```
/*****************************************************************************************************************************
* 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 V4.0.0 (23/10/2020)
  * 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 fusiveis 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)

  ESP32 remoto - controlador
  * 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  //************************************************************************************************************************* Bibliotecas
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 <PubSubClient.h>                                 // MQTT
68  #include "ThingSpeak.h"                                   // ThingSpeak
69  #include <ArduinoJson.h>                                  // Json
70
71  //_____________________________________________________________________________________________________________ template
72
73  #define i2c_addr_eeprom 0x50                              // Endereco I2C da eeprom At24LC256
74
75  template <class T> int eeWrite(int ee, const T& value)    // Classe genérica para gravação da eeprom
76  {                                                         // ee - posição na eeprom, value - estrutura de dados
77    const byte* p = (const byte*)(const void*)&value;       // Pointer para a estrutura
78    int i;                                                  // Contador
79    Wire.beginTransmission(i2c_addr_eeprom);                // Inicia transmissão I2C
80    Wire.write((int)(ee >> 8)); // MSB                      // Envia MSB da posição
81    Wire.write((int)(ee & 0xFF)); // LSB                    // Envia LSB da posição
82    for (i = 0; i < sizeof(value); i++)                     // Para todos os bytes da estrutura
83    Wire.write(*p++);                                       // Envia o byte
84    Wire.endTransmission();                                 // Encerra transmissão I2C
85    return i;                                               // Retorna o número de bytes gravados
86  }
87
88  template <class T> int eeRead(int ee, T& value)           // Classe genérica para leitura da eeprom
89  {                                                         // ee - posição na eeprom, value - estrutura de dados
90    byte* p = (byte*)(void*)&value;                         // Pointer para a estrutura
91    int i;                                                  // Contador
92    Wire.beginTransmission(i2c_addr_eeprom);                // Inicia transmissão I2C
```

```
93    Wire.write((int)(ee >> 8)); // MSB                          // Envia MSB da posição
94    Wire.write((int)(ee & 0xFF)); // LSB                        // Envia LSB da posição
95    Wire.endTransmission();                                     // Encerra transmissão I2C
96    Wire.requestFrom(i2c_addr_eeprom,sizeof(value));            // Solicita à eeprom o número de bytes da estrutura
97    for (i = 0; i < sizeof(value); i++)                         // Para esse número de bytes
98    if(Wire.available())                                        // Se tem dados no buffer
99    *p++ = Wire.read();                                         // Lê o byte para a estrutura
100   return i;                                                   // Retorna o número de bytes lidos
101 }
102
103 struct eeprom                                                 // Estrutura de dados na eeprom
104 {
105   unsigned long valido;                                       // Código para validar dados
106   unsigned long inter1;                                       // Intervalo de tempo para "elapsedMillis timeElapsed1;"
107   unsigned long inter2;                                       // Intervalo de tempo para "elapsedMillis timeElapsed2;"
108   unsigned int cod_mon;                                       // Código para imprimir ou não no monitor serial
109 } eeprom;
110
111 //_____________________________________________________________________________________________________________ WiFi
112
113 // WiFi
114 const char* ssid = ssid_ego;                                  // Nome da rede WiFi
115 const char* password = password_ego;                          // Senha da rede WiFi
116
117 //_____________________________________________________________________________________________________________ MQTT
118
119 // MQTT
120 const char* BROKER_MQTT = BROKER_MQTT_ego;                    // URL do broker MQTT
121 int BROKER_PORT = BROKER_PORT_ego;                            // Porta do Broker MQTT
122 #define ID_MQTT  ID_MQTT_ego                                  // User ID
123 #define PSW_MQTT PSW_MQTT_ego                                 // 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 //_____________________________________________________________________________________________ Variaveis e constantes
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  char temp[30];                                            // Temporario
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  bool tudo_ligado;                                         // Variável de status
155  long lastReconnectAttempt = 0;                            //  Tempo última conexão MQTT
156  String versao = "V4.0.0";                                 // ************* Versão ******************************
157
158  unsigned int dia;
159  unsigned int mes;
160  unsigned int ano;
161  unsigned int horas;
162  unsigned int minutos;
163  unsigned int segundos;
164
165  unsigned int d1;
166  unsigned int m1;
167  unsigned int a1;
168  unsigned int d2;
169  unsigned int m2;
170  unsigned int a2;
171
172  float vina1,cina1,pina1,vina2,cina2,pina2,vina3,cina3,pina3,etracer1,etracer2,etotalin,einversor,etotalout,volt;
173
174  String tempo,str;
175
176  unsigned int k,tmp;
177  uint8_t cont1;
178  char sp;
179  bool ch;
180
181  //_____________________________________________________________________________________________________________ IPAddress
182
183  IPAddress local_ip(192, 168, 5, 1);                       // IP do servidor AP
184  IPAddress gateway(192, 168, 5, 1);                        // Gateway do servidor AP
```

```
185  IPAddress subnet(255, 255, 255, 0);                         // Subnet do servidor AP
186
187  //_____________________________________________________________________________________________________ Instâncias
188
189  // Instancias
190  WiFiClient wificlient;                                      // Instancia WiFi MQTT
191  WiFiClient wifiTS;                                          // Instancia WiFi ThingSpeak
192  PubSubClient MQTT(BROKER_MQTT,BROKER_PORT,wificlient);      // Instancia MQTT passando o objeto wificlient
193  DNSServer dnsServer;                                        // Instancia DNSServer para nome de dominio
194  WebServer server(80);                                       // Instancia WebServer para serevidor AP
195  elapsedMillis timeElapsed1;                                 // Instancia elapsedMillis para eeprom.inter1
196  elapsedMillis timeElapsed2;                                 // Instancia elapsedMillis para eeprom.inter2
197  elapsedMillis timeElapsed;                                  // Instancia elapsedMillis generico
198  Nanoshield_ADC adc(0x4B);                                   // Instancia Nanoshield_ADC
199
200  //_____________________________________________________________________________________________________ ThingSpeak
201
202  // ThingSpeak
203  unsigned long myChannelNumber = myChannelNumber_ego;        // ThingSpeak ID do canal
204  const char * myWriteAPIKey = myWriteAPIKey_ego;             // ThingSpeak chave de escrita
205  String myStatus = "";                                       // ThingSpeak para teste (11)
206
207  //_____________________________________________________________________________________________________ Funções
208
209  // Funções
210  void conecta_MQTT(void);                                    // Conecta MQTT
211  bool reconnect();                                           // Reconecta MQTT
212  void conecta_wifi(unsigned char pr);                        // Conecta WiFi
213  bool recebe_status(void);                                   // Recebe os dados dos grupos e refletores
214  bool recebe_inas(void);                                     // Recebe dados das INAs
215  bool pede_p1(uint8_t sensor);                               // Pede dados à Parte 1
216  void envia_PC(unsigned int kk);                             // Envia dados ao PC
217  void mostra_LCD03(void);                                    // Mostra dados no LCD03
218  void limpa_buffer_serial(uint8_t ser,uint8_t cmp);          // Limpa buffer das seriais
219  bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde); // Aguarda caracteres nas seriais
220  bool recebido_P1(unsigned long tp);                         // ACK de P1
221  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano); // Calcula número de dias
222  void IRAM_ATTR isr1();                                      // Rotina de interrupt para botão de pedido do keypad
223  void IRAM_ATTR isr2();                                      // Rotina de interrupt para botão de toggle tudo
224  void toca_buzzer(unsigned int seg);                         // Rotina para tocar o buzzer seg segundos
225  String SendHTML(void) ;                                     // Para o servidor AP
226  void handle_NotFound(void);                                 // Para o servidor AP
227  void handle_grupo0(void);                                   // Para o servidor AP
228  void handle_grupo1(void);                                   // Para o servidor AP
229  void handle_grupo2(void);                                   // Para o servidor AP
230  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 ____________________________________________________ Setup

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

  LCD03_begin(0xC6,30);                                         // Inicia o LCD03 - tespera = 30 segundos
  delay(3000);

  str = "\nCoMoSisFog " + versao +  " parte 2";
  Serial.print(str);                                            // Identificação
  Serial.print("\nI2C ESP32 - SDA 21 SCL 22 inicializado");
  Serial.print("\nNanoshield_ADC inicializado");
  Serial.println("\nLCD03 inicializado");

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

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

