```
/*****************************************************************************************************
* 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 1 *******************************************
56   //*************************************************************************************************************
57
58   // Bibliotecas
59   #include <Wire.h>                                          // I2C
60   #include <INA226.h>                                        // INA226
61   #include "RTClib.h"                                        // DS3231M
62   #include <elapsedMillis.h>                                 // Intervalos de tempo
63   #include "SSD1306Wire.h"                                   // OLED
64   #include <HardwareSerial.h>                                // UART
65   #include <WiFi.h>                                          // WiFi
66   #include <PubSubClient.h>                                  // MQTT
67   #include "ThingSpeak.h"                                    // ThingSpeak
68   #include <ArduinoJson.h>                                   // Json
69   #include "FS.h"                                            // file system wrapper
70   #include "SD.h"                                            // micro SD
71   #include "SPI.h"                                           // Interface SPI
72
```

```
73  //_____________________________________________________________________________________________________________________
74
75  // Para display no monitor serial das configuracoes dos INA226 descomentar o comando abaixo e todos os checkConfig(n)
76  //#include "checkConfig.h"
77  // Configuracao atual:
78  // Mode - "Shunt and Bus, Continuous"
79  // Averages - "64 samples"
80  // BusConversionTime - "1.100ms"
81  // ShuntConversionTime - "1.100ms"
82  // Rshunt = 0.1 ohm
83  // Max excepted current = 0.8 A
84
85  //_____________________________________________________________________________________________________________________
86
87  // eeprom
88  #define i2c_addr_eeprom 0x57                                    // Endereco I2C da 24C32 (interna ao RTC DS3231)
89
90  template <class T> int eeWrite(int ee, const T& value)          // Generic class TypeScript
91  {
92    const byte* p = (const byte*)(const void*)&value;
93    int i;
94    Wire.beginTransmission(i2c_addr_eeprom);
95    Wire.write((int)(ee >> 8)); // MSB
96    Wire.write((int)(ee & 0xFF)); // LSB
97    for (i = 0; i < sizeof(value); i++)
98    Wire.write(*p++);
99    Wire.endTransmission();
100   return i;
101 }
102
103 template <class T> int eeRead(int ee, T& value)                 // Generic class TypeScript
104 {
105   byte* p = (byte*)(void*)&value;
106   int i;
107   Wire.beginTransmission(i2c_addr_eeprom);
108   Wire.write((int)(ee >> 8)); // MSB
```

```
109     Wire.write((int)(ee & 0xFF)); // LSB
110     Wire.endTransmission();
111     Wire.requestFrom(i2c_addr_eeprom,sizeof(value));
112     for (i = 0; i < sizeof(value); i++)
113     if(Wire.available())
114     *p++ = Wire.read();
115     return i;
116   }
117
118   struct eeprom                                          // Estrutura registrada na eeprom
119   {
120     unsigned long valido;                                // Código que confirma eeprom válida
121     float etracer1;                                      // Energia recebida no Tracer 1
122     float etracer2;                                      // Energia recebida no Tracer 2
123     float einversor;                                     // Energia cedida pelo inversor
124     unsigned long inter1;                                // Duração do primeiro ciclo em milissegundos
125     unsigned long inter2;                                // Duração do segundo ciclo em milissegundos
126     unsigned int cod_mon;                                // Código para ativar ou desativar a saída no monitor
127   } eeprom;
128
129   //_____________________________________________________________________________________________________________
130
131   // WiFi
132   const char* ssid = "Pardini";                          // Nome da rede WiFi
133   const char* password = "sitiolacicala";                // Senha da rede WiFi
134
135   //_____________________________________________________________________________________________________________
136
137   // MQTT
138   const char* BROKER_MQTT = "m16.cloudmqtt.com";         // URL do broker MQTT
139   int BROKER_PORT = 17962;                               // Porta do Broker MQTT
140   #define ID_MQTT  "lrffixsc"                            // User ID
141   #define PSW_MQTT "JUp-9gCLKD2X"                        // Password
142   #define tracer1 "comosisfog/tracer1"                   // Topico tracer1
143   #define tracer2 "comosisfog/tracer2"                   // Topico tracer2
144   #define inversor "comosisfog/inversor"                 // Topico inversor
```

```
145  #define etotin "comosisfog/etotin"                          // Topico energia total acumulada
146  #define etotout "comosisfog/etotout"                        // Topico energia total consumida
147  #define aviso1 "comosisfog/aviso1"                          // Topico aviso1
148  #define aviso2 "comosisfog/aviso2"                          // Topico aviso2
149
150  //_____________________________________________________________________________________________________________
151
152  // Instancias
153  WiFiClient wificlient;                                       // Instancia WiFi MQTT
154  WiFiClient wifiTS;                                           // Instancia WiFi ThingSpeak
155  PubSubClient MQTT(BROKER_MQTT,BROKER_PORT,wificlient);       // Instancia MQTT passando o objeto wificlient
156  HardwareSerial SerialXbee(2);                                // Instancia Serial
157  elapsedMillis timeElapsed1;                                  // Instancia elapsedMillis
158  elapsedMillis timeElapsed2;                                  // Instancia elapsedMillis
159  elapsedMillis timeElapsed;                                   // Instancia elapsedMillis
160  elapsedMillis timeElapsed24h;                                // Instancia elapsedMillis
161  INA226 ina1;                                                 // Instancia INA226 Tracer1
162  INA226 ina2;                                                 // Instancia INA226 Tracer2
163  INA226 ina3;                                                 // Instancia INA226 Inversor
164  RTC_DS3231 rtc;                                              // Instancia RTC_DS3231
165
166  //_____________________________________________________________________________________________________________
167
168  // ThingSpeak
169  unsigned long myChannelNumber = 726200;                      // ThingSpeak ID do canal
170  const char * myWriteAPIKey = "MPVSYRDHBVLOCHJ8";             // ThingSpeak chave de escrita
171  String myStatus = "";                                        // ThingSpeak para teste
172
173  //_____________________________________________________________________________________________________________
174
175  // Inicializa o display Oled
176  SSD1306Wire display(0x3C, 5, 4);                             // Para ESP32
177
178  //_____________________________________________________________________________________________________________
179
180  // Variaveis e constantes
```

```
181  // SPI MOSI - 23
182  // SPI MISO - 19
183  // SPI SCK  - 18
184  // SPI SS   - 5
185  #define fotocelula 25                                   // Pino de entrada fotocélula
186  #define inputPin 26                                     // Pino para reinicialização da eeprom
187  #define SDApin 21                                       // I2C SDA
188  #define SCLpin 22                                       // I2C SCL
189  #define RXpin 16                                        // UART RX
190  #define TXpin 17                                        // UART TX
191
192
193  uint8_t noite;
194  bool foinoite;
195
196  char daysOfTheWeek[7][12] = {"Domingo", "Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado"};
197  long lastReconnectAttempt = 0;
198  float delta1,delta2,delta3,etotalin,coef1,vina1,cina1,pina1,vina2,cina2,pina2,vina3,cina3,pina3;
199  float etotalin_24h,etotalin_24h_inicial,etotalout_24h,etotalout_24h_inicial;
200  char v_ina1[6],c_ina1[6],p_ina1[6],e_ina1[6];
201  char v_ina2[6],c_ina2[6],p_ina2[6],e_ina2[6];
202  char v_ina3[6],c_ina3[6],p_ina3[6],e_ina3[6];
203  String tempo,VB,AT,PT,WT,registro,str;
204  int grupo_rf[13][13];                                   // Grupo -> refletores que compõe o grupo
205  int grupo_am[13][3];                                    // Grupo -> ativo(0-inativo,1-ativo),
206                                                          // modo(0-automático(fotocélula),
207                                                          // 1-programado(horario de liga e desliga))
208                                                          // ld (0-desligado, 1igado)
209  float grupo_if[13][2];                                  // Grupo -> liga e desliga programação horaria
210  bool refletor_ligado[13];                               // true -> refletor ligado, false -> refletor desligado
211  char buffer[100];                                       // Buffer
212  int pcf1 = 0x20;                                        // Endereço I2C PCF8574-1
213  int pcf2 = 0x21;                                        // Endereço I2C PCF8574-2
214  unsigned long key = 2863311530;                         // Código de eeprom válida
215  uint8_t j;
216  uint8_t jj;
```

```
217  uint8_t grupo;
218  char par;
219  unsigned int ii = 0;
220  bool ver;
221  bool inversor_ligado;
222  uint8_t hora;
223  uint8_t segundo;
224  bool pula;
225  char car;
226  unsigned int inicio,fim,datseq;
227  File arquivo;
228
229  float vina1_max_dia;                                    // Voltagem máxima INA1 no dia
230  float vina1_min_dia;                                    // Voltagem mínima INA1 no dia
231  float vina2_max_dia;                                    // Voltagem máxima INA2 no dia
232  float vina2_min_dia;                                    // Voltagem mínima INA2 no dia
233  float vina3_max_dia;                                    // Voltagem máxima INA3 no dia
