```
/*****************************************************************************************************************
* This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General    *
* Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.  *
* This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied   *
* warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more  *
* details. You should have received a copy of the GNU Lesser General Public License along with this program.          *
* If not, see <http://www.gnu.org/licenses/>.                                                                         *
*                                                                                                                     *
* © Copyright 2019-2020 Filippo Pardini - filippo@robotica.eng.br - http://blog.robotica.eng.br/                      *
*****************************************************************************************************************/
/*
  Projeto CoMoSisFog V3.1.3 (03/05/2020)
  * Projeto para Controle e Monitoramento de um Sistema Fotovoltaico off grid para iluminacao de um jardim

  * O sistema é constituído por 8 painéis "Canadian CS6K-270" de 270Wp cada. Eles estão interligados de forma a
  * estabelecer dois conjuntos iguais de 4 painéis. Cada conjunto tem cada 2 painéis interligados em série e por sua
  * vez interligados em paralelo. Cada conjunto alimenta um controlador "Controlador de Carga MPPT Epsolar Tracer-4210A
  * 40A 12/24V" através de um stringbox de proteção. Os 2 controladores fazem a carga de um banco de baterias
  * constituído de 6 baterias estacionarias "Heliar Freedom DF4100" de 240Ah cada. Estas baterias estão interligadas
  * de forma a obter um banco de 24V (dois conjuntos (cj1 e cj2) em série de três baterias em paralelo cada um,
  * balanceados por equalizadores). Os controladores estão interligados ao banco de baterias através do stringbox de
  * proteção de forma invertida entre eles (um controlador alimenta o + do cj1 e o - do cj2 e o outro controlador
  * alimenta o - do cj1 e o + do cj2). Ao banco de baterias está ligado, via um disjuntor DC de 80A, um "Inversor
  * Senoidal Epsolar SHI2000-22 - 2000VA / 24Vcc / 220Vca" que alimenta os refletores LED do jardim (~ 550W) em 220Vca
  * através de fusiveis de proteção. O controle e monitoramento deste sistema é feito através de duas Partes:
  * Parte 2 - um "controlador dos refletores" e Parte 1 - um "monitor dos parâmetros elétricos de carga e descarga do
  * banco de baterias". O controlador e o monitor estão integrados no projeto CoMoSisFog que usa técnicas de IOT.

  Parte 1 - Quiosque (onde estão os paineis solares)

  ESP32 local - monitor
  * Esta parte utiliza uma placa DOIT ESP32 DEVKIT V1, display Oled, um RTC DS3231, uma eeprom 24C32 - 4K que vem com o
  * RTC DS3231, dois expansores I2C PCF8574, 3 placas de 4 relés com isolação ótica, uma fotocelula, 3 sensores INA226,
  * um Xbee, um botão NA e um micro SD. Ela faz o monitoramento dos tres INA226 (INA1 - carga do banco de baterias
  * via Tracer1, INA2 - carga do banco de baterias via Tracer2, INA3 - descarga do banco de baterias via inversor)
  * usando o RTC DS3231M, mostrando as leituras atuais de corrente, voltagem, potencia e energia armazenada e consumida
  * no display Oled e enviando, via Xbee, para o ESP32 da Parte 2. Grava no SD as voltagens máximas e mínimas das INAs
  * e energia gerada e consumida em 24h (para análise e calibração de ciclos liga/desliga). Em função desses dados, da
  * fotocélula, dos dados configurados pelo ESP32 remoto (Parte 2 via Xbee), controla e disponibiliza energia em
  * horários pré-configurados para os refletores, executa comandos e solicitações do ESP32 remoto e gerencia a energia
```

```
41    * fornecida pelas baterias, otimizando-as.
