=\"Grupo 10\" onclick=\"window.location.href='/grupo10'\">\n";
907     html += "<br><br>\n";
908     html += "<input type=\"button\" value=\"Grupo 11\" onclick=\"window.location.href='/grupo11'\">\n";
909     html += "<br><br>\n";
910     html += "<input type=\"button\" value=\"Grupo 12\" onclick=\"window.location.href='/grupo12'\">\n";
911     html += "</form>\n";
912     html += "</div>\n";
913     html += "</body>\n";
914     html += "</html>\n";
915     return html;
916   }
917
918   //_____________________________________________________________________________________________________________________
919
920   bool recebe_inas(void)
921   {
922     // Formato: ln(valor) onde l=v voltagem,l=c corrente,l=p potencia,l=e energia
923     // Para v,c,p -> n=1 ina1,n=2 ina2,n=3 ina3,valor ponto flutuante
924     // Para e -> n=i etotalin, n=o etotalout
925
926     if (!aguarda_serial(2,1000,1,9)) return false;                // Serial2,1seg,1 caractere,ponto de chamada 9
927     while (Serial2.available() > 0)                               // Aguarda chegar alguma coisa
928     {
929       switch (Serial2.read())
930       {
931         case 'v':
932           switch (Serial2.parseInt())
933           {
934             case 1:
```

```
935                 vina1 = Serial2.parseFloat();
936                 break;
937               case 2:
938                 vina2 = Serial2.parseFloat();
939                 break;
940               case 3:
941                 vina3 = Serial2.parseFloat();
942                 break;
943             }
944             Serial2.read();
945           break;
946           case 'c':
947             switch (Serial2.parseInt())
948             {
949               case 1:
950                 cina1 = Serial2.parseFloat();
951                 break;
952               case 2:
953                 cina2 = Serial2.parseFloat();
954                 break;
955               case 3:
956                 cina3 = Serial2.parseFloat();
957                 break;
958             }
959             Serial2.read();
960           break;
961           case 'p':
962             switch (Serial2.parseInt())
963             {
964               case 1:
965                 pina1 = Serial2.parseFloat();
966                 break;
967               case 2:
968                 pina2 = Serial2.parseFloat();
969                 break;
970               case 3:
```

```
971               pina3 = Serial2.parseFloat();
972               break;
973           }
974           Serial2.read();
975         break;
976         case 'e':
977           switch (Serial2.read())
978           {
979             case 'i':
980               etotalin = Serial2.parseFloat();
981               break;
982             case 'o':
983               etotalout = Serial2.parseFloat();
984               break;
985           }
986         break;
987       }
988     }
989     return true;
990   }
991
992   //_____________________________________________________________________________________________________________
993
994   bool recebe_status(void)
995   {
996     // Recebe os dados dos grupos e refletores
997     str = "";
998     if (!aguarda_serial(2,1000,1,19)) return false;                // Ponto de chamada 19
999     while (Serial2.available() > 0)                                // Aguarda chegar alguma coisa
1000    {
1001      while (Serial2.peek() != '#') str += String(Serial2.read());
1002      Serial1.print(str.c_str());                                  // Envia linha ao PC
1003      Serial2.read();                                              // Descarta #
1004    }
1005    return true;
1006  }
```

```
1007
1008  //_____________________________________________________________________________________________________
1009
1010  bool pede_p1(uint8_t sensor)
1011  {
1012    // Solicita dados a P1
1013    switch (sensor)
1014    {
1015        case 1:                                                 // Pede data e hora a p1
1016          Serial2.print("$(1)");
1017
1018          if (!aguarda_serial(2,1000,14,7))                     // Serial2,1seg,14 caracteres,ponto de chamada 7
1019          {
1020            Serial.print("\nP1 nao enviou data e hora");
1021            return false;
1022          }
1023          dia = Serial2.parseInt();
1024          mes = Serial2.parseInt();
1025          ano = Serial2.parseInt();
1026          horas = Serial2.parseInt();
1027          minutos = Serial2.parseInt();
1028          segundos = Serial2.parseInt();
1029          envia_PC(1);
1030          limpa_buffer_serial(2,10);
1031          return true;
1032        case 2:
1033          Serial2.print("$(2)");                               // Vamos pedir os dados dos INA
1034          if (!recebe_inas())                                   // Recede os dados
1035          {
1036            Serial.print("\nP1 nao enviou dados das INAs");     // P1 não enviou dados
1037            return false;
1038          }
1039          envia_PC(2);
1040          return true;
1041        case 3:
1042          Serial2.print("$(3)");                               // Pede o status dos grupos e refletores
```