```
277
278     volt = adc.readVoltage(0);                                  // Lê a voltagem no ADC
279     volt *= 5.87359;                                            // Escala em função do divisor de tensão
280     str = " Bat " + String(volt,2) + "V";                       // Monta String para display no LCD03
281     display_string_lc(&str,4,1);                                // Faz o display
282     if (volt <= 9.6)                                            // Verifica se tem que carregar a bateria
283     {
284       display_texto("  *RECARGA*");                             // Aviso para recarregar baterias - display
285       toca_buzzer(5);                                           // Aviso para recarregar baterias - buzzer
286     }
287
288     toca_buzzer(5);
289
290     conecta_wifi(1);                                            // Conecta WiFi
291
292     ThingSpeak.begin(wifiTS);                                   // Inicia ThingSpeak
293
294     Serial.print("\nThingSpeak inicializado");
295
296     WiFi.softAP(ssidsoftAP, passwordsoftAP, 13, false, 1);      // Parametros de acesso ao servidor AP
297     delay(100);
298     WiFi.softAPConfig(local_ip, gateway, subnet);               // Parametros de acesso ao servidor AP
299
300     dnsServer.setTTL(300);                                      // Servidor DNS
301     dnsServer.setErrorReplyCode(DNSReplyCode::ServerFailure);   // Servidor DNS
302     dnsServer.start(DNS_PORT, "comosisfog.net", local_ip);      // Servidor DNS
303
304     Serial2.begin(9600, SERIAL_8N1, RX1pin, TX1pin);            // Serial1 para Xbee - ESP32 - RX 16 TX 17
305     Serial2.setTimeout(10000);                                  // Time out
306     Serial.print("\nSerial para Xbee inicializada");
307
308     Serial1.begin(9600, SERIAL_8N1, RX2pin, TX2pin);           // Serial2 para PC - ESP32 - RX 4 TX 2
309     Serial1.setTimeout(10000);                                  // Time out
310     Serial.print("\nSerial para PC inicializada");
311
312     // Atribuição dos handles para o servidor AP
313
314     server.on("/", HTTP_GET, handle_OnConnect);
315     server.on("/grupo0", HTTP_GET, handle_grupo0);
316     server.on("/grupo1", HTTP_GET, handle_grupo1);
317     server.on("/grupo2", HTTP_GET, handle_grupo2);
318     server.on("/grupo3", HTTP_GET, handle_grupo3);
319     server.on("/grupo4", HTTP_GET, handle_grupo4);
320     server.on("/grupo5", HTTP_GET, handle_grupo5);
321     server.on("/grupo6", HTTP_GET, handle_grupo6);
322     server.on("/grupo7", HTTP_GET, handle_grupo7);
```

```
323     server.on("/grupo8", HTTP_GET, handle_grupo8);
324     server.on("/grupo9", HTTP_GET, handle_grupo9);
325     server.on("/grupo10", HTTP_GET, handle_grupo10);
326     server.on("/grupo11", HTTP_GET, handle_grupo11);
327     server.on("/grupo12", HTTP_GET, handle_grupo12);
328     server.onNotFound(handle_NotFound);
329
330     server.begin();                                          // Inicializa servidor AP
331     Serial.print("\nweb server e DNS inicializados");
332
333   //_____________________________________________________________________________________________________________ Força eeprom
334
335   // Força eeprom invalida para inicializar default
336   // Depois lembrar de retirar este codigo
337     //eeprom.valido = 0;
338     //eeprom.inter1 = 0;
339     //eeprom.inter2 = 0;
340     //eeprom.cod_mon = 0;
341     //eeWrite(0,eeprom);
342     //delay (30);
343   //_____________________________________________________________________________________________________________
344
345     eeRead(0,eeprom);                                        // Le eeprom
346     delay (30);
347
348     if (eeprom.valido != key)                                // Eeprom invalida, força reinicializacao default
349     {
350       eeprom.inter1 = 60000;                                 // Default - 1 minuto
351       eeprom.inter2 = 300000;                                // Default - 5 minutos
352       eeprom.cod_mon = 1;                                    // Habilita monitor serial
353       eeprom.valido = key;                                   // Valida eeprom
354
355       eeWrite(0,eeprom);                                     // Grava eeprom
356       delay (30);
357       Serial.print("\neeprom default");
358     }
359
360     pinMode(keypadpin, INPUT_PULLUP);                        // Pino a ser acionado quando for teclar no keypad
361     attachInterrupt(digitalPinToInterrupt(keypadpin),isr1,FALLING); // Define interrupt no pino keypadpin
362     pinMode(liga_desliga_tudo, INPUT_PULLUP);                // Pino a ser acionado quando for toggle tudo
363     attachInterrupt(digitalPinToInterrupt(liga_desliga_tudo),isr2,FALLING); // Define interrupt no pino liga_desliga_tudo
364
365     timeElapsed1 = 0;
366     timeElapsed2 = 0;
367     timeElapsed = 0;
368     cont1 = 0;
```