42
43    Parte 2 - Laboratorio (na residência)
44
45    ESP32 remoto - controlador
46    * Esta parte utiliza uma placa DOIT ESP32 DEVKIT V1, uma eeprom At24c256 - 32KB, um Nanoshield ADC circuitar, um
47    * display LCD03 com keypad, um buzzer, dois botões NA, um Xbee, um PC/keyboard/mouse, uma fonte 5Vdc com bateria
48    * ion-litio 3S e um celular. Ela atua como servidor AP para comunicação com o celular (o qual envia comandos
49    * para acionar os grupos de refletores) e como entrada de configurações, solicitações e comandos, via PC, para a
50    * Parte 1 via Xbee. Além disso, faz o display, no LCD03, dos dados enviados pela parte 1, comanda os grupos de
51    * refletores pelo keypad e envia os dados para a nuvem via MQTT (cloudMQTT) e ThingSpeak (para analise com MATLAB).
52  */
53
54  //*************************************************************************************************************
55  //****************************************** Este é o sketch da Parte 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 "FS.h"                                                   // file system wrapper
65  #include "SD.h"                                                   // micro SD
66  #include "SPI.h"                                                  // Interface SPI
67
68  //_____________________________________________________________________________________________________________
69
70  // Para display no monitor serial das configuracoes dos INA226 descomentar o comando abaixo e todos os checkConfig(n) em
71  // void configura_INAs(unsigned char ch)
72  //#include "checkConfig.h"                                          // Lista no monitor serial as configurações dos 3 INA226
73
74  //_____________________________________________________________________________________________________________
75
76  // Pinos I2C
77  #define SDApin 21                                                 // I2C SDA
78  #define SCLpin 22                                                 // I2C SCL
79  #define i2c_addr_eeprom 0x57                                      // Endereco I2C da 24C32 (interna ao RTC DS3231)
80  #define pcf1 0x20                                                 // Endereço I2C PCF8574-1
```

```
81  #define pcf2 0x21                                   // Endereço I2C PCF8574-2
82  #define oled 0x3C                                   // Endereço I2C OLED
83  #define endRTC 0x68                                 // Endereço I2C RTC DS3231M
84  #define endINA1 0x40                                // Endereço I2C INA1 Tracer1
85  #define endINA2 0x41                                // Endereço I2C INA2 Tracer2
86  #define endINA3 0x44                                // Endereço I2C INA3 Inversor
87
88  //_____________________________________________________________________________________________________________
89
90  template <class T> int eeWrite(int ee, const T& value)      // Classe genérica para gravação da eeprom
91  {                                                           // ee - posição na eeprom, value - estrutura de dados
92    const byte* p = (const byte*)(const void*)&value;         // Pointer para a estrutura
93    int i;                                                    // Contador
94    Wire.beginTransmission(i2c_addr_eeprom);                  // Inicia transmissão I2C
95    Wire.write((int)(ee >> 8)); // MSB                       // Envia MSB da posição
96    Wire.write((int)(ee & 0xFF)); // LSB                     // Envia LSB da posição
97    for (i = 0; i < sizeof(value); i++)                       // Para todos os bytes da estrutura
98    Wire.write(*p++);                                         // Envia o byte
99    Wire.endTransmission();                                   // Encerra transmissão I2C
100   return i;                                                 // Retorna o número de bytes gravados
101 }
102
103 template <class T> int eeRead(int ee, T& value)             // Classe genérica para leitura da eeprom
104 {                                                           // ee - posição na eeprom, value - estrutura de dados
105   byte* p = (byte*)(void*)&value;                           // Pointer para a estrutura
106   int i;                                                    // Contador
107   Wire.beginTransmission(i2c_addr_eeprom);                  // Inicia transmissão I2C
108   Wire.write((int)(ee >> 8)); // MSB                       // Envia MSB da posição
