```
/*****************************************************************************************************
* 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 V2.5.4 (07/11/2019)
  * 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
  * 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, enviando para a nuvem via MQTT (cloudMQTT) e ThingSpeak (para analise com MATLAB) e enviando, via
38    * Xbee, para o ESP32 da Parte 2. Grava no SD as voltagens máximas e mínimas das INAs e energia gerada e consumida em
39    * 24h (para análise e calibração de ciclos liga/desliga). Em função desses dados, da fotocélula, dos dados
40    * configurados pelo ESP32 remoto (Parte 2 via Xbee), controla e disponibiliza energia em horários pré-configurados
41    * para os refletores, executa comandos e solicitações do ESP32 remoto e gerencia a energia fornecida pelas baterias,
42    * otimizando-as.
43
44    Parte 2 - Laboratorio (na residência)
45
46    ESP32 remoto
47    * Esta parte utiliza uma placa DOIT ESP32 DEVKIT V1, uma eeprom At24c256 - 32KB, um display LCD03 com keypad, um
48    * buzzer, dois botões NA, um Xbee, um PC/keyboard/mouse e um celular. Ela atua como servidor AP para comunicação com
49    * o celular (o qual envia comandos para acionar os grupos de refletores) e como entrada de configurações, solicitações
50    * e comandos, via PC, para a Parte 1 via Xbee. Além disso, faz o display, no LCD03, dos dados enviados pela parte 1
51    * e comanda os grupos de refletores pelo keypad.
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 <HardwareSerial.h>                                       // Segunda UART
63  #include <Wire.h>                                                 // I2C
64  #include <LCD03_N1.h>                                             // LCD03
65  #include <elapsedMillis.h>                                        // Intervalos de tempo
66
67  //_____________________________________________________________________________________________________________
68
69  #define i2c_addr_eeprom 0x50                                      // Endereco I2C da 24LC256
70
71  template <class T> int eeWrite(int ee, const T& value)            // Classe genérica para gravação da eeprom
72  {                                                                 // ee - posição na eeprom, value - estrutura de dados
```

```
73    const byte* p = (const byte*)(const void*)&value;          // Pointer para a estrutura
74    int i;                                                     // Contador
75    Wire.beginTransmission(i2c_addr_eeprom);                   // Inicia transmissão I2C
76    Wire.write((int)(ee >> 8)); // MSB                         // Envia MSB da posição
77    Wire.write((int)(ee & 0xFF)); // LSB                       // Envia LSB da posição
78    for (i = 0; i < sizeof(value); i++)                        // Para todos os bytes da estrutura
79    Wire.write(*p++);                                          // Envia o byte
80    Wire.endTransmission();                                    // Encerra transmissão I2C
81    return i;                                                  // Retorna o número de bytes gravados
82  }
83
84  template <class T> int eeRead(int ee, T& value)              // Classe genérica para leitura da eeprom
85  {                                                            // ee - posição na eeprom, value - estrutura de dados
86    byte* p = (byte*)(void*)&value;                            // Pointer para a estrutura
87    int i;                                                     // Contador
88    Wire.beginTransmission(i2c_addr_eeprom);                   // Inicia transmissão I2C
89    Wire.write((int)(ee >> 8)); // MSB                         // Envia MSB da posição
90    Wire.write((int)(ee & 0xFF)); // LSB                       // Envia LSB da posição
91    Wire.endTransmission();                                    // Encerra transmissão I2C
92    Wire.requestFrom(i2c_addr_eeprom,sizeof(value));           // Solicita à eeprom o número de bytes da estrutura
93    for (i = 0; i < sizeof(value); i++)                        // Para esse número de bytes
94    if(Wire.available())                                       // Se tem dados no buffer
95    *p++ = Wire.read();                                        // Lê o byte para a estrutura
96    return i;                                                  // Retorna o número de bytes lidos
97  }
98
99  struct eeprom                                                // Estrutura de dados na eeprom
100 {
101   unsigned long valido;                                      // Código para validar dados
102   unsigned long inter1;                                      // Intervalo de tempo para "elapsedMillis timeElapsed1;"
103   unsigned long inter2;                                      // Intervalo de tempo para "elapsedMillis timeElapsed2;"
104   unsigned int cod_mon;                                      // Código para imprimir ou não no monitor serial
105 } eeprom;
106
107 //_____________________________________________________________________________________________________________________
108
```

```
109  // Variaveis e constantes
110  #define SDApin 21                                   // I2C SDA
111  #define SCLpin 22                                   // I2C SCL