234  float vina3_min_dia;                                    // Voltagem mínima INA3 no dia
235
236  //_____________________________________________________________________________________________________________________
237
238  // Funções
239  void telainicial();
240  void conecta_MQTT(void);
241  boolean reconnect();
242  void configura_INAs(unsigned char ch);
243  void conecta_wifi(unsigned char pr);
244  bool toggle_grupo_refletores(uint8_t grp);
245  void inicializa_grupos_refletores(void);
246  void limpa_buffer_serial(uint8_t cmp);
247  bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde);
248  void liga_desliga_grupos_automaticos(uint8_t ld);
249  void liga_desliga_grupos_programados(void);
250  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano);
251  bool SD_readStringUntil(const char car);
252  bool tem_algo_ligado(void);
```

```
void liga_inversor(void);
void desliga_inversor(void);

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

void setup()
{
  // Definição pinos __________________________________________________________________________________________________
  pinMode(fotocelula, INPUT_PULLUP);                        // Pino a ser acionado pela fotocelula
  pinMode(inputPin, INPUT_PULLUP);                          // Modo do inputPin para eeprom default

  // Inicia monitor serial ____________________________________________________________________________________________
  Serial.begin(115200);
  while (!Serial);

  // Inicia cartão SD _________________________________________________________________________________________________
  if(!SD.begin())
  {
    //Serial.println("Montagem do cartao falhou");
    SerialXbee.print("<Montagem SD falhou>");
    while(1);
  }
  uint8_t cardType = SD.cardType();
  if(cardType == CARD_NONE){
    //Serial.println("Sem cartao inserido");
    SerialXbee.print("<Sem SD inserido>");
    while(1);
  }
  Serial.println("SD inicializado");
  Serial.print("SD tipo: ");
  if(cardType == CARD_MMC) Serial.println("MMC");
  else if(cardType == CARD_SD) Serial.println("SDSC");
  else if(cardType == CARD_SDHC) Serial.println("SDHC");
  else Serial.println("DESCONHECIDO");
```

```
289
290     uint64_t cardSize = SD.cardSize() / (1024 * 1024);
291     Serial.printf("SD tamanho: %lluMB\n", cardSize);
292
293     //*************************************************************************************************************
294     //SD.remove("/dados.txt");                                    // *******Só quando precisa reinicializar /dados.txt******
295     //*************************************************************************************************************
296
297     arquivo = SD.open("/dados.txt", FILE_APPEND);
298     if(!arquivo)
299     {
300       //Serial.print("\nFalha na abertura do arquivo");
301       SerialXbee.print("<Falha no arquivo 1>");
302       while(1);
303     }
304
305     // Inicia Xbee ____________________________________________________________________________________________________
306     SerialXbee.begin(115200, SERIAL_8N1, RXpin, TXpin);       // Inicia serial UART Xbee
307
308     // Inicia I2C _____________________________________________________________________________________________________
309     Wire.begin(SDApin,SCLpin);                                // I2C ESP32 - SDA 21 SCL 22
310
311     // Inicia ThingSpeak ______________________________________________________________________________________________
312     ThingSpeak.begin(wifiTS);
313
314     // Inicia display Oled ____________________________________________________________________________________________
315     display.init();
316
317     // Inicializa os grupos de refletores _____________________________________________________________________________
318     inicializa_grupos_refletores();                           // Inicializa os PCF8574 com as saidas 0 (grupos desligados)
319
320     // Inicializa INAs 226 ____________________________________________________________________________________________
321     configura_INAs(1);                                        // Configura INAs 1 - imprime dados 0 - nao imprime
322
323     // Display da tela inicial no OLED ________________________________________________________________________________
324     telainicial();
```

```
325
326     // No caso de eeprom inválida ou inicialização forçada ____________________________________________________
327     eeRead(0,eeprom);                                       // Le eeprom
328     delay(30);
329
330     if ((eeprom.valido != key) || (digitalRead(inputPin) == LOW)) // Força reinicializacao default
331     {
332       eeprom.etracer1 = 0.0;
333       eeprom.etracer2 = 0.0;
334       eeprom.einversor = 0.0;
335       eeprom.inter1 = 60000;                                // 1 minuto
336       eeprom.inter2 = 300000;                               // 5 minutos
337       eeprom.cod_mon = 1;                                   // Habilita monitor serial
338       eeprom.valido = key;                                  // Valida a eeprom
339
340       eeWrite(0,eeprom);                                    // Grava eeprom
341       delay (30);
342
343       Serial.print("\neeprom default");
344       Serial.println("deixar o pino inputPin flutuante");
345     }
346     else  Serial.print("\neeprom valida");                  // Eeprom valida
347
348     coef1 = eeprom.inter1 / 3600000.0;                      // Para calculo da energia (Wh)
349
350     if (eeprom.cod_mon == 1)
351     {
352       Serial.println("configuracao na eeprom");
353       Serial.println("interval1    " + String(eeprom.inter1));
354       Serial.println("interval2    " + String(eeprom.inter2));
355       Serial.println("cod_monitor  " + String(eeprom.cod_mon));
356       Serial.println("");
357     }
358
359     // Inicia WiFi ____________________________________________________________________________________________
360     conecta_wifi(1);                                        // Conecta WiFi 1 - imprime dados 0 - nao imprime
```

```
361
362     // Imprime no display Oled ____________________________________________________________________________________
363     DateTime now = rtc.now();                                   // Pega data e hora atual
364     // Monta string com data e hora atual
365     tempo = String(now.day()) + '/' + String(now.month()) + '/' + String(now.year()) + ' ' + String(now.hour())
366     + ':' + String(now.minute()) +  ':' + String(now.second());
367     display.clear();
368     display.setTextAlignment(TEXT_ALIGN_LEFT);
369     display.setFont(ArialMT_Plain_10);
370     display.drawString(1, 5, tempo);
371     display.drawString(1, 15, "configuracao da eeprom");
372     display.drawString(1, 25, "interval1       " + String(eeprom.inter1));
373     display.drawString(1, 35, "interval2       " + String(eeprom.inter2));
374     display.drawString(1, 45, "cod_monitor  " + String(eeprom.cod_mon));
375     display.display();
376     delay(5000);
377
378     // Inicializações ____________________________________________________________________________________________
379     pula = false;
380     foinoite = false;
381     inversor_ligado = false;
382     timeElapsed1 = 0;                                           // Inicializa contagem
383     timeElapsed2 = 0;                                           // Inicializa contagem
384     etotalin = 0.0;
385
386     // Inicializa
387     vina1_max_dia = 0.0;                                        // Voltagem máxima INA1 no dia
388     vina1_min_dia = 30.0;                                       // Voltagem mínima INA1 no dia
389     vina2_max_dia = 0.0;                                        // Voltagem máxima INA2 no dia
390     vina2_min_dia = 30.0;                                       // Voltagem mínima INA2 no dia
391     vina3_max_dia = 0.0;                                        // Voltagem máxima INA3 no dia
392     vina3_min_dia = 30.0;                                       // Voltagem mínima INA3 no dia
393
394     if ((digitalRead(fotocelula) == LOW)) noite = 1;            // Inicializa a fotocelula
395     else noite = 0;
396
```

```
}

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

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

  //_____________________________________________________________________________________________________________________

  // Verifica se tem comunicacao vindo de P2 para solicitação, configuração ou comando
  if (SerialXbee.available() > 0)
  {
    // Verifica se P2 enviou alguma coisa: comando, coniguração p1, configuração dos grupos de refletores, configuração
    // de seus modos ou solicitação de dados
    // Indicadores válidos:
    // '!' - Comando toggle dos grupos de refletores - formato - !(g) - onde g é o número do grupo. g = 0 => tudo
    // '$' - Solicitação de dados - formato - $(i) - onde i=1 data-hora, i=2 dados INAs e energias, i=3 status grupos
    //       e refletores
    // '@' - Solicitação de dados - formato - @(di,df) - onde di - data de início em dias desde 01/01/2019, df - data
    //       de fim em dias desde 01/01/2019. Ex: @(283,289) (equivale a @(10/10/2019,16/10/2019) vai ler o SD e enviar
    //       os dados relativos a esse período
    // '#' - Configuração de ciclos em p1 - formato - #n(v) - onde n=1 interval1_p1, n=2 interval2_p1,
    //       n=3 cod_monitor_p1, v = 0 ou 1 ou valor em minutos
    // '&' - Configuração dos grupos de refletores - formato - &n(r,...,r) - onde n é o número do grupo
    //       e r o número do refletor. grupo[x][0] vai conter o número de refletores registrados no grupo.
    //       O grupo[0] tem que ser definido com todos os refletores **************************************************
    // '*' - Configuração dos horarios liga/desliga dos grupos e modo de operação de cada grupo
    //       Modo de operação: ativo -> sim(1)/não(0), automatico -> fotocelula, programado -> hora liga/desliga,
    //       manual -> comando de toggle via p2,PC ou celular - formato - *n(a,b,c,d) - onde n é o número do grupo
    //       a=0 -> inativo, a=1 -> ativo, b=0 -> automático(fotocelula), b=1 ->programado(c e d só existem neste caso),
    //       c -> hora liga (formato: h.m h->horas, m->minutos/100 guardado como float h+m), d -> hora desliga (idem)