109   Wire.write((int)(ee & 0xFF)); // LSB                     // Envia LSB da posição
110   Wire.endTransmission();                                   // Encerra transmissão I2C
111   Wire.requestFrom(i2c_addr_eeprom,sizeof(value));          // Solicita à eeprom o número de bytes da estrutura
112   for (i = 0; i < sizeof(value); i++)                       // Para esse número de bytes
113   if(Wire.available())                                      // Se tem dados no buffer
114   *p++ = Wire.read();                                       // Lê o byte para a estrutura
115   return i;                                                 // Retorna o número de bytes lidos
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   // Instancias
132   elapsedMillis timeElapsed1;                         // Instancia elapsedMillis
133   elapsedMillis timeElapsed2;                         // Instancia elapsedMillis
134   elapsedMillis timeElapsed;                          // Instancia elapsedMillis
135   elapsedMillis timeElapsed24h;                       // Instancia elapsedMillis
136   INA226 ina1;                                        // Instancia INA226 Tracer1
137   INA226 ina2;                                        // Instancia INA226 Tracer2
138   INA226 ina3;                                        // Instancia INA226 Inversor
139   RTC_DS3231 rtc;                                     // Instancia RTC_DS3231
140   SSD1306Wire display(oled,SDApin,SCLpin);            // Instância OLED Para ESP32
141
142   //_____________________________________________________________________________________________________________
143
144   // Variaveis e constantes
145   // SPI MOSI - 23
146   // SPI MISO - 19
147   // SPI SCK  - 18
148   // SPI SS   - 5
149   #define fotocelula 25                               // Pino de entrada fotocélula
150   #define inputPin 26                                 // Pino para reinicialização da eeprom
151   #define RXpin 16                                    // UART RX
152   #define TXpin 17                                    // UART TX
153   #define ACK Serial2.write(0x6)                      // ACK
154   uint8_t noite;
155   bool foinoite;
156   bool ad1,ad2,ad3,ad4,ad5,ad6,ad7,ad8;
157   float temperatura;
158
159   char daysOfTheWeek[7][12] = {"Domingo", "Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado"};
160   float delta1,delta2,delta3,etotalin,coef1,vina1,cina1,pina1,vina2,cina2,pina2,vina3,cina3,pina3;
```

```
161  float etotalin_24h,etotalin_24h_inicial,etotalout_24h,etotalout_24h_inicial;
162  char v_ina1[6],c_ina1[6],p_ina1[6],e_ina1[6];
163  char v_ina2[6],c_ina2[6],p_ina2[6],e_ina2[6];
164  char v_ina3[6],c_ina3[6],p_ina3[6],e_ina3[6];
165  String tempo,VB,AT,PT,WT,registro,str;
166  int grupo_rf[13][13];                                  // grupo_rf[grupo][x] -> refletores que compõe o grupo
167                                                         // grupo_rf[grupo][0] -> número de refletores no grupo
168  int grupo_am[13][3];                                   // grupo_am[grupo][0] -> ativo(0-inativo,1-ativo),
169                                                         // grupo_am[grupo][1] -> modo(0-automático(fotocélula),
170                                                         // 1-programado(horario de liga e desliga))
171                                                         // grupo_am[grupo][2] -> grupo ligado(1), grupo desligado(0)
172  float grupo_if[13][2];                                 // grupo_if[grupo][0] -> liga programação horaria
173                                                         // grupo_if[grupo][1] -> desliga programação horaria
174  bool refletor_ligado[13];                              // true -> refletor ligado, false -> refletor desligado
175  unsigned long key = 2863311530;                        // Código de eeprom válida
176  String versao = "V3.1.3";                              // ************* Versão ***********************************
177  uint8_t j;
178  uint8_t jj;
179  unsigned int ii = 0;
180  uint8_t grupo;
181  char par;
182  bool inversor_ligado;
183  uint8_t hora;
184  bool pula;
185  unsigned int inicio,fim,datseq;
186  File arquivo;
187
188  float vina1_max_dia;                                   // Voltagem máxima INA1 no dia
189  float vina1_min_dia;                                   // Voltagem mínima INA1 no dia
190  float vina2_max_dia;                                   // Voltagem máxima INA2 no dia