112  #define RX1pin 16                                   // UART1 RX
113  #define TX1pin 17                                   // UART1 TX
114  #define RX2pin 4                                    // UART2 RX
115  #define TX2pin 2                                    // UART2 TX
116  #define liga_desliga_tudo 26                        // Pino de interrupt para ligar ou desligar tudo
117  #define keypadpin 25                                // Pino de interrupt para o keypad
118  #define buzzer 27                                   // Pino para acionamento do buzzer
119  const byte DNS_PORT = 53;
120  const char *ssid = "CoMoSisFog";
121  const char *password = "villaadriana";
122  unsigned long key = 2863311530;
123  uint8_t cont1;
124  volatile bool pedido_keypad;
125  volatile bool liga_desliga;
126  unsigned int k;
127  unsigned int dia;
128  unsigned int mes;
129  unsigned int ano;
130  unsigned int horas;
131  unsigned int minutos;
132  unsigned int segundos;
133  char entrada[21];                                   // Vetor para receber digitação (1 linha do LCD03)
134  String str;
135  char sp;
136  unsigned int d1;
137  unsigned int m1;
138  unsigned int a1;
139  unsigned int d2;
140  unsigned int m2;
141  unsigned int a2;
142
143  float vina1,cina1,pina1,vina2,cina2,pina2,vina3,cina3,pina3,etotalin,etotalout;
144
```

```
145  //_____________________________________________________________________________________________________________________
146
147  IPAddress local_ip(192, 168, 5, 1);                          // IP do servidor AP
148  IPAddress gateway(192, 168, 5, 1);                           // Gateway do servidor AP
149  IPAddress subnet(255, 255, 255, 0);                          // Subnet do servidor AP
150
151  //_____________________________________________________________________________________________________________________
152
153  HardwareSerial Seria1(1);                                    // Instancia HardwareSerial para PC
154  HardwareSerial Seria2(2);                                    // Instancia HardwareSerial para Xbee
155  DNSServer dnsServer;                                         // Instancia DNSServer para nome de dominio
156  WebServer server(80);                                        // Instancia WebServer para serevidor AP
157  elapsedMillis timeElapsed1;                                  // Instancia elapsedMillis para eeprom.inter1
158  elapsedMillis timeElapsed2;                                  // Instancia elapsedMillis para eeprom.inter2
159  elapsedMillis timeElapsed;                                   // Instancia elapsedMillis generico
160
161  //_____________________________________________________________________________________________________________________
162
163  void recebe_status(void);                                    // Recebe os dados dos grupos e refletores
164  void recebe_inas(void);                                      // Recebe dados das INAs
165  void pede_p1(uint8_t sensor);                                // Pede dados à Parte 1
166  void envia_PC(unsigned int kk);                              // Envia dados ao PC
167  void mostra_LCD03(void);                                     // Mostra dados no LCD03
168  void limpa_buffer_serial(uint8_t ser,uint8_t cmp);           // Limpa buffer das seriais
169  bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde); // Aguarda caracteres nas seriais
170  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano); // Calcula número de dias
171  void IRAM_ATTR isr1();                                       // Rotina de interrupt para botão de pedido do keypad
172  void IRAM_ATTR isr2();                                       // Rotina de interrupt para botão de toggle tudo
173  void toca_buzzer(unsigned int seg);                          // Rotina para tocar o buzzer seg segundos
174  String SendHTML(void) ;                                      // Para o servidor AP
175  void handle_NotFound(void);                                  // Idem os abaixo
176  void handle_grupo1(void);
177  void handle_grupo2(void);
178  void handle_grupo3(void);
179  void handle_grupo4(void);
180  void handle_grupo5(void);
```

```
181  void handle_grupo6(void);
182  void handle_grupo7(void);
183  void handle_grupo8(void);
184  void handle_grupo9(void);
185  void handle_grupo10(void);
186  void handle_grupo11(void);
187  void handle_grupo12(void);
188  void handle_OnConnect(void);
189
190  //_____________________________________________________________________________________________________________
191
192  void setup()
193  {
194
195    Serial.begin(115200);                                    // Monitor serial
196    while (!Serial);
197    Serial.println("\nCoMoSisFog V2.5 parte 2");
198
199    Wire.begin(SDApin,SCLpin);                               // I2C ESP32 - SDA 21 SCL 22
200    LCD03_begin(0xC6,10);                                    // Inicia o LCD03
201    Serial.println("LCD03 inicializado");
202
203    backlight_on;
204    clear_screen;
205    hide_cursor;
206    display_texto_lc(" CoMoSisFog V2.5",1,1);
207    display_texto_lc(" Filippo Pardini",2,1);
208    delay (1000);
209
210    WiFi.softAP(ssid, password, 13, false, 1);               // Parametros de acesso ao servidor AP