```

```
433       // Monta string com data e hora atual
434       DateTime now = rtc.now();                                   // Pega data e hora atual
435       tempo = String(now.day()) + '/' + String(now.month()) + '/' + String(now.year()) + ' ' + String(now.hour())
436       + ':' + String(now.minute()) +  ':' + String(now.second());
437
438       while (SerialXbee.available() > 0)                          // Enquanto tiver dados no buffer
439       {
440         par = SerialXbee.read();
441         switch (par)
442         {
443           case '!':                                               // Pedido para toggle de um grupo
444             if (!toggle_grupo_refletores(SerialXbee.parseInt())) Serial.print("\nToggle nao executado - grupo inativo");
445             else Serial.print("\nToggle executado - grupo ativo");
446             break;
447           case '$':                                               // Solicitação de data/hora, dados INAs
448             switch (SerialXbee.parseInt())
449             {
450               case 1:                                             // Solicitação de data/hora
451                 SerialXbee.print(tempo.c_str());
452                 break;
453               case 2:                                             // Solicitação de dados INAs
454                 SerialXbee.print("v1("); SerialXbee.print(vina1,2); SerialXbee.print("),");
455                 SerialXbee.print("c1("); SerialXbee.print(cina1,2); SerialXbee.print("),");
456                 SerialXbee.print("p1("); SerialXbee.print(pina1,2); SerialXbee.print("),");
457                 SerialXbee.print("v2("); SerialXbee.print(vina2,2); SerialXbee.print("),");
458                 SerialXbee.print("c2("); SerialXbee.print(cina2,2); SerialXbee.print("),");
459                 SerialXbee.print("p2("); SerialXbee.print(pina2,2); SerialXbee.print("),");
460                 SerialXbee.print("v3("); SerialXbee.print(vina3,2); SerialXbee.print("),");
461                 SerialXbee.print("c3("); SerialXbee.print(cina3,2); SerialXbee.print("),");
462                 SerialXbee.print("p3("); SerialXbee.print(pina3,2); SerialXbee.print("),");
463                 SerialXbee.print("ei("); SerialXbee.print(etotalin,2); SerialXbee.print("),");
464                 SerialXbee.print("eo("); SerialXbee.print(eeprom.einversor,2); SerialXbee.print(")");
465                 break;
466               case 3:                                             // Solicitação do status de grupos e refletores
467                 for (j=1;j<=12;j++)                               // Composição dos grupos
468                 {
```

```
469                   SerialXbee.print("Grupo "); SerialXbee.print(j); SerialXbee.print(" => ");
470                   for (jj=1;jj<=grupo_rf[j][0];jj++)
471                   {
472                     SerialXbee.print(grupo_rf[j][jj]);
473                     if (jj < grupo_rf[j][0]) SerialXbee.print(", ");
474                     if (jj == grupo_rf[j][0]) SerialXbee.print("#");
475                   }
476                 }
477                 for (j=1;j<=12;j++)                              // Status dos refletores
478                 {
479                   SerialXbee.print("Refletor "); SerialXbee.print(j); SerialXbee.print(" => ");
480                   if (refletor_ligado[j]) SerialXbee.print("ligado");
481                   else SerialXbee.print("desligado");
482                   if (j == 12) SerialXbee.print("#");
483                 }
484                 break;
485               }
486             break;
487             case '#':
488               eeRead(0,eeprom);                                  // Le eeprom
489               delay(30);
490               switch (SerialXbee.parseInt())                     // Configuraçãoi de ciclos
491               {
492                 case 1:                                          // É interval1_p1
493                   eeprom.inter1 = SerialXbee.parseInt();
494                   break;
495                 case 2:                                          // É interval2_p1
496                   eeprom.inter2 = SerialXbee.parseInt();
497                   break;
498                 case 3:                                          // É cod_monitor_p1
499                   eeprom.cod_mon = SerialXbee.parseInt();
500                   break;
501               }
502               eeWrite(0,eeprom);                                 // Grava eeprom
503               delay (30);
504               break;
```

```
505           case '&':                                          // Configuração da composição dos grupos
506             // Nota: o grupo 1 contém todos os refletores para comando total
507             grupo = SerialXbee.parseInt();
508             j = 1;
509             while (SerialXbee.available() > 0)               // Enquanto tiver dados no buffer
510             {
511               grupo_rf[grupo][j] = SerialXbee.parseInt();
512               j++;
513             }
514             grupo_rf[grupo][0] = j;
515             break;
516           case '*':                                          // Configuração do modo dos grupos
517             grupo = SerialXbee.parseInt();
518             grupo_am[grupo][0] = SerialXbee.parseInt();
519             grupo_am[grupo][1] = SerialXbee.parseInt();
520             if (grupo_am[grupo][1] == 1)                     // É programado
521             {
522               grupo_if[grupo][0] = SerialXbee.parseFloat();  // Liga h.m (h - hora(0-24), m - minutos(0-60))
523               grupo_if[grupo][1] = SerialXbee.parseFloat();  // Desliga h.m (h - hora(0-24), m - minutos(0-60))
524             }
525             break;
526           case '@':                                          // Solicitação de dados gravado0s no SD
527             inicio = SerialXbee.parseInt();
528             fim = SerialXbee.parseInt();
529
530             if (inicio > fim)
531             {
532               Serial.println("\nData de inicio maior que data de fim");
533               break;
534             }
535
536             arquivo.close();
537             arquivo = SD.open("/dados.txt");
538             if(!arquivo)
539             {
540               //Serial.print("\nFalha na abertura do arquivo");
```

```
541                 SerialXbee.print("<Falha no arquivo 2>");
542                 while(1);
543               }
544               else
545               {
546                 if (!SD_readStringUntil('<'))                    // Lê o primeiro registro
547                 {
548                   //Serial.print("\nRegistro nao encontrado no SD - erro 1");
549                   SerialXbee.print("<Falha registro 1>");
550                   while(1);
551                 }
552                 // Encontrou o primeiro registro
553                 datseq = arquivo.parseInt();                     // Vamos ler a datseq do primeiro registro
554
555                 while (datseq < fim)                             // Vamos executar enquanto datseq < fim
556                 {
557                   while (datseq < inicio)                        // Se o datseq é menor que o inicio,
558                   {
559                     if (!SD_readStringUntil('<'))                // vamos ler o arquivo até o próximo registro
560                     {
561                       //Serial.print("\nRegistro nao encontrado no SD - erro 2");
562                       SerialXbee.print("<Falha registro 2>");
563                       break;
564                     }
565                     datseq = arquivo.parseInt();                 // até acharmos um registro com datseq >= inicio
566                   }
567
568                   // Achamos um, vamos imprimir
569                   if (!SD_readStringUntil('>'))
570                   {
571                     //Serial.print("\nFim de registro nao encontrado no SD - erro 3");
572                     SerialXbee.print("<Falha registro 3>");
573                     break;
574                   }
575                   str = '<' + String(datseq)+ str;
576                   Serial.print("\n" + str);
```

```
577                 if (!SD_readStringUntil('<'))                  // ler o arquivo até o próximo registro
578                 {
579                   //Serial.print("\nRegistro nao encontrado no SD - erro 4");
580                   SerialXbee.print("<Falha registro 4>");
581                   break;
582                 }
583                 datseq = arquivo.parseInt();
584               }
585               if (datseq == fim)                               // Achamos o último, vamos enviar para Parte 2
586               {
587                 if (!SD_readStringUntil('>'))
588                 {
589                   //Serial.print("\nFim de registro nao encontrado no SD - erro 5");
590                   SerialXbee.print("<Falha registro 5>");
591                   while(1);
592                 }
593                 str = '<' + String(datseq) + str;
594                 Serial.print("\n" + str);
595               }
596               else;
597               arquivo.close();
598             }
599             break;
600           }
601         }
602
603         //_____________________________________________________________________________________________________________
604
605         // Avisa no display Oled
606         display.clear();
607         display.setTextAlignment(TEXT_ALIGN_LEFT);
608         display.setFont(ArialMT_Plain_10);
609         display.drawString(1, 5, tempo);
610         display.drawString(1, 15, "nova configuracao gravada");
611         display.drawString(1, 25, "interval1       " + String(eeprom.inter1));
612         display.drawString(1, 35, "interval2       " + String(eeprom.inter2));
```

```
613       display.drawString(1, 45, "cod_monitor  " + String(eeprom.cod_mon));
614       display.display();
615       delay(5000);
616
617       // Avisa no MQTT
618       MQTT.publish(aviso2,"eeprom configurada");
619
620       // Avisa no monitor serial
621       if (eeprom.cod_mon == 1)
622       {
623         Serial.println("");
624         Serial.println("nova configuracao gravada na eeprom");
625         Serial.println("interval1    " + String(eeprom.inter1));
626         Serial.println("interval2    " + String(eeprom.inter2));
627         Serial.println("cod_monitor  " + String(eeprom.cod_mon));
628         Serial.println("");
629       }
630     }
631
632     //_____________________________________________________________________________________________________________
633
634     DateTime now = rtc.now();                                   // Pega data e hora atual
635     // Monta string com data e hora atual
636     hora = now.hour();
637     tempo = String(now.day()) + '/' + String(now.month()) + '/' + String(now.year()) + ' ' + String(hora)
638     + ':' + String(now.minute()) +  ':' + String(now.second());
639
640     if ((hora == 5) && !pula)                                   // Inicia periodo de 24h às 05h, executa uma única vez
641     {
642       // Vamos iniciar a contabilizar quanta energia foi acumulada e quanta foi consumida no periodo de 24h (dia/noite)
643       timeElapsed24h = 0;
644       etotalin_24h_inicial = etotalin;                          // Energia total de entrada até agora (05h)
645       etotalout_24h_inicial = eeprom.einversor;                 // Energia total de saída até agora (05h)
646       pula = true;
647     }
648
```