```
1043            if (!recebe_status())                                   // Recede os dados
1044            {
1045              Serial.print("\nP1 nao enviou dados de status");      // P1 não enviou dados
1046              return false;
1047            }
1048            return true;
1049          default:
1050            Serial.print("\nCodigo invalido - pede_p1");
1051            return false;
1052      }
1053    }
1054
1055    //_____________________________________________________________________________________________________________________
1056
1057    void envia_PC(unsigned int kk)
1058    {
1059      // Envia dados ao PC
1060      switch (kk)
1061      {
1062        case 1:                                                     // Data e hora
1063          Serial1.print("\n");
1064          Serial1.print(dia,DEC);
1065          Serial1.print('/');
1066          Serial1.print(mes,DEC);
1067          Serial1.print('/');
1068          Serial1.print(ano,DEC);
1069          Serial1.print(' ');
1070          Serial1.print(horas,DEC);
1071          Serial1.print(':');
1072          Serial1.print(minutos,DEC);
1073          Serial1.print(':');
1074          Serial1.print(segundos,DEC);
1075          break;
1076        case 2:                                                     // Dados das inas
1077          Serial1.print("\nvina1 = ");
1078          Serial1.print(vina1,2);
```

```
1079          Serial1.print(" cina1 = ");
1080          Serial1.print(cina1,2);
1081          Serial1.print(" pina1 = ");
1082          Serial1.print(pina1,2);
1083          Serial1.print("\nvina2 = ");
1084          Serial1.print(vina2,2);
1085          Serial1.print(" cina2 = ");
1086          Serial1.print(cina2,2);
1087          Serial1.print(" pina2 = ");
1088          Serial1.print(pina2,2);
1089          Serial1.print("\nvina3 = ");
1090          Serial1.print(vina3,2);
1091          Serial1.print(" cina3 = ");
1092          Serial1.print(cina3,2);
1093          Serial1.print(" pina3 = ");
1094          Serial1.print(pina3,2);
1095          Serial1.print("\netotalin = ");
1096          Serial1.print(etotalin,2);
1097          Serial1.print(" etotalout = ");
1098          Serial1.print(etotalout,2);
1099          break;
1100        default:
1101          Serial.print("\nCodigo invalido - envia_PC");
1102          break;
1103      }
1104    }
1105
1106    //_____________________________________________________________________________________________________________________
1107
1108    void mostra_LCD03(void)
1109    {
1110      // Mostra dados no LCD03
1111      cont1++;
1112      switch (cont1)
1113      {
1114        case 1:
```

```
1115          clear_screen;
1116          display_texto_lc(" vina1 = ",1,1);
1117          display_float(vina1);
1118          display_texto_lc(" cina1 = ",2,1);
1119          display_float(cina1);
1120          display_texto_lc(" pina1 = ",3,1);
1121          display_float(pina1);
1122          break;
1123        case 2:
1124          clear_screen;
1125          display_texto_lc(" vina2 = ",1,1);
1126          display_float(vina2);
1127          display_texto_lc(" cina2 = ",2,1);
1128          display_float(cina2);
1129          display_texto_lc(" pina2 = ",3,1);
1130          display_float(pina2);
1131          break;
1132        case 3:
1133          clear_screen;
1134          display_texto_lc(" vina3 = ",1,1);
1135          display_float(vina3);
1136          display_texto_lc(" cina3 = ",2,1);
1137          display_float(cina3);
1138          display_texto_lc(" pina3 = ",3,1);
1139          display_float(pina3);
1140          break;
1141        case 4:
1142          clear_screen;
1143          display_texto_lc(" etotalin = ",1,1);
1144          display_float(etotalin);
1145          display_texto_lc(" etotalout = ",2,1);
1146          display_float(etotalout);
1147          cont1 = 0;
1148          break;
1149        default:
1150          Serial.println("Contagem invalida");
```

```
1151          break;
1152        }
1153      }
1154
1155      //_____________________________________________________________________________________________________________
1156
1157      void IRAM_ATTR isr1()
1158      {
1159        // Rotina para processamento do interrupt do keypad
1160        pedido_keypad = true;
1161      }
1162
1163      //_____________________________________________________________________________________________________________
1164
1165      void IRAM_ATTR isr2()
1166      {
1167        // Rotina para processamento do interrupt de liga/desliga tudo
1168        liga_desliga = true;
1169      }
1170
1171      //_____________________________________________________________________________________________________________
1172
1173      void limpa_buffer_serial(uint8_t ser,uint8_t cmp)
1174      {
1175        // Limpa o buffer das seriais
1176        uint8_t i = 1;
1177
1178        while (i <= cmp)
1179        {
1180          if (ser == 1) {Serial1.read();i++;}
1181          else if (ser == 2) {Serial2.read();i++;}
1182          else return;
1183        }
1184        delay(1000);
1185      }
1186
```