211    delay(100);
212    WiFi.softAPConfig(local_ip, gateway, subnet);            // Parametros de acesso ao servidor AP
213
214    dnsServer.setTTL(300);                                   // Servidor DNS
215    dnsServer.setErrorReplyCode(DNSReplyCode::ServerFailure);// Servidor DNS
216    dnsServer.start(DNS_PORT, "comosisfog.net", local_ip);   // Servidor DNS
```

```
217
218     Serial2.begin(115200, SERIAL_8N1, RX1pin, TX1pin);            // Serial1 para Xbee - ESP32 - RX 16 TX 17
219     Serial2.setTimeout(10000);
220     Serial.println("Serial para Xbee inicializada");
221
222     Serial1.begin(115200, SERIAL_8N1, RX2pin, TX2pin);            // Serial2 para PC - ESP32 - RX 4 TX 2
223     Serial1.setTimeout(10000);
224     Serial.println("Serial para PC inicializada");
225
226     // Atribuição dos handles para o servidor AP
227
228     server.on("/", HTTP_GET, handle_OnConnect);
229     server.on("/grupo1", HTTP_GET, handle_grupo1);
230     server.on("/grupo2", HTTP_GET, handle_grupo2);
231     server.on("/grupo3", HTTP_GET, handle_grupo3);
232     server.on("/grupo4", HTTP_GET, handle_grupo4);
233     server.on("/grupo5", HTTP_GET, handle_grupo5);
234     server.on("/grupo6", HTTP_GET, handle_grupo6);
235     server.on("/grupo7", HTTP_GET, handle_grupo7);
236     server.on("/grupo8", HTTP_GET, handle_grupo8);
237     server.on("/grupo9", HTTP_GET, handle_grupo9);
238     server.on("/grupo10", HTTP_GET, handle_grupo10);
239     server.on("/grupo11", HTTP_GET, handle_grupo11);
240     server.on("/grupo12", HTTP_GET, handle_grupo12);
241     server.onNotFound(handle_NotFound);
242
243     server.begin();                                               // Inicializa servidor AP
244     Serial.println("web server e DNS inicializados");
245
246   // Força eeprom invalida para inicializar default **************
247   // Depois lembrar de retirar este codigo
248     //eeprom.valido = 0;
249     //eeprom.inter1 = 0;
250     //eeprom.inter2 = 0;
251     //eeprom.cod_mon = 0;
252     //eeWrite(0,eeprom);                                          // Grava eeprom
```

```
253     //delay (30);
254   // Até aqui ****************************************************
255
256     eeRead(0,eeprom);                                               // Le eeprom
257     delay (30);
258
259     if (eeprom.valido != key)                                       // Eeprom invalida, força reinicializacao default
260     {
261       eeprom.inter1 = 60000;                                        // Default - 1 minuto
262       eeprom.inter2 = 300000;                                       // Default - 5 minutos
263       eeprom.cod_mon = 1;                                           // Habilita monitor serial
264       eeprom.valido = key;                                          // Valida eeprom
265
266       eeWrite(0,eeprom);                                            // Grava eeprom
267       delay (30);
268       Serial.println("eeprom default");
269     }
270
271     timeElapsed1 = 0;
272     timeElapsed2 = 0;
273     timeElapsed = 0;
274     cont1 = 0;
275     pedido_keypad = false;
276     pinMode(keypadpin, INPUT_PULLUP);                               // Pino a ser acionado quando for teclar no keypad
277     attachInterrupt(digitalPinToInterrupt(keypadpin),isr1,FALLING); // Define interrupt no pino keypadpin
278     liga_desliga = false;
279     pinMode(liga_desliga_tudo, INPUT_PULLUP);                       // Pino a ser acionado quando for toggle tudo
280     attachInterrupt(digitalPinToInterrupt(liga_desliga_tudo),isr2,FALLING); // Define interrupt no pino liga_desliga_tudo
281     pinMode(buzzer, OUTPUT);                                        // Pino para acionar o buzzer
282     digitalWrite(buzzer, LOW);                                      // Inicialmente desativado
283   }
284
285   //_____________________________________________________________________________________________________________________
286
287   void loop()
288   {
```

```
289     dnsServer.processNextRequest();
290     server.handleClient();
291
292   //_____________________________________________ Continuamente ____________________________________________________
293
294     if (Serial2.available() > 10)                                   // Parte 1 enviou algo?
295     {
296       // Verifica se a Parte 1 enviou mensagem - formato - <texto>
297       if (Serial2.read() == '<')
298       {
299         str = " " + Serial2.readStringUntil('>');
300         clear_screen;
301         display_string_lc(&str,1,1);                                // Display no LCD03
302         toca_buzzer(3);                                             // Toca buzzer
303         while (read_keypad() != '#')
304         {
305           delay(3000);
306           toca_buzzer(3);
307         }
308       }
309     }
310
311     //_____________________________________________________________________________________________________________
312
313     if (Serial1.available() >= 4)                                   // Pc enviou algo?
314     {
315       // Verifica se o PC enviou alguma coisa: comando, coniguração p1, coniguração p2, configuração dos grupos de
316       // refletores, configuração de seus modos ou solicitação de dados. Ou a Parte 1 enviou mensagem.