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

// **************************************************** Grava SD******************************************************

  if (timeElapsed24h >= 86400000)                            // Executa a cada 24h começando às 05:00
  {
    etotalin_24h = etotalin - etotalin_24h_inicial;          // Energia total acumulada de entrada em 24h
    etotalout_24h = eeprom.einversor - etotalout_24h_inicial; // Energia total acumulada de saída em 24h
    etotalin_24h_inicial = etotalin;                         // Energia total de entrada até agora (05h)
    etotalout_24h_inicial = eeprom.einversor;                // Energia total de saída até agora (05h)
    timeElapsed24h = 0;

    // Vamos gravar o SD:
    // <datseq,dia,mes,ano,vina1_max_dia,vina1_min_dia,vina2_max_dia,vina2_min_dia,vina3_max_dia,
    // vina3_min_dia,etotalin_24h,etotalout_24h>

    DateTime now = rtc.now();

    registro = '<'+String(dias_desde_01_01_2019(now.day(),now.month(),now.year()))+','+String(now.day())+
    ','+String(now.month())+','+String(now.year())+','+String(vina1_max_dia)+','+String(vina1_min_dia)+
    ','+String(vina2_max_dia)+','+String(vina2_min_dia)+','+String(vina3_max_dia)+','+String(vina3_min_dia)+
    ','+String(etotalin_24h)+','+String(etotalout_24h)+'>';

    arquivo.close();
    arquivo = SD.open("/dados.txt", FILE_APPEND);
    if(!arquivo)
    {
      //erial.print("\nFalha na abertura do arquivo");
      SerialXbee.print("<Falha no arquivo 3>");
      while(1);
    }
    arquivo.print(registro);
    Serial.print("\n" + registro);

```

```
      // Reinicializa
      vina1_max_dia = 0.0;                                    // Voltagem máxima INA1 no dia
      vina1_min_dia = 30.0;                                   // Voltagem mínima INA1 no dia
      vina2_max_dia = 0.0;                                    // Voltagem máxima INA2 no dia
      vina2_min_dia = 30.0;                                   // Voltagem mínima INA2 no dia
      vina3_max_dia = 0.0;                                    // Voltagem máxima INA3 no dia
      vina3_min_dia = 30.0;                                   // Voltagem mínima INA3 no dia