191  float vina2_min_dia;                                   // Voltagem mínima INA2 no dia
192  float vina3_max_dia;                                   // Voltagem máxima INA3 no dia
193  float vina3_min_dia;                                   // Voltagem mínima INA3 no dia
194
195  //_____________________________________________________________________________________________________________
196
197  // Funções
198  void telainicial();
199  void configura_INAs(unsigned char ch);
200  void saidas_altas_pcf1_pcf2(void);
```

```
201  bool toggle_grupo_refletores(uint8_t grp);
202  void inicializa_grupos_refletores(void);
203  void limpa_buffer_serial(uint8_t cmp);
204  bool ppe(uint8_t ncar,unsigned long intervalo,uint8_t deonde);
205  void liga_desliga_grupos_automaticos(uint8_t ld);
206  void liga_desliga_grupos_programados(void);
207  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano);
208  bool SD_readStringUntil(const char car);
209  bool tem_algo_ligado(void);
210  void liga_inversor(void);
211  void desliga_inversor(void);
212
213  //_____________________________________________________________________________________________________________________________
214  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
215  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
216  //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
217  //_____________________________________________Processa uma vez _____________________________________________________________
218
219  void setup()
220  {
221    byte error,addr;
222    int nDevices;
223
224    // Definição pinos ____________________________________________________________________________________________________________
225    pinMode(fotocelula, INPUT_PULLUP);                    // Modo do pino fotocelula
226    pinMode(inputPin, INPUT_PULLUP);                      // Modo do inputPin para eeprom default
227
228    // Inicia monitor serial ______________________________________________________________________________________________________
229    Serial.begin(115200);                                 // Define velocidade
230    while (!Serial);                                      // aguarda
231
232    // Inicia I2C _________________________________________________________________________________________________________________
233    Wire.begin(SDApin,SCLpin);                            // Define pinos I2C
234
235    // _____________________________________________ Inicio varredura I2C ____________________________________________________
236
237    Serial.println("\nVarredura I2C");
238    Serial.println("Procurando...\n");
239    nDevices = 0;
240    ad1=ad2=ad3=ad4=ad5=ad6=ad7=ad8 = false;
```

```
241
242     for(addr = 1;addr < 127;addr++)
243     {
244       Wire.beginTransmission(addr);
245       error = Wire.endTransmission();
246       if (error == 0)
247       {
248         switch (addr)
249         {
250           case 0x57:
251             Serial.print("Encontrada eeprom 24C32 no endereço 0x57\n");
252             ad1 = true;
253             break;
254           case 0x20:
255             Serial.print("Encontrado PCF8574_1 no endereço 0x20\n");
256             ad2 = true;
257             break;
258           case 0x21:
259             Serial.print("Encontrado PCF8574_2 no endereço 0x21\n");
260             ad3 = true;
261             break;
262           case 0x3C:
263             Serial.print("Encontrado OLED no endereço 0x3C\n");
264             ad4 = true;
265             break;
266           case 0x68:
267             Serial.print("Encontrado RTC DS3231M no endereço 0x68\n");
268             ad5 = true;
269             break;
270           case 0x40:
271             Serial.print("Encontrado INA1 Tracer1 no endereço 0x40\n");
272             ad6 = true;
273             break;
274           case 0x41:
275             Serial.print("Encontrado INA2 Tracer2 no endereço 0x41\n");
276             ad7 = true;
277             break;
278           case 0x44:
279             Serial.print("Encontrado INA3 Inversor no endereço 0x44\n");
280             ad8 = true;
```

```
281             break;
282           default:
283             Serial.print("Endereço desconhecido\n");
284         }
285         nDevices++;
286       }
287       else if (error==4)
288       {
289         Serial.print("Erro desconhecido no endereço 0x");
290         if (addr<16) Serial.print("0");
291         Serial.println(addr,HEX);
292       }
293     }