317       // Indicadores válidos:
318       // '!' - Comando toggle dos grupos de refletores - formato - !(g) - onde g é o número do grupo.  g = 0 => tudo
319       // '$' - Solicitação de dados - formato - $(i) - onde i=1 data-hora, i=2 dados INAs e energias, i=3 status dos
320       //       grupos e refletores
321       // '@' - Solicitação de dados - formato - @(di,df) - onde di - data de início, df - data de fim, ambos no formato
322       //       dia,mes,ano. Ex: @(1,8,2019/31,8,2019) vai ler o SD e enviar os dados relativos a esse período
323       // '%' - Configuração de ciclos em p2 - formato - %n(v) - onde n=1 eeprom.inter1, n=2 eeprom.inter2,
324       //       n=3 eepro.cod_mon, v = 0 ou 1 ou valor em minutos
```

```
325       // '#' - Configuração de ciclos em p1 - formato - #n(v) - onde n=1 inter1, n=2 inter2,
326       //       n=3 cod_mon, v = 0 ou 1 ou valor em minutos
327       // '&' - Configuração dos grupos de refletores - formato - &n(r,...,r) - onde n é o número do grupo
328       //       e r o número do refletor
329       // '*' - Configuração dos horarios liga/desliga dos grupos e modo de operação de cada grupo
330       //       Modo de operação: ativo -> sim(1)/não(0), automatico -> fotocelula, programado -> hora liga/desliga,
331       //       manual -> comando de toggle via p2,PC ou celular - formato - *n(a,b,c,d) - onde n é o número do grupo
332       //       a=0 -> inativo, a=1 => ativo, b=1 -> automático(fotocelula), b=2 -> manual(comando PC,p2,celular),
333       //       b=3 -> programado (c e d só existem neste caso), c -> hora liga (formato hh:mm hh -> hora formato 24,
334       //       mm -> minutos), d -> hora desliga (idem hora liga)
335
336       sp = Serial1.peek();
337
338       switch (sp)
339       {
340         case '%':
341           if (!aguarda_serial(1,10000,5,1)) break;              // Serial1,10seg,5 caracteres,ponto de chamada 1
342           switch(Serial1.parseInt())                            // Configura ciclos em p2
343           {
344             case 1:                                             // Interval1_p2 em minutos
345               eeprom.inter1 = Serial1.parseInt() * 60000;       // Passa para milisegundos
346               break;
347             case 2:                                             // Interval2_p2 em minutos
348               eeprom.inter2 = Serial1.parseInt() * 60000;       // Passa para milisegundos
349               break;
350             case 3:                                             // eeprom.cod_mon
351               eeprom.cod_mon = Serial1.parseInt();
352               break;
353             default:
354               Serial.println("Codigo invalido 1");
355               break;
356           }
357           eeWrite(0,eeprom);                                    // Grava eeprom
358           delay (30);
359           Serial.println("eeprom gravada 1");
360           break;
```

```
361         case '$':
362           if (!aguarda_serial(1,10000,4,2)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 2
363           k = Serial1.parseInt();
364           pede_p1(k);
365           envia_PC(k);
366           Serial.println("Pediu P1 e enviou PC");
367           break;
368         case '!':
369           if (!aguarda_serial(1,10000,4,3)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 3
370           str = Serial1.readStringUntil(')') + ')';
371           Serial2.print(str);
372           Serial.print("\nEnviou para P1: "); Serial.print(str); Serial.print("\n");
373           break;
374         case '#':
375           if (!aguarda_serial(1,10000,5,4)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 4
376           str = Serial1.readStringUntil(')') + ')';
377           Serial2.print(str);
378           Serial.print("\nEnviou para P1: "); Serial.print(str); Serial.print("\n");
379           break;
380         case '&':
381           if (!aguarda_serial(1,10000,5,5)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 5
382           str = Serial1.readStringUntil(')') + ')';
383           Serial2.print(str);
384           Serial.print("\nEnviou para P1: "); Serial.print(str); Serial.print("\n");
385           break;
386         case '*':
387           if (!aguarda_serial(1,10000,6,6)) break;                // Serial1,10seg,4 caracteres,ponto de chamada 6
388           str = Serial1.readStringUntil(')') + ')';
389           Serial2.print(str);
390           Serial.print("\nEnviou para P1: "); Serial.print(str); Serial.print("\n");
391           break;
392         case '@':
393           d1 = Serial1.parseInt();
394           m1 = Serial1.parseInt();
395           a1 = Serial1.parseInt();
396           d2 = Serial1.parseInt();
```

```
397           m2 = Serial1.parseInt();
398           a2 = Serial1.parseInt();
399
400           str = String(dias_desde_01_01_2019(d1,m1,a1)) + ',' + String(dias_desde_01_01_2019(d2,m2,a2));
401           Serial2.print(str);
402           Serial.print("\nEnviou para P1: "); Serial.print(str); Serial.print("\n");
403
404           // Vamos aguardar a Parte 1 enviar o que foi pedido e mandar para o PC
405           while (Serial2.available() > 0)
406           {
407             while(Serial2.available() < 40);
408             str = Serial2.readStringUntil('>');
409             Serial1.print("\n"+str);
410             Serial2.read();
411           }
412           break;
413         default:
414           Serial.print("\nCodigo invalido ");
415           Serial.print(sp);
416           break;
417       }
418       limpa_buffer_serial(1,10);
419     }
420
421     //_____________________________________________________________________________________________________________
422
423     // Verifica se foi apertado o botão toggle tudo
424
425     if (liga_desliga)                                                   // Toggle tudo
426     {
427       Serial2.print("!(0)");                                           // Envia a p1 toggle tudo
428       Serial.println("\nEnviou pedido de toggle tudo a P1");
429       liga_desliga = false;
430     }
431
432     //_____________________________________________________________________________________________________________
```

```
433
434     // Verifica se veio solicitação do keypad para teclar comando de ligar/desligar grupo de refletores (códigos 1 a 12)
435     // Podemos utilizar os códigos > 12 para comandos de outra natureza.