    }

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

    if (timeElapsed1 >= eeprom.inter1)                        // Executa a cada ciclo de duracao inter1
    {                                                         // Default 1 minuto
      // Lê os sensores INA226
      vina1 = ina1.readBusVoltage();                          // Voltagem INA1
      cina1 = ina1.readShuntCurrent();                        // Corrente INA1
      pina1 = ina1.readBusPower();                            // Potência INA1
      delta1 = pina1 * coef1;                                 // Delta energia
      vina2 = ina2.readBusVoltage();                          // Voltagem INA2
      cina2 = ina2.readShuntCurrent();                        // Corrente INA2
      pina2 = ina2.readBusPower();                            // Potência INA2
      delta2 = pina2 * coef1;                                 // Delta energia
      vina3 = ina3.readBusVoltage();                          // Voltagem INA3
      cina3 = ina3.readShuntCurrent();                        // Corrente INA3
      pina3 = ina3.readBusPower();                            // Potência INA3
      delta3 = pina3 * coef1;                                 // Delta energia
      eeprom.etracer1 = eeprom.etracer1 + delta1;             // Integra energia
      eeprom.etracer2 = eeprom.etracer2 + delta2;             // Integra energia
      eeprom.einversor = eeprom.einversor + delta3;           // Integra energia
      etotalin = eeprom.etracer1 + eeprom.etracer2;           // Integra energia total acumulada

      if (eeprom.cod_mon == 1)
      {
```

```
721         // Imprime data e hora atual
722         Serial.println("");
723         Serial.println(tempo);
724         Serial.println("");
725         // Imprime leituras relativas ao Tracer1
726         Serial.println("INA226-1 Tracer1");
727         Serial.println("-----------------------------------------------");
728         Serial.print("Bus voltage:   ");
729         Serial.print(vina1, 2);
730         Serial.println(" V");
731         Serial.print("Shunt current: ");
732         Serial.print(cina1, 2);
733         Serial.println(" A");
734         Serial.print("Bus power:     ");
735         Serial.print(pina1, 2);
736         Serial.println(" W");
737         Serial.print("Bus energy:    ");
738         Serial.print(delta1, 2);
739         Serial.println(" Wh");
740         Serial.println("-----------------------------------------------");
741         Serial.println("");
742         // Imprime leituras relativas ao Tracer2
743         Serial.println("INA226-2 Tracer2");
744         Serial.println("-----------------------------------------------");
745         Serial.print("Bus voltage:   ");
746         Serial.print(vina2, 2);
747         Serial.println(" V");
748         Serial.print("Shunt current: ");
749         Serial.print(cina2, 2);
750         Serial.println(" A");
751         Serial.print("Bus power:     ");
752         Serial.print(pina2, 2);
753         Serial.println(" W");
754         Serial.print("Bus energy:    ");
755         Serial.print(delta2, 2);
756         Serial.println(" Wh");
```

```
757         Serial.println("-----------------------------------------------");
758         Serial.println("");
759         // Imprime leituras relativas ao Inversor
760         Serial.println("INA226-3 Inversor");
761         Serial.println("-----------------------------------------------");
762         Serial.print("Bus voltage:   ");
763         Serial.print(vina3, 2);
764         Serial.println(" V");
765         Serial.print("Shunt current: ");
766         Serial.print(cina3, 2);
767         Serial.println(" A");
768         Serial.print("Bus power:     ");
769         Serial.print(pina3, 2);
770         Serial.println(" W");
771         Serial.print("Bus energy:    ");
772         Serial.print(delta3, 2);
773         Serial.println(" Wh");
774         Serial.println("-----------------------------------------------");
775       }
776
777       //_____________________________________________________________________________________________________________
778
779       // Para display na OLED
780       dtostrf(vina1, 5, 2, v_ina1);
781       dtostrf(cina1, 5, 2, c_ina1);
782       dtostrf(pina1, 5, 2, p_ina1);
783       dtostrf(eeprom.etracer1, 5, 2, e_ina1);
784
785       dtostrf(vina2, 5, 2, v_ina2);
786       dtostrf(cina2, 5, 2, c_ina2);
787       dtostrf(pina2, 5, 2, p_ina2);
788       dtostrf(eeprom.etracer2, 5, 2, e_ina2);
789
790       dtostrf(vina3, 5, 2, v_ina3);
791       dtostrf(cina3, 5, 2, c_ina3);
792       dtostrf(pina3, 5, 2, p_ina3);
```

```
793       dtostrf(eeprom.einversor, 5, 2, e_ina3);
794
795       ii++;
796       if (ii > 3) ii = 1;
797       switch (ii)
798       {
799         case 1:
800         VB = "V-Banco     " + String(v_ina1);
801         AT = "A-Tracer1   " + String(c_ina1);
802         PT = "P-Tracer1   " + String(p_ina1);
803         WT = "Wh-Tracer1 " + String(e_ina1);
804         break;
805         case 2:
806         VB = "V-Banco     " + String(v_ina2);
807         AT = "A-Tracer2   " + String(c_ina2);
808         PT = "P-Tracer2   " + String(p_ina2);
809         WT = "Wh-Tracer2 " + String(e_ina2);
810         break;
811         case 3:
812         VB = "V-Banco      " + String(v_ina3);
813         AT = "A-Inversor   " + String(c_ina3);
814         PT = "P-Inversor   " + String(p_ina3);
815         WT = "Wh-Inversor " + String(e_ina3);
816         break;
817       }
818
819       display.clear();
820       display.setTextAlignment(TEXT_ALIGN_LEFT);
821       display.setFont(ArialMT_Plain_10);
822       display.drawString(1, 5, tempo);
823       display.drawString(1, 15, VB);
824       display.drawString(1, 25, AT);
825       display.drawString(1, 35, PT);
826       display.drawString(1, 45, WT);
827       display.display();
828
```

```
    //____________________________________________________________________________________________________________________

    // Para verificar se a fotocelula está ativa e quais grupos de refletores devem ser ligados

    noite = digitalRead(fotocelula);                          // Lê o pino da fotocélula

    if ((noite == 1) && (!foinoite))                          // Iniciou a noite, liga
    {
      // Vamos ligar os grupos que tem o modo automático
      liga_desliga_grupos_automaticos(1);
      foinoite = true;
    }
    else if ((noite == 0) && foinoite)                        // Acabou a noite, desliga
    {
      // Vamos desligar os grupos que tem o modo automático
      liga_desliga_grupos_automaticos(0);
      foinoite = false;
    }
    else;                                                     // Continua noite ou continua dia

    //____________________________________________________________________________________________________________________

    // Para verificar horario de ligar ou desligar os grupos de refletores programados

    liga_desliga_grupos_programados();

    //____________________________________________________________________________________________________________________

      timeElapsed1 = 0;                                       // Reset do contador
    }

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

    if (timeElapsed2 >= eeprom.inter2)                        // Executa quando completa um ciclo de duracao inter2
```