294     if (nDevices == 0) Serial.println("Nenhum equipamento encontrado\n");
295     if (nDevices == 8) Serial.println("Todos os equipamentos presentes\n");
296
297     // _____________________________________________ Fim varredura I2C ____________________________________________
298
299     // Inicia Xbee ____________________________________________________________________________________________________
300     Serial2.begin(115200, SERIAL_8N1, RXpin, TXpin);
301
302     // Inicia cartão SD _______________________________________________________________________________________________
303     if(!SD.begin())
304     {
305       Serial.print("\nMontagem do cartao falhou");            // Montagem do cartão falhou
306       Serial2.print("<Montagem SD falhou>");                  // Envia para P2 para display no LCD03
307       while(1);
308     }
309     uint8_t cardType = SD.cardType();                         // Tipo do cartão
310     if(cardType == CARD_NONE){
311       Serial.print("\nSem SD inserido");                      // Sem cartão
312       Serial2.print("<Sem SD inserido>");                     // Envia para P2 para display no LCD03
313       while(1);
314     }
315     Serial.print("\nSD inicializado");                       // Tipo do cartão SD
316     Serial.print("\nSD tipo: ");
317     if(cardType == CARD_MMC) Serial.print("MMC");
318     else if(cardType == CARD_SD) Serial.print("SDSC");
319     else if(cardType == CARD_SDHC) Serial.print("SDHC");
320     else Serial.print("DESCONHECIDO");
```

```
321
322     uint64_t cardSize = SD.cardSize() / (1024 * 1024);        // Tamanho do cartão SD
323     Serial.printf("\nSD tamanho: %lluMB", cardSize);
324
325     //*************************************************************************************************************
326     //SD.remove("/dados.txt");                                  // *******Só quando precisa reinicializar /dados.txt********
327     //*************************************************************************************************************
328
329     arquivo = SD.open("/dados.txt", FILE_APPEND);             // Abre o arquivo para append
330     if(!arquivo)
331     {
332       Serial.print("\nFalha na abertura do arquivo");
333       Serial2.print("<Falha no arquivo (1)>");                 // Envia para P2 para display no LCD03
334       while(1);
335     }
336
337     // Inicia display Oled ____________________________________________________________________________________________
338     display.init();
339
340     // Inicia rtc
341     if (!rtc.begin())
342     {
343       Serial.print("\nRTC inexistente");
344       while (1);
345     }
346
347     //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));           // Só para acertar o RTC, depois de carregar e executar,
348                                                               // comentar (//) e recarregar
349
350     // Inicializa os grupos de refletores _____________________________________________________________________________
351     inicializa_grupos_refletores();                           // Inicializa os PCF8574 com as saidas 0 (grupos desligados)
352
353     // Inicializa INAs 226 ____________________________________________________________________________________________
354     if (ad6 & ad7 & ad8)
355     {
356       configura_INAs(1);                                      // Configura INAs 1 - imprime dados 0 - nao imprime
357       ina1.enableOverPowerLimitAlert();                       // Habilita alerta de OverPower INA1 - Tracer1
358       ina2.enableOverPowerLimitAlert();                       // Habilita alerta de OverPower INA2 - Tracer2
359       ina3.enableBusUnderLimitAlert();                        // Habilita alerta de UnderLimitVoltage INA3 - Inversor
360       ina1.setPowerLimit(1084.0);                             // Configura potência limite INA1 - Tracer1
```

```
361       ina2.setPowerLimit(1084.0);                          // Configura potência limite INA2 - Tracer2
362       ina3.setBusVoltageLimit(22.2);                       // Configura Voltagem limite INA3 - Inversor
363     }
364
365     // Inicializa pcf1 e pcf2 com as saidas altas para que os relés iniciem desligados ____________________________
366     saidas_altas_pcf1_pcf2();                              // Coloca as saidas dos pcf1 e pcf2 altas porque elas
367                                                            // iniciam baixas e os reles são ligados com baixa.