436     if (pedido_keypad)
437     {
438       if(entrada_teclado(entrada))                                  // Recebe digitação. A leitura é encerrada
439       {                                                             // quando é digitado #
440         String este = String(entrada);
441         if (este.toInt() > 12)
442         {
443           // Comando > 12 , programe aqui se precisar retirando o comando abaixo
444           Serial.print("\nComando > 12 nao programado");
445         }
446         else
447         {
448           display_vectorchar_lc(entrada,1,1);                       // Display do que recebeu linha 1 coluna 1
449           Serial_print(entrada,1,5);                                // Imprime na serial o que recebeu a partir da posição 1
450                                                                     // por 5 posições
451           Serial2.print("!(");
452           Serial2.print(entrada);                                   // Envia a p1 o número do grupo a ser toggled *********
453           Serial2.print(')');
454           Serial.print("\nEnviou pedido de toggle a P1");
455         }
456       }
457       else
458       {
459         Serial.print("\nAlgo nao funcionou");
460       }
461       pedido_keypad = false;
462     }
463
464   //_____________________________________________ Ciclos de tempo ____________________________________________________
465
466     if (timeElapsed1 > eeprom.inter1)                               // Executa a cada ciclo de duracao eeprom.inter1
467     {
468       pede_p1(1);                                                   // dd/mm/aaaa-hh:mm:ss
```

```
469
470       timeElapsed1 = 0;                                     // Reinicia ciclo
471     }
472
473     if (timeElapsed2 > eeprom.inter2)                       // Executa a cada ciclo de duracao eeprom.inter2
474     {
475       Serial2.print("$(2)");                                // Vamos pedir os dados dos INA
476       recebe_inas();                                        // Recede os dados
477       mostra_LCD03();                                       // Mostra no LCD03
478
479       timeElapsed2 = 0;                                     // Reinicia ciclo
480     }
481   }
482
483   //_____________________________________________________________________________________________________________________
484
485   void handle_OnConnect(void)
486   {
487     Serial.println("Cliente conectado");
488     server.send(200, "text/html", SendHTML());
489   }
490
491   //_____________________________________________________________________________________________________________________
492
493   // HTTP request: Grupo 1
494   void handle_grupo1(void)
495   {
496     Serial.println("Grupo 1");
497     Serial2.print("!(1)");                                   // Grupo 1 contém todos os refletores liga/desliga tudo
498     server.send(200, "text/html", SendHTML());
499   }
500
501   //_____________________________________________________________________________________________________________________
502
503   // HTTP request: Grupo 2
504   void handle_grupo2(void)
```

```
505   {
506     Serial.println("Grupo 2");
507     Serial2.print("!(2)");
508     server.send(200, "text/html", SendHTML());
509   }
510
511   //________________________________________________________________________________________________________________________
512
513   // HTTP request: Grupo 3
514   void handle_grupo3(void)
515   {
516     Serial.println("Grupo 3");
517     Serial2.print("!(3)");
518     server.send(200, "text/html", SendHTML());
519   }
520
521   //________________________________________________________________________________________________________________________
522
523   // HTTP request: Grupo 4
524   void handle_grupo4(void)
525   {
526     Serial.println("Grupo 4");
527     Serial2.print("!(4)");
528     server.send(200, "text/html", SendHTML());
529   }
530
531   //________________________________________________________________________________________________________________________
532
533   // HTTP request: Grupo 5
534   void handle_grupo5(void)
535   {
536     Serial.println("Grupo 5");
537     Serial2.print("!(5)");
538     server.send(200, "text/html", SendHTML());
539   }
540
```

```
541  //_____________________________________________________________________________________________________________________
542
543  // HTTP request: Grupo 6
544  void handle_grupo6(void)
545  {
546    Serial.println("Grupo 6");
547    Serial2.print("!(6)");
548    server.send(200, "text/html", SendHTML());
549  }
550
551  //_____________________________________________________________________________________________________________________
552
553  // HTTP request: Grupo 7
554  void handle_grupo7(void)
555  {
556    Serial.println("Grupo 7");
557    Serial2.print("!(7)");
558    server.send(200, "text/html", SendHTML());
559  }
560
561  //_____________________________________________________________________________________________________________________
562
563  // HTTP request: Grupo 8
564  void handle_grupo8(void)
565  {