```
865     {                                                       // Default 5 minutos
866       // Vamos pegar máximos e mínimos da voltagem das INAs (no dia)
867       if (vina1 > vina1_max_dia) vina1_max_dia = vina1;
868       else if (vina1 < vina1_min_dia) vina1_min_dia = vina1;
869       else;
870       if (vina2 > vina2_max_dia) vina2_max_dia = vina2;
871       else if (vina2 < vina2_min_dia) vina2_min_dia = vina2;
872       else;
873       if (vina3 > vina3_max_dia) vina3_max_dia = vina3;
874       else if (vina3 < vina3_min_dia) vina3_min_dia = vina3;
875       else;
876
877       if (eeprom.cod_mon == 1)
878       {
879         Serial.println("-------------------------------------------------");
880         Serial.print("Energia Tracer1: ");
881         Serial.print(eeprom.etracer1, 2);
882         Serial.println(" Wh");
883         Serial.println("-------------------------------------------------");
884         Serial.println("");
885
886         Serial.println("-------------------------------------------------");
887         Serial.print("Energia Tracer2: ");
888         Serial.print(eeprom.etracer2, 2);
889         Serial.println(" Wh");
890         Serial.println("-------------------------------------------------");
891         Serial.println("");
892
893         Serial.println("-------------------------------------------------");
894         Serial.print("Energia Inversor: ");
895         Serial.print(eeprom.einversor, 2);
896         Serial.println(" Wh");
897         Serial.println("-------------------------------------------------");
898         Serial.println("");
899
900         Serial.println("-------------------------------------------------");
```

```
901           Serial.print("Energia acumulada: ");
902           Serial.print(etotalin, 2);
903           Serial.println(" Wh");
904           Serial.println("-----------------------------------------------");
905           Serial.println("");
906
907           Serial.println("-----------------------------------------------");
908           Serial.print("Energia consumida: ");
909           Serial.print(eeprom.einversor, 2);
910           Serial.println(" Wh");
911           Serial.println("-----------------------------------------------");
912           Serial.println("");
913         }
914
915         //_____________________________________________________________________________________________________________
916
917         // Documentos para Json
918         StaticJsonDocument<100> doc1;
919         StaticJsonDocument<100> doc2;
920         StaticJsonDocument<100> doc3;
921         StaticJsonDocument<100> doc4;
922         StaticJsonDocument<100> doc5;
923
924         doc1["data"] = tempo;
925         JsonArray ina1 = doc1.createNestedArray("tracer1");
926         ina1.add(String(vina1, 2));
927         ina1.add(String(cina1, 2));
928         ina1.add(String(eeprom.etracer1, 2));
929
930         serializeJson(doc1, buffer);
931
932         if (eeprom.cod_mon == 1)
933         {
934           Serial.println("");
935           Serial.println(buffer);
936         }
```

```
937
938       MQTT.publish(tracer1, buffer);                          // Publica MQTT tracer1
939
940       doc2["data"] = tempo;
941       JsonArray ina2 = doc2.createNestedArray("tracer2");
942       ina2.add(String(vina2, 2));
943       ina2.add(String(cina2, 2));
944       ina2.add(String(eeprom.etracer2, 2));
945
946       serializeJson(doc2, buffer);
947
948       if (eeprom.cod_mon == 1)
949       {
950         Serial.println("");
951         Serial.println(buffer);
952       }
953
954       MQTT.publish(tracer2, buffer);                          // Publica MQTT tracer2
955
956       doc3["data"] = tempo;
957       JsonArray ina3 = doc3.createNestedArray("inversor");
958       ina3.add(String(vina3, 2));
959       ina3.add(String(cina3, 2));
960       ina3.add(String(eeprom.einversor, 2));
961
962       serializeJson(doc3, buffer);
963
964       if (eeprom.cod_mon == 1)
965       {
966         Serial.println("");
967         Serial.println(buffer);
968       }
969
970       MQTT.publish(inversor, buffer);                         // Publica MQTT inversor
971
972       doc4["data"] = tempo;
```

```
973       doc4["energia_total_in"] = String(etotalin, 2);
974       serializeJson(doc4, buffer);
975
976       if (eeprom.cod_mon == 1)
977       {
978         Serial.println("");
979         Serial.println(buffer);
980       }
981
982       MQTT.publish(etotin, buffer);                          // Publica MQTT energia acumulada
983
984       doc5["data"] = tempo;
985       doc5["energia_total_out"] = String(eeprom.einversor, 2);
986       serializeJson(doc5, buffer);
987
988       if (eeprom.cod_mon == 1)
989       {
990         Serial.println("");
991         Serial.println(buffer);
992       }
993
994       MQTT.publish(etotout, buffer);                         // Publica MQTT energia
          consumida
995
996       //_____________________________________________________________________________________________________
997
998       // Define os valores para os campos do canal ThingSpeak
999       ThingSpeak.setField(1, vina1);
1000      ThingSpeak.setField(2, cina1);
1001      ThingSpeak.setField(3, vina2);
1002      ThingSpeak.setField(4, cina2);
1003      ThingSpeak.setField(5, vina3);
1004      ThingSpeak.setField(6, cina3);
1005      ThingSpeak.setField(7, etotalin);
1006      ThingSpeak.setField(8, eeprom.einversor);
1007
```

```
1008        // Define o status para o canal do ThingSpeak
1009        if(cina1 > cina2)
1010        {
1011          myStatus = String("corrente de carga tracer1 > tracer2");
1012        }
1013        else if(cina1 < cina2)
1014        {
1015          myStatus = String("corrente de carga tracer2 > tracer1");
1016        }
1017        else if(eeprom.einversor > etotalin)
1018        {
1019          myStatus = String("energia consumida maior que energia acumulada");
1020        }
1021        else {}
1022
1023        ThingSpeak.setStatus(myStatus);                          // Grava o status
1024
1025        // Envia dados para o canal do ThingSpeak
1026        int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
1027        if(x == 200)
1028        {
1029          if (eeprom.cod_mon == 1) Serial.print("\nCanal atualizado com sucesso");
1030        }
1031        else
1032        {
1033          if (eeprom.cod_mon == 1)
1034          {
1035            Serial.println("\nProblema fazendo o update do canal. Codigo de erro HTTP " + String(x));
1036          }
1037        }
1038
1039        //_____________________________________________________________________________________________________________
1040
1041        // Grava eeprom
1042        eeWrite(0,eeprom);
1043        delay (30);
```

```

    if (eeprom.cod_mon == 1)
    {
      Serial.print("energia tracer1 ");
      Serial.print(eeprom.etracer1, 2);
      Serial.println(" Wh");
      Serial.print("energia tracer2 ");
      Serial.print(eeprom.etracer2, 2);
      Serial.println(" Wh");
      Serial.print("energia inversor ");
      Serial.print(eeprom.einversor, 2);
      Serial.println(" Wh");
      Serial.println("");
    }

    timeElapsed2 = 0;                                       // Reset do contador
  }

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

}

//______________________________________________________________________________________________________________________

void telainicial()                                          // Tela inicial do OLED
{
  display.clear();
  display.setTextAlignment(TEXT_ALIGN_CENTER);
  display.setFont(ArialMT_Plain_16);
  display.drawString(63, 5, "CoMoSisFog V1.0");
  display.drawString(63, 25, "ESP32");
  display.drawString(63, 45, "Filippo Pardini");
  display.display();
}
```

```
1080
1081  //_____________________________________________________________________________________________________________
1082
1083  void conecta_MQTT(void)
1084  {
1085    if (!MQTT.connected())
1086    {
1087      long now = millis();
1088      if (now - lastReconnectAttempt > 5000)
1089      {
1090        lastReconnectAttempt = now;
1091        if (reconnect())                                    // Tentativa de reconexao
1092        {
1093          if (eeprom.cod_mon == 1)
1094          {
1095            Serial.println("");
1096            Serial.println("Broker MQTT conectado");
1097            Serial.println("");
1098          }
1099          lastReconnectAttempt = 0;
1100        }
1101      }
1102    }
1103  }
1104
1105  //_____________________________________________________________________________________________________________
1106
1107  boolean reconnect()
1108  {
1109    if (MQTT.connect("quiosque",ID_MQTT,PSW_MQTT))
1110    {
1111      MQTT.publish(aviso1,"conectado");                     // Quando reconectado, publica aviso
1112    }
1113    return MQTT.connected();
1114  }
1115
```