```
  pedido_keypad = false;
  liga_desliga = false;
  tudo_ligado = false;
}

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

void loop()
{
  int j;

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

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

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

  str = " Bat " + String(volt,2) + "V";                         // Monta String para display no LCD03
  display_string_lc(&str,4,1);                                  // Faz o display
  if (volt <= 10.5)                                             // Verifica se tem que carregar a bateria
  {
    display_texto("  *RECARGA*");                               // Aviso para recarregar baterias - display
    toca_buzzer(5);                                             // Aviso para recarregar baterias - buzzer
  }

  //____________________________________________________________________________________________________________________ liga/desliga tudo

  // Verifica se foi apertado o botão liga/desliga tudo
  if (liga_desliga)                                               // Toggle tudo
  {
    if (!tudo_ligado)
    {
      Serial2.print('>');                                         // Envia a p1 liga tudo
      Serial.print("\nEnviou pedido de ligar tudo a P1");
      tudo_ligado = true;
    }
    else
    {
      Serial2.print('<');                                         // Envia a p1 desliga tudo
```

```
415         Serial.print("\nEnviou pedido de desligar tudo a P1");
416         tudo_ligado = false;
417       }
418       liga_desliga = false;
419     }
420
421     //_____________________________________________________________________________________________ enviou texto
422
423     if (Serial2.available() > 0)                          // *** Atenção - verificar 0 é OK ***
424     {
425       // Verifica se P1 enviou algum texto <....>
426       if (Serial2.read() == '<')                          // Verifica se a P1 enviou mensagem - formato - <texto>
427       {
428         str = " " + Serial2.readStringUntil('>');
429         clear_line(1);
430         clear_line(2);
431         clear_line(3);
432         display_string_lc(&str,1,1);                      // Display no LCD03
433         toca_buzzer(5);                                   // Toca buzzer
434         while (read_keypad() != '#')                      // Espera para ler. Repete até digitar #
435         {
436           delay(3000);                                    // Aguarda
437           toca_buzzer(5);                                 // Toca buzzer
438         }
439       }
440     }
441
442     //_____________________________________________________________________________________________ comandos
443
444     if (Serial1.available() >= 4)                         // Pc enviou algo?*** Atenção - verificar se 4 é OK ***
445     {
446       // Verifica se o PC enviou alguma coisa: comando, coniguração p1, coniguração p2, configuração dos grupos de
447       // refletores, configuração de seus modos ou solicitação de dados. Ou a Parte 1 enviou mensagem.
448       // Indicadores válidos:
449       // '!' - Comando toggle dos grupos de refletores - formato - !(g) - onde g é o número do grupo.  g = 0 => tudo
450       // '$' - Solicitação de dados - formato - $(i) - onde i=1 data-hora, i=2 dados INAs e energias, i=3 status dos
451       //       grupos e refletores
452       // '@' - Solicitação de dados - formato - @(di,df) - onde di - data de início, df - data de fim, ambos no formato
453       //       dia,mes,ano. Ex: @(1,8,2019/31,8,2019) vai ler o SD e enviar os dados relativos a esse período
454       // '|' - Configuração de ciclos em p2 - formato - |n(v) - onde n=1 eeprom.inter1, n=2 eeprom.inter2,
455       //       n=3 eepro.cod_mon, v = 0 ou 1 ou valor em minutos
456       // '#' - Configuração de ciclos em p1 - formato - #n(v) - onde n=1 inter1, n=2 inter2,
457       //       n=3 cod_mon, v = 0 ou 1 ou valor em minutos
458       // '&' - Configuração dos grupos de refletores - formato - &n(r,...,r) - onde n é o número do grupo
459       //       e r o número do refletor
460       // '=' - Configuração dos horarios liga/desliga dos grupos e modo de operação de cada grupo
```

```
461       //       Modo de operação: ativo -> sim(1)/não(0), automatico -> fotocelula, programado -> hora liga/desliga,
462       //       manual -> comando de toggle via p2,PC ou celular - formato - =n(a,b,c,d) - onde n é o número do grupo
463       //       a=0 -> inativo, a=1 => ativo, b=1 -> automático(fotocelula), b=2 -> manual(comando PC,p2,celular),
464       //       b=3 -> programado (c e d só existem neste caso), c -> hora liga (formato hh:mm hh -> hora formato 24,
465       //       mm -> minutos), d -> hora desliga (idem hora liga)
466
467       sp = Serial1.peek();
468
469       switch (sp)
470       {
471         case '|':
472           ch = true;
473           if (!aguarda_serial(1,10000,5,1)) break;               // Serial1,10seg,5 caracteres,ponto de chamada 1
474           switch(Serial1.parseInt())                             // Configura ciclos em p2
475           {
476             case 1:                                              // Interval1_p2 em minutos
477               tmp = Serial1.parseInt();
478               eeprom.inter1 = tmp * 60000;                       // Passa para milisegundos
479               Serial1.print("\n§1(");
480               Serial1.print(tmp);
481               Serial1.print(")");
482               break;
483             case 2:                                              // Interval2_p2 em minutos
484               tmp = Serial1.parseInt();
485               eeprom.inter2 = tmp * 60000;                       // Passa para milisegundos
486               Serial1.print("\n§2(");
487               Serial1.print(tmp);
488               Serial1.print(")");
489               break;
490             case 3:                                              // eeprom.cod_mon
491               eeprom.cod_mon = Serial1.parseInt();
492               Serial1.print("\n§3(");
493               Serial1.print(eeprom.cod_mon);
494               Serial1.print(")");
495               break;
496             default:
497               Serial.println("\nComando § - parametro invalido ");
498               ch = false;
499               break;
500           }
501           if (ch)
502           {
503             eeWrite(0,eeprom);                                   // Grava eeprom
504             delay (30);
505             Serial.print("\neeprom gravada P2");
506             MQTT.publish(aviso2,"eeprom P2 configurada");        // Avisa no MQTT
```