```
1187  //_____________________________________________________________________________________________________________
1188
1189  bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde)
1190  {
1191    // Aguarda a serial enviar ncar caracteres no tempo intervalo mseg. Indica o ponto do programa que chamou - deonde
1192    timeElapsed = 0;                                          // Inicia contagem de tempo
1193    while(timeElapsed < intervalo)                            // Executa enquanto não passar o intervalo
1194    {
1195      if (ser == 1)                                           // Serial1
1196      {
1197        if (Serial1.available() < ncar);
1198        else return true;
1199      }
1200      else if (ser == 2)                                      // Serial2
1201      {
1202        if (Serial2.available() < ncar);
1203        else return true;
1204      }
1205      else
1206      {
1207        Serial.print("\nSerial");
1208        Serial.print(ser);
1209        Serial.print(" invalida");
1210        Serial.print(" chamada de ");
1211        Serial.print(deonde);
1212        return false;
1213      }
1214    }
1215    Serial.print("\nTime Out esperando Serial");
1216    Serial.print(ser);
1217    Serial.print(" chamada de ");
1218    Serial.print(deonde);
1219    return false;
1220  }
1221
1222  //_____________________________________________________________________________________________________________
```

```
1223
1224  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano)
1225  {
1226    // Retorna o número de dias desde 01/01/2019
1227    unsigned int an,nd = 0;
1228    unsigned int dd[14] = {0,0,31,59,90,120,151,181,212,243,273,304,334,365};
1229
1230    for (an=2019;an<ano;an++)
1231    {
1232      if ((an % 4) == 0) nd += 366;                          // É ano bissexto (vale até 2100)
1233      else nd += 365;
1234    }
1235    if (((an % 4) == 0) && (mes > 2)) nd += (dd[mes] + dia + 1);  // É ano bissexto (vale até 2100)
1236    else nd += (dd[mes] + dia);
1237    return nd;
1238  }
1239
1240  //_____________________________________________________________________________________________________________
1241
1242  void toca_buzzer(unsigned int seg)
1243  {
1244    // Toca o buzzer por seg segundos
1245    unsigned int tp = seg * 1000;
1246    digitalWrite(buzzer, HIGH);                              // Ativa buzzer
1247    delay(tp);
1248    digitalWrite(buzzer, LOW);                               // Desativa buzzer
1249  }
1250
1251  //_____________________________________________________________________________________________________________
1252
1253  void conecta_MQTT(void)
1254  {
1255    // Conecta MQTT
1256    if (!MQTT.connected())
1257    {
1258      long now = millis();
```

```
1259        if (now - lastReconnectAttempt > 5000)
1260        {
1261          lastReconnectAttempt = now;
1262          if (reconnect())                                    // Tentativa de reconexao
1263          {
1264            Serial.println("");
1265            Serial.println("Broker MQTT conectado");
1266            Serial.println("");
1267            lastReconnectAttempt = 0;
1268          }
1269        }
1270      }
1271    }
1272
1273    //_____________________________________________________________________________________________________________
1274
1275    bool reconnect()
1276    {
1277      // Reconecta MQTT
1278      if (MQTT.connect("quiosque",ID_MQTT,PSW_MQTT))
1279      {
1280        MQTT.publish(aviso1,"conectado");                     // Quando reconectado, publica aviso
1281      }
1282      return MQTT.connected();
1283    }
1284
1285    //_____________________________________________________________________________________________________________
1286
1287    void conecta_wifi(unsigned char pr)
1288    {
1289      // Conecta WiFi
1290      WiFi.begin(ssid, password);                             // Inicia WiFi
1291      while (WiFi.status() != WL_CONNECTED)
1292      {
1293        delay(500);
1294        Serial.println("Conectando WiFi..");
```

```
1295      }
1296      Serial.print("Connectado na rede ");
1297      Serial.println(ssid);
1298      if (pr == 1)
1299      {
1300        IPAddress ip = WiFi.localIP();                    // Imprime o IP
1301        Serial.print("IP: ");
1302        Serial.println(ip);
1303        long rssi = WiFi.RSSI();                          // Imprime o nível do sinal
1304        Serial.print("Nivel do sinal (RSSI): ");
1305      }
1306    }
1307
1308    //_____________________________________________________________________________________________________________
1309
1310    bool recebido_P1(unsigned long tp,uint8_t lg)
1311    {
1312      // Aguarda ACK de P1
1313      if (!aguarda_serial(2,tp,1,lg)) return false;       // Serial2,tp seg,1 caractere,ponto de chamada lg - Time out
1314      else if (Serial2.read() != 0x6) return false;       // Não é ACK
1315      else return true;                                   // É ACK
1316    }
1317
1318    //_____________________________________________________________________________________________________________
1319
```