```
1116  //_____________________________________________________________________________________________________________
1117
1118  void configura_INAs(unsigned char ch)
1119  {
1120    // Inicia INA1 endereco 0x40 - Tracer 1 - shunt 30A - 75mV
1121    ina1.begin(0x40);
1122    // Configura
1123    ina1.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
1124    INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
1125    // Calibra INA226 - shunt = 0.0025 ohm, corrente maxima esperada = 20A
1126    ina1.calibrate(0.0025, 20);
1127
1128    // Inicia INA2 endereco 0x41 - Tracer 2 - shunt 30A - 75mV
1129    ina2.begin(0x41);
1130    // Configura
1131    ina2.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
1132    INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
1133    // Calibra INA226 - shunt = 0.0025 ohm, corrente maxima esperada = 20A
1134    ina2.calibrate(0.0025, 20);
1135
1136    // Inicia INA3 endereco 0x44 - Inversor - shunt 50A - 75mV
1137    ina3.begin(0x44);
1138    // Configura
1139    ina3.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
1140    INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
1141    // Calibra INA226 - shunt = 0.0015 ohm, corrente maxima esperada = 30A
1142    ina3.calibrate(0.0015, 30);
1143
1144  /*
1145    if (ch == 1)
1146    {
1147      // Display das configuracoes
1148      checkConfig(1);
1149      checkConfig(2);
1150      checkConfig(3);
1151    }
```

```
1152   */
1153   }
1154
1155   //_____________________________________________________________________________________________________________
1156
1157   void conecta_wifi(unsigned char pr)
1158   {
1159     WiFi.begin(ssid, password);                                  // Inicia WiFi
1160     while (WiFi.status() != WL_CONNECTED)
1161     {
1162       delay(500);
1163       if (eeprom.cod_mon == 1)
1164       {
1165         Serial.println("Conectando WiFi..");
1166       }
1167     }
1168     if (eeprom.cod_mon == 1)
1169     {
1170       Serial.print("Connectado na rede ");
1171       Serial.println(ssid);
1172     }
1173     if (pr == 1)
1174     {
1175       IPAddress ip = WiFi.localIP();                             // Imprime o IP
1176       if (eeprom.cod_mon == 1)
1177       {
1178         Serial.print("IP: ");
1179         Serial.println(ip);
1180       }
1181
1182       long rssi = WiFi.RSSI();                                   // Imprime o nível do sinal
1183       if (eeprom.cod_mon == 1)
1184       {
1185         Serial.print("Nivel do sinal (RSSI): ");
1186         Serial.println(rssi);
1187       }
```

```
1188      }
1189    }
1190
1191    //_____________________________________________________________________________________________________________
1192
1193    bool toggle_grupo_refletores(uint8_t grp)
1194    {
1195      // Faz o toggle (inversão) de um grupo de refletores: se estava ligado -> desliga e se estava desligado -> liga
1196      // Nota: o grupo 0 por definição contém todos os refletores, portanto liga/desliga todos
1197
1198      int adr,ref,rf,m;
1199      byte atual,tmp;
1200
1201      // Vamos verificar se é um grupo ativo
1202      if (grupo_am[grp][1] != 1) return false;                    // grupo inativo, retorna
1203
1204      // É ativo, faz o toggle dos refletores do grupo
1205      for (m=1;m<=grupo_rf[grp][0];m++)                           // Varre os refletores do grupo
1206      {
1207        rf = grupo_rf[grp][m];                                    // refletor (entre 1 e 12)
1208        if (rf <= 8) {adr = pcf1; ref = rf;}                      // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1209        else {adr = pcf2; ref = rf - 8;}                          // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1210
1211        Wire.requestFrom(adr,1);                                  // Solicita 1 byte ao Pcf correto
1212        while (Wire.available() == 0);                            // Aguarda estar no buffer
1213        atual = Wire.read();                                      // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1214                                                                  // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1215                                                                  // -> refletor 12
1216
1217        tmp = atual;                                              // Temporario
1218        tmp <<= (8 - ref);                                        // Pega o bit do refletor
1219        tmp >>= 7;                                                // Pega o bit do refletor
1220        if (tmp == 0)                                             // Se refletor desligado -> liga
1221        {
1222          if (!inversor_ligado) liga_inversor();                  // Se inversor desligado, liga inversor
1223          atual |= 1 << (ref - 1);                                // Toggle do refletor rf
```

```
1224          refletor_ligado[rf] = true;                          // Marca como refletor ligado
1225        }
1226        else                                                   // Se refletor ligado -> desliga
1227        {
1228          atual &= ~(1 << (ref - 1));                          // Toggle do refletor rf
1229          refletor_ligado[rf] = false;                         // Marca como refletor desligado
1230          if (!tem_algo_ligado()) desliga_inversor();          // Se não tem nada ligado, desliga inversor
1231        }
1232
1233        // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1234        Wire.beginTransmission(adr);                           // Inicia transmissão I2C para o Pcf correto
1235        Wire.write(atual);                                     // Novo byte de acionamento dos refletores do grupo
1236        Wire.endTransmission();                                // Encerra transmissão I2C
1237      }
1238      if (grupo_am[grp][2] == 0) grupo_am[grp][2] = 1;         // Marca grupo como ligado
1239      else grupo_am[grp][2] = 0;                               // Marca grupo como desligado
1240      return true;
1241    }
1242
1243    //_____________________________________________________________________________________________________________________
1244
1245    void inicializa_grupos_refletores(void)
1246    {
1247      // Inicializa todos os grupos de refletores e refletores como desligados
1248      uint8_t m;
1249
1250      Wire.beginTransmission(pcf1);
1251      Wire.write(0);
1252      Wire.endTransmission();
1253      Wire.beginTransmission(pcf2);
1254      Wire.write(0);
1255      Wire.endTransmission();
1256
1257      for (m=1;m<=12;m++) refletor_ligado[m] = false;
1258    }
1259
```

```
1260  //_____________________________________________________________________________________________________________________
1261
1262
1263  void limpa_buffer_serial(uint8_t cmp)
1264  {
1265    uint8_t i = 1;
1266
1267    while (i <= cmp)
1268    {
1269      SerialXbee.read();
1270      i++;
1271    }
1272    delay(1000);
1273  }
1274
1275  //_____________________________________________________________________________________________________________________
1276
1277  bool aguarda_serial(uint8_t ser,unsigned long intervalo,uint8_t ncar,uint8_t deonde)
1278  {
1279    timeElapsed = 0;
1280    while(timeElapsed < intervalo)                    // Executa enquanto não passar o intervalo
1281    {
1282      if (ser == 1)
1283      {
1284        if (Serial1.available() < ncar);
1285        else return true;
1286      }
1287      else if (ser == 2)
1288      {
1289        if (Serial2.available() < ncar);
1290        else return true;
1291      }
1292      else
1293      {
1294        Serial.print("\nSerial");
1295        Serial.print(ser);
```