```
507           }
508           break;
509         case '$':
510           if (!aguarda_serial(1,10000,4,2)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 2
511           k = Serial1.parseInt();
512           if (!pede_p1(k))
513           {
514             Serial.print("\nSem resposta de P1 - ponto de chamada 14"); // ponto de chamada 14
515           }
516           break;
517         case '!':
518           if (!aguarda_serial(1,10000,4,3)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 3
519           str = Serial1.readStringUntil(')') + ')';               // Lê comando
520           Serial2.print(str);                                     // Envia comando para P1
521           if (!recebido_P1(10000,10))                             // Serial2,10seg,1 caractere,ponto de chamada 10
522           {
523             Serial.print("\nSem resposta de P1 - ponto de chamada 15"); // ponto de chamada 15
524             break;
525           }
526           Serial.print("\nEnviou para P1: "); Serial.print(str);
527           break;
528         case '#':
529           if (!aguarda_serial(1,10000,5,4)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 4
530           str = Serial1.readStringUntil(')') + ')';               // Lê comando
531           Serial2.print(str);                                     // Envia comando para P1
532           if (!recebido_P1(10000,11))                             // Serial2,10seg,1 caractere,ponto de chamada 11
533           {
534             Serial.print("\nSem resposta de P1 - ponto de chamada 16"); // ponto de chamada 16
535             break;
536           }
537           Serial.print("\nEnviou para P1: "); Serial.print(str);
538           MQTT.publish(aviso1,"eeprom P1 configurada");           // Avisa no MQTT
539           break;
540         case '&':
541           if (!aguarda_serial(1,10000,5,5)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 5
542           str = Serial1.readStringUntil(')') + ')';               // Lê comando
543           Serial2.print(str);                                     // Envia comando para P1
544           if (!recebido_P1(10000,12))                             // Serial2,10seg,1 caractere,ponto de chamada 12
545           {
546             Serial.print("\nSem resposta de P1 - ponto de chamada 17"); // ponto de chamada 17
547             break;
548           }
549           Serial.print("\nEnviou para P1: "); Serial.print(str);
550           break;
551         case '=':
552           if (!aguarda_serial(1,10000,6,6)) break;                // Serial1,10seg,6 caracteres,ponto de chamada 6
```

```
553           str = Serial1.readStringUntil(')') + ')';              // Lê comando
554           Serial2.print(str);                                    // Envia comando para P1
555           if (!recebido_P1(10000,13))                            // Serial2,10seg,1 caractere,ponto de chamada 13
556           {
557             Serial.print("\nSem resposta de P1 - ponto de chamada 18"); // ponto de chamada 18
558             break;
559           }
560           Serial.print("\nEnviou para P1: "); Serial.print(str);
561           break;
562         case '@':
563           d1 = Serial1.parseInt();
564           m1 = Serial1.parseInt();
565           a1 = Serial1.parseInt();
566           d2 = Serial1.parseInt();
567           m2 = Serial1.parseInt();
568           a2 = Serial1.parseInt();
569
570           str = String(dias_desde_01_01_2019(d1,m1,a1)) + ',' + String(dias_desde_01_01_2019(d2,m2,a2)); // Comando
571           Serial2.print(str);                                    // Envia comando para P1
572           Serial.print("\nEnviou para P1: "); Serial.print(str);
573           // Vamos aguardar a Parte 1 enviar o que foi pedido e mandar para o PC
574           if (!aguarda_serial(2,10000,40,8)) break;              // Serial2,10seg,40 caracteres,ponto de chamada 8
575           str = Serial2.readStringUntil('>');                    // Recebe de P1
576           Serial1.print("\n"+str);                               // Manda para o PC
577           Serial2.read();                                        // Acerta ponto de leitura
578           break;
579         default:
580           Serial.print("\nCodigo de comando ");
581           Serial.print(sp);
582           Serial.print(" invalido");
583           break;
584       }
585       limpa_buffer_serial(1,40);                                 // Limpa buffer
586     }
587
588     //_____________________________________________________________________________________________________ comandos keypad
589
590     // Verifica se veio solicitação do keypad para teclar comando de ligar/desligar refletores (códigos 1 a 12)
591     // Podemos utilizar os códigos > 12 para comandos de outra natureza.
592     if (pedido_keypad)                                           // Verifica se tem interrupt de teclado
593     {                                                            // Sim
594       clear_line(1);
595       clear_line(2);
596       clear_line(3);
597       display_texto_lc(" Digite o comando",1,1);
598       temp[0] = '%';
```

```
    temp[1] = '(';
    j = 1;

    if(entrada_teclado(entrada))                                // Recebe digitação. A leitura é encerrada
    {                                                           // quando é digitado #
      // Formato: r*r*r* ... r# onde r - número do refletor, * - divisor obrigatório, # - encerra comando
      Serial.print("\nDigitado: ");
      Serial.print(entrada);
      Serial.print('#');
      while (entrada[j] != '\0')
      {
        if (entrada[j] == '*') temp[j+1] = ',';                 // É separador de parâmetros
        else temp[j+1] = entrada[j];                            // É digito
        j++;
      }
      clear_line(1);
      clear_line(2);
      temp[j+1] = ')'; temp[j+2] = '\0';
      Serial2.print(temp);                                      // Envia a p1 o comando
      Serial.print("\nComando enviado P1: ");
      Serial.print(temp);                                       // Imprime comando
      String trs = ' ' + String(temp);
      display_texto_lc(" Enviou pedido a P1",1,1);
      display_string_lc(&trs,2,1);                              // Display do que recebeu e enviou P1
    }
    else                                                        // Deu erro em entrada_teclado
    {
      Serial.print("\nAlgo nao funcionou - comando keypad");
      display_texto_lc(" Algo nao funcionou",2,1);
    }
    pedido_keypad = false;
  }

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

  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 _______________________________________________________ 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
  }

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

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

    //____________________________________________ Publicação MQTT _____________________________________________________ MQTT