566    Serial.println("Grupo 8");
567    Serial2.print("!(8)");
568    server.send(200, "text/html", SendHTML());
569  }
570
571  //_____________________________________________________________________________________________________________________
572
573  // HTTP request: Grupo 9
574  void handle_grupo9(void)
575  {
576    Serial.println("Grupo 9");
```

```
577     Serial2.print("!(9)");
578     server.send(200, "text/html", SendHTML());
579   }
580
581   //_____________________________________________________________________________________________________________
582
583   // HTTP request: Grupo 10
584   void handle_grupo10(void)
585   {
586     Serial.println("Grupo 10");
587     Serial2.print("!(10)");
588     server.send(200, "text/html", SendHTML());
589   }
590
591   //_____________________________________________________________________________________________________________
592
593   // HTTP request: Grupo 11
594   void handle_grupo11(void)
595   {
596     Serial.println("Grupo 11");
597     Serial2.print("!(11)");
598     server.send(200, "text/html", SendHTML());
599   }
600
601   //_____________________________________________________________________________________________________________
602
603   // HTTP request: Grupo 12
604   void handle_grupo12(void)
605   {
606     Serial.println("Grupo 12");
607     Serial2.print("!(12)");
608     server.send(200, "text/html", SendHTML());
609   }
610
611   //_____________________________________________________________________________________________________________
612
```

```
613   // HTTP request: other
614   void handle_NotFound(void)
615   {
616     Serial.println("Page not found");
617     server.send(404, "text/plain", "Not found");
618   }
619
620   //_____________________________________________________________________________________________________________________
621
622   String SendHTML(void)
623   {
624     String html = "<!DOCTYPE html>\n";
625     html += "<html>\n";
626     html += "<head>\n";
627     html += "<title>CoMoSisFog</title>\n";
628     html += "<meta name=\"viewport\" content=\"width=device-width, initial-scale=1\">\n";
629     html += "</head>\n";
630     html += "<body>\n";
631     html += "<div align=\"center\">\n";
632     html += "<h1>CoMoSisFog</h1>\n";
633     html += "<br>\n";
634     html += "<form method=\"GET\">\n";
635     html += "<input type=\"button\" value=\"Grupo 1\" onclick=\"window.location.href='/grupo1'\">\n";
636     html += "<br><br>\n";
637     html += "<input type=\"button\" value=\"Grupo 2\" onclick=\"window.location.href='/grupo2'\">\n";
638     html += "<br><br>\n";
639     html += "<input type=\"button\" value=\"Grupo 3\" onclick=\"window.location.href='/grupo3'\">\n";
640     html += "<br><br>\n";
641     html += "<input type=\"button\" value=\"Grupo 4\" onclick=\"window.location.href='/grupo4'\">\n";
642     html += "<br><br>\n";
643     html += "<input type=\"button\" value=\"Grupo 5\" onclick=\"window.location.href='/grupo5'\">\n";
644     html += "<br><br>\n";
645     html += "<input type=\"button\" value=\"Grupo 6\" onclick=\"window.location.href='/grupo6'\">\n";
646     html += "<br><br>\n";
647     html += "<input type=\"button\" value=\"Grupo 7\" onclick=\"window.location.href='/grupo7'\">\n";
648     html += "<br><br>\n";
```

```
649     html += "<input type=\"button\" value=\"Grupo 8\" onclick=\"window.location.href='/grupo8'\">\n";
650     html += "<br><br>\n";
651     html += "<input type=\"button\" value=\"Grupo 9\" onclick=\"window.location.href='/grupo9'\">\n";
652     html += "<br><br>\n";
653     html += "<input type=\"button\" value=\"Grupo 10\" onclick=\"window.location.href='/grupo10'\">\n";
654     html += "<br><br>\n";
655     html += "<input type=\"button\" value=\"Grupo 11\" onclick=\"window.location.href='/grupo11'\">\n";
656     html += "<br><br>\n";
657     html += "<input type=\"button\" value=\"Grupo 12\" onclick=\"window.location.href='/grupo12'\">\n";
658     html += "</form>\n";
659     html += "</div>\n";
660     html += "</body>\n";
661     html += "</html>\n";
662     return html;
663   }
664
665   //_____________________________________________________________________________________________________________________
666
667   void recebe_inas(void)
668   {
669     // Formato: ln(valor) onde l=v voltagem,l=c corrente,l=p potencia,l=e energia
670     // Para v,c,p -> n=1 ina1,n=2 ina2,n=3 ina3,valor ponto flutuante
671     // Para e -> n=i etotalin, n=o etotalout
672
673     while (Serial2.available() > 0)                                    // Aguarda chegar alguma coisa
674     {
675       switch (Serial2.read())
676       {
677         case 'v':
678           switch (Serial2.parseInt())
679           {
680             case 1:
681               vina1 = Serial2.parseFloat();
682               break;
683             case 2:
684               vina2 = Serial2.parseFloat();
```

```
685               break;
686             case 3:
687               vina3 = Serial2.parseFloat();
688               break;
689           }
690           Serial2.read();
691         break;
692         case 'c':