```
1296          Serial.print(" invalida");
1297          Serial.print(" chamada de ");
1298          Serial.print(deonde);
1299          return false;
1300        }
1301      }
1302      Serial.print("\nTime Out esperando Serial");
1303      Serial.print(ser);
1304      Serial.print(" chamada de ");
1305      Serial.print(deonde);
1306      return false;
1307    }
1308
1309    //_____________________________________________________________________________________________________________
1310
1311    void liga_desliga_grupos_automaticos(uint8_t ld)
1312    {
1313      int adr,ref,rf,n,m;
1314      byte atual;
1315
1316      for (n=1;n<=12;n++)                                   // Varre os grupos
1317      {
1318        if ((grupo_am[n][0] == 1) && (grupo_am[n][1] == 0))     // Grupo ativo e automático, liga ou desliga (ld)
1319        {
1320          for (m=1;m<=grupo_rf[n][0];m++)                   // Varre os refletores do grupo
1321          {
1322            rf = grupo_rf[n][m];                            // refletor (entre 1 e 12)
1323            if (rf <= 8) {adr = pcf1; ref = rf;}            // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1324            else {adr = pcf2; ref = rf - 8;}                // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1325
1326            Wire.requestFrom(adr,1);                        // Solicita 1 byte ao Pcf correto
1327            while (Wire.available() == 0);                  // Aguarda estar no buffer
1328            atual = Wire.read();                            // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1329                                                            // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1330                                                            // -> refletor 12
1331            if (ld == 1)                                    // Liga refletor do grupo
```

```
1332            {
1333              atual |= 1 << (ref - 1);                          // ld = 1 => liga
1334              refletor_ligado[rf] = true;                       // Marca como refletor ligado
1335            }
1336            else                                                // Desliga refletores do grupo
1337            {
1338              atual &= ~(1 << (ref - 1));                       // ld = 0 => desliga
1339              refletor_ligado[rf] = false;                      // Marca como refletor desligado
1340            }
1341
1342            // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1343            Wire.beginTransmission(adr);                        // Inicia transmissão I2C para o Pcf correto
1344            Wire.write(atual);                                  // Novo byte de acionamento dos refletores do grupo
1345            Wire.endTransmission();                             // Encerra transmissão I2C
1346          }
1347          if (ld == 1) grupo_am[n][2] = 1;                      // Marca grupo como ligado
1348          else grupo_am[n][2] = 0;                              // Marca grupo como desligado
1349        }
1350      }
1351    }
1352
1353    //_____________________________________________________________________________________________________________
1354
1355    void liga_desliga_grupos_programados(void)
1356    {
1357      int adr,ref,rf,n,m,ld;
1358      byte atual;
1359      float agora;
1360
1361      DateTime now = rtc.now();
1362      agora = (float)now.hour() + ((float)now.minute() / 100.0);
1363
1364      for (n=1;n<=12;n++)                                       // Varre os grupos
1365      {
1366        if ((grupo_am[n][0] == 1) && (grupo_am[n][1] == 1))     // Grupo ativo e programado
1367        {
```

```
1368          // Vamos verificar se está na hora de ligar ou desligar grupo
1369          if ((agora >= grupo_if[n][0]) && (grupo_am[n][2] == 0)) ld = 1; // Hora de ligar
1370          else if ((agora >= grupo_if[n][1]) && (agora < grupo_if[n][0]) && (grupo_am[n][2] == 1)) ld = 0;  // Hora de
1371                                                                                                           // desligar
1372          else;
1373
1374          for (m=1;m<=grupo_rf[n][0];m++)                          // Varre os refletores do grupo
1375          {
1376            rf = grupo_rf[n][m];                                   // refletor (entre 1 e 12)
1377            if (rf <= 8) {adr = pcf1; ref = rf;}                   // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1378            else {adr = pcf2; ref = rf - 8;}                       // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1379
1380            Wire.requestFrom(adr,1);                               // Solicita 1 byte ao Pcf correto
1381            while (Wire.available() == 0);                         // Aguarda estar no buffer
1382            atual = Wire.read();                                   // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1383                                                                   // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1384                                                                   // -> refletor 12
1385            if (ld == 1)                                           // Liga refletor do grupo
1386            {
1387              atual |= 1 << (ref - 1);                             // ld = 1 => liga
1388              refletor_ligado[rf] = true;                          // Marca como refletor ligado
1389            }
1390            else                                                   // Desliga refletores do grupo
1391            {
1392              atual &= ~(1 << (ref - 1));                          // ld = 0 => desliga
1393              refletor_ligado[rf] = false;                         // Marca como refletor desligado
1394            }
1395
1396            // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1397            Wire.beginTransmission(adr);                           // Inicia transmissão I2C para o Pcf correto
1398            Wire.write(atual);                                     // Novo byte de acionamento dos refletores do grupo
1399            Wire.endTransmission();                                // Encerra transmissão I2C
1400          }
1401          if (ld == 1) grupo_am[n][2] = 1;                         // Marca grupo como ligado
1402          else grupo_am[n][2] = 0;                                 // Marca grupo como desligado
1403        }
```

```
1404      }
1405    }
1406
1407    //_____________________________________________________________________________________________________________
1408
1409    unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano)
1410    {
1411      unsigned int an,nd = 0;
1412      unsigned int dd[14] = {0,0,31,59,90,120,151,181,212,243,273,304,334,365};
1413
1414      for (an=2019;an<ano;an++)
1415      {
1416        if ((an % 4) == 0) nd += 366;                              // É ano bissexto (vale até 2100)
1417        else nd += 365;
1418      }
1419      if (((an % 4) == 0) && (mes > 2)) nd += (dd[mes] + dia + 1);  // É ano bissexto (vale até 2100)
1420      else nd += (dd[mes] + dia);
1421      return nd;
1422    }
1423
1424    //_____________________________________________________________________________________________________________
1425
1426    bool SD_readStringUntil(const char car)
1427    {
1428      // Lê o string de caracteres até o primeiro caractere car encontrado inclusive
1429
1430      str = "";
1431      char cc;
1432
1433      while(1)
1434      {
1435        cc = arquivo.read();
1436        if ((uint8_t)cc == -1) return false;
1437        if (cc != car) str += cc;
1438        else
1439        {
```

```
1440          str += cc;
1441          return true;
1442        }
1443      }
1444    }
1445
1446    //_____________________________________________________________________________________________________________
1447
1448    bool tem_algo_ligado(void)
1449    {
1450      uint8_t k;
1451      bool ret = false;
1452
1453      for (k=1;k<=12;k++) ret |= refletor_ligado[k];
1454      return ret;
1455    }
1456
1457    //_____________________________________________________________________________________________________________
1458
1459    void liga_inversor(void)
1460    {
1461      byte este;
1462
1463      Wire.requestFrom(pcf2,1);                    // Solicita 1 byte ao Pcf2
1464      while (Wire.available() == 0);               // Aguarda estar no buffer
1465      este = Wire.read();                          // Lê o byte, bit 7 -> inversor
1466      este |= 1 << 7;                              // Liga bit 7
1467      // Atualiza Pcf2 para ligar inversor
1468      Wire.beginTransmission(pcf2);                // Inicia transmissão I2C para o Pcf2
1469      Wire.write(este);                            // Novo byte de acionamento
1470      Wire.endTransmission();                      // Encerra transmissão I2C
1471      inversor_ligado = true;
1472    }
1473
1474    //_____________________________________________________________________________________________________________
1475
```

```
1476  void desliga_inversor(void)
1477  {
1478    byte este;
1479
1480    Wire.requestFrom(pcf2,1);                          // Solicita 1 byte ao Pcf2
1481    while (Wire.available() == 0);                     // Aguarda estar no buffer
1482    este = Wire.read();                                // Lê o byte, bit 7 -> inversor
1483    este &= ~(1 << 7);                                 // Desliga bit 7
1484    // Atualiza Pcf2 para desligar inversor
1485    Wire.beginTransmission(pcf2);                      // Inicia transmissão I2C para o Pcf correto
1486    Wire.write(este);                                  // Novo byte de acionamento
1487    Wire.endTransmission();                            // Encerra transmissão I2C
1488    inversor_ligado = false;
1489  }
1490
1491  //____________________________________________________________________________________________________________________________
1492
```