    // Documentos para Json
    StaticJsonDocument<100> doc1;
    StaticJsonDocument<100> doc2;
    StaticJsonDocument<100> doc3;
    StaticJsonDocument<100> doc4;
    StaticJsonDocument<100> doc5;

    doc1["data"] = tempo;
    JsonArray ina1 = doc1.createNestedArray("tracer1");
    ina1.add(String(vina1, 2));
    ina1.add(String(cina1, 2));
```

```
691       ina1.add(String(etracer1, 2));
692
693       serializeJson(doc1, buffer);
694
695       Serial.println(" ");
696       Serial.println(buffer);
697
698       MQTT.publish(tracer1, buffer);                              // Publica MQTT tracer1
699
700       doc2["data"] = tempo;
701       JsonArray ina2 = doc2.createNestedArray("tracer2");
702       ina2.add(String(vina2, 2));
703       ina2.add(String(cina2, 2));
704       ina2.add(String(etracer2, 2));
705
706       serializeJson(doc2, buffer);
707
708       Serial.println(" ");
709       Serial.println(buffer);
710
711       MQTT.publish(tracer2, buffer);                              // Publica MQTT tracer2
712
713       doc3["data"] = tempo;
714       JsonArray ina3 = doc3.createNestedArray("inversor");
715       ina3.add(String(vina3, 2));
716       ina3.add(String(cina3, 2));
717       ina3.add(String(einversor, 2));
718
719       serializeJson(doc3, buffer);
720
721       Serial.println(" ");
722       Serial.println(buffer);
723
724       MQTT.publish(inversor, buffer);                             // Publica MQTT inversor
725
726       doc4["data"] = tempo;
727       doc4["energia_total_in"] = String(etotalin, 2);
728       serializeJson(doc4, buffer);
729
730       Serial.println(" ");
731       Serial.println(buffer);
732
733       MQTT.publish(etotin, buffer);                               // Publica MQTT energia acumulada
734
735       doc5["data"] = tempo;
736       doc5["energia_total_out"] = String(einversor, 2);
```

```
737       serializeJson(doc5, buffer);
738
739       Serial.println(" ");
740       Serial.println(buffer);
741
742       MQTT.publish(etotout, buffer);                              // Publica MQTT energia consumida
743
744       //_____________________________________________ Publicação ThingSpeak _____________________________________________ ThingSpeak
745
746       // Define os valores para os campos do canal ThingSpeak
747       ThingSpeak.setField(1, vina1);
748       ThingSpeak.setField(2, cina1);
749       ThingSpeak.setField(3, vina2);
750       ThingSpeak.setField(4, cina2);
751       ThingSpeak.setField(5, vina3);
752       ThingSpeak.setField(6, cina3);
753       ThingSpeak.setField(7, etotalin);
754       ThingSpeak.setField(8, einversor);
755
756       // Define o status para o canal do ThingSpeak
757       if(cina1 > cina2)
758       {
759         myStatus = String("Corrente de carga tracer1 > tracer2");
760       }
761       else if(cina1 < cina2)
762       {
763         myStatus = String("Corrente de carga tracer2 > tracer1");
764       }
765       else if(einversor > etotalin)
766       {
767         myStatus = String("Energia consumida maior que energia acumulada");
768       }
769       else;
770
771       ThingSpeak.setStatus(myStatus);                             // Grava o status
772
773       // Envia dados para o canal do ThingSpeak
774       int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
775       if(x == 200)
776       {
777         Serial.print("\nCanal atualizado com sucesso");
778       }
779       else
780       {
781         Serial.print("\nProblema fazendo o update do canal. Codigo de erro HTTP " + String(x));
782       }
```

```

    //__________________________________________________________________________________________________________________ ADC

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

    str = " Bat " + String(volt,2) + "V";                           // Monta String para display no LCD03
    display_string_lc(&str,4,1);                                    // Faz o display
    if (volt <= 10.5)                                               // Verifica se tem que carregar a bateria
    {
      display_texto("  *RECARGA*");                                 // Aviso para recarregar baterias - display
      toca_buzzer(5);                                               // Aviso para recarregar baterias - buzzer
    }

    //__________________________________________________________________________________________________________________

    timeElapsed2 = 0;                                               // Reinicia ciclo
  }
}

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//______________________________________________________________________________________________________________________ handle_OnConnect

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

//______________________________________________________________________________________________________________________ handle_grupo0

// 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());
}

//______________________________________________________________________________________________________________________ handle_grupo1

// HTTP request: Grupo 1
void handle_grupo1(void)
{
```

```
829     Serial.print("\nGrupo 1");
830     Serial2.print("!(1)");
831     server.send(200, "text/html", SendHTML());
832   }
833
834   //_____________________________________________________________________________________________ handle_grupo2
835
836   // HTTP request: Grupo 2
837   void handle_grupo2(void)
838   {
839     Serial.print("\nGrupo 2");
840     Serial2.print("!(2)");
841     server.send(200, "text/html", SendHTML());
842   }
843
844   //_____________________________________________________________________________________________ handle_grupo3
845
846   // HTTP request: Grupo 3
847   void handle_grupo3(void)
848   {
849     Serial.print("\nGrupo 3");
850     Serial2.print("!(3)");
851     server.send(200, "text/html", SendHTML());
852   }
853
854   //_____________________________________________________________________________________________ handle_grupo4
855
856   // HTTP request: Grupo 4
857   void handle_grupo4(void)
858   {
859     Serial.print("\nGrupo 4");
860     Serial2.print("!(4)");
861     server.send(200, "text/html", SendHTML());
862   }
863
864   //_____________________________________________________________________________________________ handle_grupo5
865
866   // HTTP request: Grupo 5
867   void handle_grupo5(void)
868   {
869     Serial.print("\nGrupo 5");
870     Serial2.print("!(5)");
871     server.send(200, "text/html", SendHTML());
872   }
873
874   //_____________________________________________________________________________________________ handle_grupo6
```