693           switch (Serial2.parseInt())
694           {
695             case 1:
696               cina1 = Serial2.parseFloat();
697               break;
698             case 2:
699               cina2 = Serial2.parseFloat();
700               break;
701             case 3:
702               cina3 = Serial2.parseFloat();
703               break;
704           }
705           Serial2.read();
706         break;
707         case 'p':
708           switch (Serial2.parseInt())
709           {
710             case 1:
711               pina1 = Serial2.parseFloat();
712               break;
713             case 2:
714               pina2 = Serial2.parseFloat();
715               break;
716             case 3:
717               pina3 = Serial2.parseFloat();
718               break;
719           }
720           Serial2.read();
```

```
721         break;
722         case 'e':
723           switch (Serial2.read())
724           {
725             case 'i':
726               pina1 = Serial2.parseFloat();
727               break;
728             case 'o':
729               pina2 = Serial2.parseFloat();
730               break;
731           }
732         break;
733       }
734     }
735   }
736
737   //_____________________________________________________________________________________________________________
738
739   void recebe_status(void)                                    // Recebe os dados dos grupos e refletores
740   {
741     while (Serial2.available() > 0)                           // Aguarda chegar alguma coisa
742     {
743       Serial1.print("\n");                                    // Inicia nova linha no PC
744       while (Serial2.peek() != '#') Serial1.print(Serial2.read());// Envia os dados ao PC
745     }
746   }
747
748   //_____________________________________________________________________________________________________________
749
750   void pede_p1(uint8_t sensor)
751   {
752     Serial.print("\nEntrou em pede_p1");
753     switch (sensor)
754     {
755         case 1:                                               // Pede data e hora a p1
756           Serial2.print("$(1)");
```

```
757           //while (Serial2.available() < 14);                    // dd/mm/aaaa,hh:mm:ss
758           if (!aguarda_serial(2,10000,14,7)) break;              // Serial2,10seg,14 caracteres,ponto de chamada 7
759           dia = Serial2.parseInt();
760           mes = Serial2.parseInt();
761           ano = Serial2.parseInt();
762           horas = Serial2.parseInt();
763           minutos = Serial2.parseInt();
764           segundos = Serial2.parseInt();
765
766           Serial.print("\ndia = ");
767           Serial.print(dia);
768           Serial.print("\nmes = ");
769           Serial.print(mes);
770           Serial.print("\nano = ");
771           Serial.print(ano);
772           Serial.print("\nhoras = ");
773           Serial.print(horas);
774           Serial.print("\nminutos = ");
775           Serial.print(minutos);
776           Serial.print("\nsegundos = ");
777           Serial.print(segundos);
778
779           limpa_buffer_serial(2,10);
780           break;
781         case 2:
782           Serial2.print("$(2)");                                  // Vamos pedir os dados dos INA
783           recebe_inas();                                         // Recede os dados
784           break;
785         case 3:
786           Serial2.print("$(3)");                                  // Vamos pedir o status dos grupos e refletores
787           recebe_status();                                       // Recede os dados
788           break;
789         default:
790           Serial.print("\nCodigo invalido pede_p1");
791           break;
792     }
```

```
793   }
794
795   //_____________________________________________________________________________________________________________________
796
797   void envia_PC(unsigned int kk)
798   {
799     switch (kk)
800     {
801       case 1:                                                   // Data e hora
802         Serial1.print("\n");
803         Serial1.print(dia,DEC);
804         Serial1.print('/');
805         Serial1.print(mes,DEC);
806         Serial1.print('/');
807         Serial1.print(ano,DEC);
808         Serial1.print('-');
809         Serial1.print(horas,DEC);
810         Serial1.print(':');
811         Serial1.print(minutos,DEC);
812         Serial1.print(':');
813         Serial1.print(segundos,DEC);
814         break;
815       case 2:                                                   // Dados das inas
816         Serial1.print("\nvina1 = ");
817         Serial1.print(vina1,2);
818         Serial1.print(" cina1 = ");
819         Serial1.print(cina1,2);
820         Serial1.print(" pina1 = ");
821         Serial1.print(pina1,2);
822         Serial1.print("\nvina2 = ");
823         Serial1.print(vina2,2);
824         Serial1.print(" cina2 = ");
825         Serial1.print(cina2,2);
826         Serial1.print(" pina2 = ");
827         Serial1.print(pina2,2);
828         Serial1.print("\nvina3 = ");
```

```
829         Serial1.print(vina3,2);
830         Serial1.print(" cina3 = ");