```
875
876  // HTTP request: Grupo 6
877  void handle_grupo6(void)
878  {
879    Serial.print("\nGrupo 6");
880    Serial2.print("!(6)");
881    server.send(200, "text/html", SendHTML());
882  }
883
884  //_____________________________________________________________________________________________ handle_grupo7
885
886  // HTTP request: Grupo 7
887  void handle_grupo7(void)
888  {
889    Serial.print("\nGrupo 7");
890    Serial2.print("!(7)");
891    server.send(200, "text/html", SendHTML());
892  }
893
894  //_____________________________________________________________________________________________ handle_grupo8
895
896  // HTTP request: Grupo 8
897  void handle_grupo8(void)
898  {
899    Serial.print("\nGrupo 8");
900    Serial2.print("!(8)");
901    server.send(200, "text/html", SendHTML());
902  }
903
904  //_____________________________________________________________________________________________ handle_grupo9
905
906  // HTTP request: Grupo 9
907  void handle_grupo9(void)
908  {
909    Serial.print("\nGrupo 9");
910    Serial2.print("!(9)");
911    server.send(200, "text/html", SendHTML());
912  }
913
914  //_____________________________________________________________________________________________ handle_grupo10
915
916  // HTTP request: Grupo 10
917  void handle_grupo10(void)
918  {
919    Serial.print("\nGrupo 10");
920    Serial2.print("!(10)");
```

```
921   server.send(200, "text/html", SendHTML());
922 }
923
924 //_____________________________________________________________________________ handle_grupo11
925
926 // HTTP request: Grupo 11
927 void handle_grupo11(void)
928 {
929   Serial.print("\nGrupo 11");
930   Serial2.print("!(11)");
931   server.send(200, "text/html", SendHTML());
932 }
933
934 //_____________________________________________________________________________ handle_grupo12
935
936 // HTTP request: Grupo 12
937 void handle_grupo12(void)
938 {
939   Serial.print("\nGrupo 12");
940   Serial2.print("!(12)");
941   server.send(200, "text/html", SendHTML());
942 }
943
944 //_____________________________________________________________________________ handle_NotFound
945
946 // HTTP request: other
947 void handle_NotFound(void)
948 {
949   Serial.print("\nPage not found");
950   server.send(404, "text/plain", "Not found");
951 }
952
953 //_____________________________________________________________________________ SendHTML
954
955 String SendHTML(void)
956 {
957   String html = "<!DOCTYPE html>\n";
958   html += "<html>\n";
959   html += "<head>\n";
960   html += "<title>CoMoSisFog</title>\n";
961   html += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">\n";
962   html += "</head>\n";
963   html += "<body>\n";
964   html += "<div align=\"center\">\n";
965   html += "<h1>CoMoSisFog</h1>\n";
966   html += "<br>\n";
```

```
967     html += "<form method=\"GET\">\n";
968     html += "<input type=\"button\" value=\"Grupo 1\" onclick=\"window.location.href='/grupo1'\">\n";
969     html += "<br><br>\n";
970     html += "<input type=\"button\" value=\"Grupo 2\" onclick=\"window.location.href='/grupo2'\">\n";
971     html += "<br><br>\n";
972     html += "<input type=\"button\" value=\"Grupo 3\" onclick=\"window.location.href='/grupo3'\">\n";
973     html += "<br><br>\n";
974     html += "<input type=\"button\" value=\"Grupo 4\" onclick=\"window.location.href='/grupo4'\">\n";
975     html += "<br><br>\n";
976     html += "<input type=\"button\" value=\"Grupo 5\" onclick=\"window.location.href='/grupo5'\">\n";
977     html += "<br><br>\n";
978     html += "<input type=\"button\" value=\"Grupo 6\" onclick=\"window.location.href='/grupo6'\">\n";
979     html += "<br><br>\n";
980     html += "<input type=\"button\" value=\"Grupo 7\" onclick=\"window.location.href='/grupo7'\">\n";
981     html += "<br><br>\n";
982     html += "<input type=\"button\" value=\"Grupo 8\" onclick=\"window.location.href='/grupo8'\">\n";
983     html += "<br><br>\n";
984     html += "<input type=\"button\" value=\"Grupo 9\" onclick=\"window.location.href='/grupo9'\">\n";
985     html += "<br><br>\n";
986     html += "<input type=\"button\" value=\"Grupo 10\" onclick=\"window.location.href='/grupo10'\">\n";
987     html += "<br><br>\n";
988     html += "<input type=\"button\" value=\"Grupo 11\" onclick=\"window.location.href='/grupo11'\">\n";
989     html += "<br><br>\n";
990     html += "<input type=\"button\" value=\"Grupo 12\" onclick=\"window.location.href='/grupo12'\">\n";
991     html += "</form>\n";
992     html += "</div>\n";
993     html += "</body>\n";
994     html += "</html>\n";
995     return html;
996   }
997
998   //_____________________________________________________________________________________________________ recebe_inas
999
1000  bool recebe_inas(void)
1001  {
1002    // Formato: ln(valor) onde l=v voltagem,l=c corrente,l=p potencia,l=e energia
1003    // Para v,c,p -> n=1 ina1,n=2 ina2,n=3 ina3,valor ponto flutuante
1004    // Para e -> n=i etotalin, n=o etotalout
1005
1006    if (!aguarda_serial(2,1000,1,9)) return false;              // Serial2,1seg,1 caractere,ponto de chamada 9
1007    while (Serial2.available() > 0)                             // Aguarda chegar alguma coisa
1008    {
1009      switch (Serial2.read())
1010      {
1011        case 'v':
1012          switch (Serial2.parseInt())
```

```
1013          {
1014            case 1:
1015              vina1 = Serial2.parseFloat();
1016              break;
1017            case 2:
1018              vina2 = Serial2.parseFloat();
1019              break;
1020            case 3:
1021              vina3 = Serial2.parseFloat();
1022              break;
1023          }
1024          Serial2.read();
1025        break;
1026        case 'c':
1027          switch (Serial2.parseInt())
1028          {
1029            case 1:
1030              cina1 = Serial2.parseFloat();
1031              break;
1032            case 2:
1033              cina2 = Serial2.parseFloat();
1034              break;
1035            case 3:
1036              cina3 = Serial2.parseFloat();
1037              break;
1038          }
1039          Serial2.read();
1040        break;
1041        case 'p':
1042          switch (Serial2.parseInt())
1043          {
1044            case 1:
1045              pina1 = Serial2.parseFloat();
1046              break;
1047            case 2:
1048              pina2 = Serial2.parseFloat();
1049              break;
1050            case 3:
1051              pina3 = Serial2.parseFloat();
1052              break;
1053          }
1054          Serial2.read();
1055        break;
1056        case 'e':
1057          switch (Serial2.read())
1058          {
```