831         Serial1.print(cina3,2);
832         Serial1.print(" pina3 = ");
833         Serial1.print(pina3,2);
834         Serial1.print("\netotalin = ");
835         Serial1.print(etotalin,2);
836         Serial1.print(" etotalout = ");
837         Serial1.print(etotalout,2);
838         break;
839       default:
840         Serial.println("Codigo invalido");
841         break;
842     }
843   }
844
845   //_____________________________________________________________________________________________________________
846
847   void mostra_LCD03(void)
848   {
849     cont1++;
850     switch (cont1)
851     {
852       case 1:
853         clear_screen;
854         display_texto_lc(" vina1 = ",1,1);
855         display_float(vina1);
856         display_texto_lc(" cina1 = ",2,1);
857         display_float(cina1);
858         display_texto_lc(" pina1 = ",3,1);
859         display_float(pina1);
860         break;
861       case 2:
862         clear_screen;
863         display_texto_lc(" vina2 = ",1,1);
864         display_float(vina2);
```

```
865         display_texto_lc(" cina2 = ",2,1);
866         display_float(cina2);
867         display_texto_lc(" pina2 = ",3,1);
868         display_float(pina2);
869         break;
870       case 3:
871         clear_screen;
872         display_texto_lc(" vina3 = ",1,1);
873         display_float(vina3);
874         display_texto_lc(" cina3 = ",2,1);
875         display_float(cina3);
876         display_texto_lc(" pina3 = ",3,1);
877         display_float(pina3);
878         break;
879       case 4:
880         clear_screen;
881         display_texto_lc(" etotalin = ",1,1);
882         display_float(etotalin);
883         display_texto_lc(" etotalout = ",2,1);
884         display_float(etotalout);
885         cont1 = 0;
886         break;
887       default:
888         Serial.println("Contagem invalida");
889         break;
890     }
891   }
892
893   //_____________________________________________________________________________________________________________
894
895   void IRAM_ATTR isr1()                                    // Rotina para processamento do interrupt do pino
896   {
897     pedido_keypad = true;
898   }
899
900   //_____________________________________________________________________________________________________________
```

```
901
902 void IRAM_ATTR isr2()                                    // Rotina para processamento do interrupt do pino
903 {
904   liga_desliga = true;
905 }
906
907 //_____________________________________________________________________________________________________________
908
909 void limpa_buffer_serial(uint8_t ser,uint8_t cmp)
910 {
911   uint8_t i = 1;
912
913   while (i <= cmp)
914   {
915     if (ser == 1) {Serial1.read();i++;}
916     else if (ser == 2) {Serial2.read();i++;}
917     else return;
918   }
919   delay(1000);
920 }
921
922 //_____________________________________________________________________________________________________________
923
924 bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde)
925 {
926   timeElapsed = 0;
927   while(timeElapsed < intervalo)                           // Executa enquanto não passar o intervalo
928   {
929     if (ser == 1)
930     {
931       if (Serial1.available() < ncar);
932       else return true;
933     }
934     else if (ser == 2)
935     {
936       if (Serial2.available() < ncar);
```

```
937         else return true;
938       }
939       else
940       {
941         Serial.print("\nSerial");
942         Serial.print(ser);
943         Serial.print(" invalida");
944         Serial.print(" chamada de ");
945         Serial.print(deonde);
946         return false;
947       }
948     }
949     Serial.print("\nTime Out esperando Serial");
950     Serial.print(ser);
951     Serial.print(" chamada de ");
952     Serial.print(deonde);
953     return false;
954   }
955
956   //_____________________________________________________________________________________________________________
957
958   unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano)
959   {
960     unsigned int an,nd = 0;
961     unsigned int dd[14] = {0,0,31,59,90,120,151,181,212,243,273,304,334,365};
962
963     for (an=2019;an<ano;an++)
964     {
965       if ((an % 4) == 0) nd += 366;                              // É ano bissexto (vale até 2100)
966       else nd += 365;
967     }
968     if (((an % 4) == 0) && (mes > 2)) nd += (dd[mes] + dia + 1);  // É ano bissexto (vale até 2100)
969     else nd += (dd[mes] + dia);
970     return nd;
971   }
972
```

```
973  //_____________________________________________________________________________________________________________________
974
975  void toca_buzzer(unsigned int seg)
976  {
977    unsigned int tp = seg * 1000;
978    digitalWrite(buzzer, HIGH);                                    // Ativa buzzer
979    delay(tp);
980    digitalWrite(buzzer, LOW);                                     // Desativa buzzer
981  }
982
983  //_____________________________________________________________________________________________________________________
984
985
986
987
988
```