```
1059            case 'i':
1060              etotalin = Serial2.parseFloat();
1061              break;
1062            case 'o':
1063              etotalout = Serial2.parseFloat();
1064              break;
1065          }
1066        break;
1067      }
1068    }
1069    return true;
1070  }
1071
1072  //_____________________________________________________________________________________________ recebe_status
1073
1074  bool recebe_status(void)
1075  {
1076    // Recebe os dados dos grupos e refletores
1077    str = "";
1078    if (!aguarda_serial(2,1000,1,19)) return false;              // Ponto de chamada 19
1079    while (Serial2.available() > 0)                              // Aguarda chegar alguma coisa
1080    {
1081      while (Serial2.peek() != '#') str += String(Serial2.read());
1082      Serial1.print(str.c_str());                                // Envia linha ao PC
1083      Serial2.read();                                            // Descarta #
1084    }
1085    return true;
1086  }
1087
1088  //_____________________________________________________________________________________________ pede_p1
1089
1090  bool pede_p1(uint8_t sensor)
1091  {
1092    // Solicita dados a P1
1093    switch (sensor)
1094    {
1095        case 1:                                                  // Pede data e hora a p1
1096          Serial2.print("$(1)");
1097
1098          if (!aguarda_serial(2,1000,14,7))                      // Serial2,1seg,14 caracteres,ponto de chamada 7
1099          {
1100            Serial.print("\nP1 nao enviou data e hora");
1101            return false;
1102          }
1103          dia = Serial2.parseInt();
1104          mes = Serial2.parseInt();
```

```
1105          ano = Serial2.parseInt();
1106          horas = Serial2.parseInt();
1107          minutos = Serial2.parseInt();
1108          segundos = Serial2.parseInt();
1109          envia_PC(1);
1110          limpa_buffer_serial(2,10);
1111          return true;
1112        case 2:
1113          Serial2.print("$(2)");                          // Vamos pedir os dados dos INA
1114          if (!recebe_inas())                             // Recede os dados
1115          {
1116            Serial.print("\nP1 nao enviou dados das INAs");       // P1 não enviou dados
1117            return false;
1118          }
1119          envia_PC(2);
1120          return true;
1121        case 3:
1122          Serial2.print("$(3)");                          // Pede o status dos grupos e refletores
1123          if (!recebe_status())                           // Recede os dados
1124          {
1125            Serial.print("\nP1 nao enviou dados de status");      // P1 não enviou dados
1126            return false;
1127          }
1128          return true;
1129        default:
1130          Serial.print("\nCodigo invalido - pede_p1");
1131          return false;
1132    }
1133  }
1134
1135  //_____________________________________________________________________________________________________ envia_PC
1136
1137  void envia_PC(unsigned int kk)
1138  {
1139    // Envia dados ao PC
1140    switch (kk)
1141    {
1142      case 1:                                             // Data e hora
1143        Serial1.print("\n");
1144        Serial1.print(dia,DEC);
1145        Serial1.print('/');
1146        Serial1.print(mes,DEC);
1147        Serial1.print('/');
1148        Serial1.print(ano,DEC);
1149        Serial1.print(' ');
1150        Serial1.print(horas,DEC);
```

```
1151        Serial1.print(':');
1152        Serial1.print(minutos,DEC);
1153        Serial1.print(':');
1154        Serial1.print(segundos,DEC);
1155        break;
1156      case 2:                                                  // Dados das inas
1157        Serial1.print("\nvina1 = ");
1158        Serial1.print(vina1,2);
1159        Serial1.print(" cina1 = ");
1160        Serial1.print(cina1,2);
1161        Serial1.print(" pina1 = ");
1162        Serial1.print(pina1,2);
1163        Serial1.print("\nvina2 = ");
1164        Serial1.print(vina2,2);
1165        Serial1.print(" cina2 = ");
1166        Serial1.print(cina2,2);
1167        Serial1.print(" pina2 = ");
1168        Serial1.print(pina2,2);
1169        Serial1.print("\nvina3 = ");
1170        Serial1.print(vina3,2);
1171        Serial1.print(" cina3 = ");
1172        Serial1.print(cina3,2);
1173        Serial1.print(" pina3 = ");
1174        Serial1.print(pina3,2);
1175        Serial1.print("\netotalin = ");
1176        Serial1.print(etotalin,2);
1177        Serial1.print(" etotalout = ");
1178        Serial1.print(etotalout,2);
1179        break;
1180      default:
1181        Serial.print("\nCodigo invalido - envia_PC");
1182        break;
1183    }
1184  }
1185
1186  //_____________________________________________________________________________________________________________ mostra_LCD03
1187
1188  void mostra_LCD03(void)
1189  {
1190    // Mostra dados no LCD03
1191    cont1++;
1192    switch (cont1)
1193    {
1194      case 1:
1195        clear_line(1);
1196        clear_line(2);
```

```
1197        clear_line(3);
1198        display_texto_lc(" vina1 = ",1,1);
1199        display_float(vina1);
1200        display_texto_lc(" cina1 = ",2,1);
1201        display_float(cina1);
1202        display_texto_lc(" pina1 = ",3,1);
1203        display_float(pina1);
1204        break;
1205      case 2:
1206        clear_line(1);
1207        clear_line(2);
1208        clear_line(3);
1209        display_texto_lc(" vina2 = ",1,1);
1210        display_float(vina2);
1211        display_texto_lc(" cina2 = ",2,1);
1212        display_float(cina2);
1213        display_texto_lc(" pina2 = ",3,1);
1214        display_float(pina2);
1215        break;
1216      case 3:
1217        clear_line(1);
1218        clear_line(2);
1219        clear_line(3);
1220        display_texto_lc(" vina3 = ",1,1);
1221        display_float(vina3);
1222        display_texto_lc(" cina3 = ",2,1);
1223        display_float(cina3);
1224        display_texto_lc(" pina3 = ",3,1);
1225        display_float(pina3);
1226        break;
1227      case 4:
1228        clear_line(1);
1229        clear_line(2);
1230        clear_line(3);
1231        display_texto_lc(" etotalin = ",1,1);
1232        display_float(etotalin);
1233        display_texto_lc(" etotalout = ",2,1);
1234        display_float(etotalout);
1235        cont1 = 0;
1236        break;
1237      default:
1238        Serial.println("Contagem invalida");
1239        break;
1240    }
1241  }
1242
```

```
1243  //_____________________________________________________________________________________________________ isr1
1244
1245  void IRAM_ATTR isr1()
1246  {
1247    // Rotina para processamento do interrupt do keypad
1248    pedido_keypad = true;
1249  }
1250
1251  //_____________________________________________________________________________________________________ isr2
1252
1253  void IRAM_ATTR isr2()
1254  {
1255    // Rotina para processamento do interrupt de liga/desliga tudo
1256    liga_desliga = true;
1257  }
1258
1259  //_____________________________________________________________________________________________________ limpa_buffer_serial
1260
1261  void limpa_buffer_serial(uint8_t ser,uint8_t cmp)
1262  {
1263    // Limpa o buffer das seriais
1264    uint8_t i = 1;
1265
1266    while (i <= cmp)
1267    {
1268      if (ser == 1) {Serial1.read();i++;}
1269      else if (ser == 2) {Serial2.read();i++;}
1270      else return;
1271    }
1272    delay(1000);
1273  }
1274
1275  //_____________________________________________________________________________________________________ aguarda_serial
1276
1277  bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde)
1278  {
1279    // Aguarda a serial enviar ncar caracteres no tempo intervalo mseg. Indica o ponto do programa que chamou - deonde
1280    timeElapsed = 0;                                    // Inicia contagem de tempo
1281    while(timeElapsed < intervalo)                      // Executa enquanto não passar o intervalo
1282    {
1283      if (ser == 1)                                     // Serial1
1284      {
1285        if (Serial1.available() < ncar);
1286        else return true;
1287      }
1288      else if (ser == 2)                                // Serial2
```

```
1289      {
1290        if (Serial2.available() < ncar);
1291        else return true;
1292      }
1293      else                                                  // Serial inválida
1294      {
1295        Serial.print("\nSerial ");
1296        Serial.print(ser);
1297        Serial.print(" invalida, chamada de ");
1298        Serial.print(deonde);
1299        toca_buzzer(5);
1300        return false;
1301      }
1302    }
1303    Serial.print("\nTime Out esperando Serial");              // Time out
1304    Serial.print(ser);
1305    Serial.print(" chamada de ");
1306    Serial.print(deonde);
1307    toca_buzzer(5);
1308    return false;
1309  }
1310
1311  //_____________________________________________________________________________________________________ dias_desde_01_01_2019
1312
1313  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano)
1314  {
1315    // Retorna o número de dias desde 01/01/2019
1316    unsigned int an,nd = 0;
1317    unsigned int dd[14] = {0,0,31,59,90,120,151,181,212,243,273,304,334,365};
1318
1319    for (an=2019;an<ano;an++)
1320    {
1321      if ((an % 4) == 0) nd += 366;                         // É ano bissexto (vale até 2100)
1322      else nd += 365;
1323    }
1324    if (((an % 4) == 0) && (mes > 2)) nd += (dd[mes] + dia + 1);  // É ano bissexto (vale até 2100)
1325    else nd += (dd[mes] + dia);
1326    return nd;
1327  }
1328
1329  //_____________________________________________________________________________________________________ toca_buzzer
1330
1331  void toca_buzzer(unsigned int seg)
1332  {
1333    // Toca o buzzer por seg segundos
1334    long intervalo = (long)(seg * 1000);
```

```
1335
1336    timeElapsed = 0;                                    // Inicia contagem de tempo
1337    while(timeElapsed < intervalo)                      // Executa enquanto não passar o intervalo
1338    {
1339      digitalWrite(buzzer, HIGH);
1340      delay(500);
1341      digitalWrite(buzzer, LOW);
1342      delay(500);
1343    }
1344  }
1345
1346  //_____________________________________________________________________________________________________ conecta_MQTT
1347
1348  void conecta_MQTT(void)
1349  {
1350    // Conecta MQTT
1351    if (!MQTT.connected())                              // Se não estiver conectado
1352    {
1353      long now = millis();
1354      if (now - lastReconnectAttempt > 5000)
1355      {
1356        lastReconnectAttempt = now;
1357        if (reconnect())                                // Tentativa de reconexão
1358        {
1359          Serial.print("\nBroker MQTT conectado");
1360          lastReconnectAttempt = 0;
1361        }
1362      }
1363    }
1364  }
1365
1366  //_____________________________________________________________________________________________________ reconnect
1367
1368  bool reconnect()
1369  {
1370    // Reconecta MQTT
1371    if (MQTT.connect("quiosque",ID_MQTT,PSW_MQTT))
1372    {
1373      MQTT.publish(aviso1,"conectado");                 // Quando reconectado, publica aviso
1374    }
1375    return MQTT.connected();
1376  }
1377
1378  //_____________________________________________________________________________________________________ conecta_wifi
1379
1380  void conecta_wifi(unsigned char pr)
```

```
1381  {
1382    // Conecta WiFi
1383    WiFi.begin(ssid, password);                          // Inicia WiFi
1384    while (WiFi.status() != WL_CONNECTED)                // Aguarda conexão
1385    {
1386      delay(500);
1387      Serial.println("Conectando WiFi..");
1388    }
1389    Serial.print("Connectado na rede ");                 // Conectado
1390    Serial.println(ssid);
1391    if (pr == 1)
1392    {
1393      IPAddress ip = WiFi.localIP();                     // Imprime o IP
1394      Serial.print("IP: ");
1395      Serial.println(ip);
1396      long rssi = WiFi.RSSI();                           // Imprime o nível do sinal
1397      Serial.print("Nivel do sinal (RSSI): ");
1398      Serial.println(rssi);
1399    }
1400  }
1401
1402  //_____________________________________________________________________________________________________ recebido_P1
1403
1404  bool recebido_P1(unsigned long tp,uint8_t lg)
1405  {
1406    // Aguarda ACK de P1
1407    if (!aguarda_serial(2,tp,1,lg)) return false;        // Serial2,tp seg,1 caractere,ponto de chamada,time out
1408    else if (Serial2.read() != 0x6) return false;        // Não é ACK
1409    else return true;                                    // É ACK
1410  }
1411
1412  //_____________________________________________________________________________________________________
1413
```