368                                                            // Inicialmente portanto, como precisamos que os relés
369                                                            // inciem desligados, temos que colocar as saidas
370                                                            // iniciais dos pcf1 e pcf2 altas
371
372     // Display da tela inicial no OLED ____________________________________________________________________________
373     telainicial();
374
375     // Le eeprom __________________________________________________________________________________________________
376     eeRead(0,eeprom);
377     delay(30);
378
379     // Se eeprom não válida ou reinicialização forçada via inputPin - define eeprom default
380     if ((eeprom.valido != key) || (digitalRead(inputPin) == LOW))
381     {
382       eeprom.etracer1 = 0.0;
383       eeprom.etracer2 = 0.0;
384       eeprom.einversor = 0.0;
385       eeprom.inter1 = 60000;                               // 1 minuto
386       eeprom.inter2 = 300000;                              // 5 minutos
387       eeprom.cod_mon = 1;                                  // Habilita/desabilita monitor serial (não usado no momento)
388       eeprom.valido = key;                                 // Valida a eeprom
389
390       eeWrite(0,eeprom);                                   // Grava eeprom
391       delay (30);
392
393       Serial.print("\neeprom default");
394       Serial.print("\ndeixar o pino inputPin aberto");
395     }
396     else  Serial.print("\neeprom valida");                  // Eeprom valida
397
398     coef1 = eeprom.inter1 / 3600000.0;                     // Para calculo da energia (Wh)
399
400     Serial.print("\nconfiguracao na eeprom");
```

```
  Serial.print("\ninterval1    " + String(eeprom.inter1));
  Serial.print("\ninterval2    " + String(eeprom.inter2));
  Serial.print("\ncod_monitor  " + String(eeprom.cod_mon));

  // Inicializações ____________________________________________________________________________________________________
  pula = false;
  foinoite = false;
  inversor_ligado = false;
  timeElapsed1 = 0;                                         // Inicializa contagem
  timeElapsed2 = 0;                                         // Inicializa contagem
  etotalin = 0.0;                                           // Energia total de entrada
  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 ((digitalRead(fotocelula) == LOW)) noite = 1;          // Verifica a fotocelula - 1 => noite; 0 => dia
  else noite = 0;

}

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

void loop()
{
  DateTime now = rtc.now();                                 // Pega data e hora atual

  // Monta string com data e hora atual
  hora = now.hour();
  tempo = String(now.day()) + '/' + String(now.month()) + '/' + String(now.year()) + ' ' + String(hora)
  + ':' + String(now.minute()) +  ':' + String(now.second());

  if ((hora == 5) && !pula)                                 // Inicia periodo de 24h às 05h, executa uma única vez
  {
    // Vamos iniciar a contabilizar quanta energia foi acumulada e quanta foi consumida no periodo de 24h (dia/noite)
```

```
441       timeElapsed24h = 0;
442       etotalin_24h_inicial = etotalin;                          // Energia total de entrada até agora (05h)
443       etotalout_24h_inicial = eeprom.einversor;                 // Energia total de saída até agora (05h)
444       pula = true;
445     }
446
447     // Verifica se tem comunicacao vindo de P2 para solicitação, configuração ou comando
448     if (Serial2.available() > 0)
449     {
450       // Verifica se P2 enviou alguma coisa: comando, coniguração p1, configuração dos grupos de refletores, configuração
451       // de seus modos ou solicitação de dados
452       // Indicadores válidos:
453       // '!' - Comando toggle dos grupos de refletores - formato - !(g) - onde g é o número do grupo. g = 0 => tudo
454       // '$' - Solicitação de dados - formato - $(i) - onde i=1 data-hora, i=2 dados INAs e energias, i=3 status grupos
455       //       e refletores
456       // '@' - Solicitação de dados - formato - @(di,df) - onde di - data de início em dias desde 01/01/2019, df - data
457       //       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
458       //       os dados relativos a esse período
459       // '#' - Configuração de ciclos em p1 - formato - #n(v) - onde n=1 interval1_p1, n=2 interval2_p1,
460       //       n=3 cod_monitor_p1, v = 0 ou 1 ou valor em minutos
461       // '&' - Configuração dos grupos de refletores - formato - &n(r,...,r) - onde n é o número do grupo
462       //       e r o número do refletor. grupo_rf[grupo][13] vai conter o número de refletores registrados no grupo.
463       //       O grupo_rf[grupo][0] contém o número de refletores do grupo. O grupo_rf[0][refletores]tem que ser definido
464       //       com todos os refletores
465       // '*' - Configuração dos horarios liga/desliga dos grupos e modo de operação de cada grupo
466       //       grupo_am[grupo][0] - modo de operação: ativo -> sim(1)/não(0), grupo_am[grupo][1] - automatico(0) ->
467       //       fotocelula, programado(1) -> hora liga/desliga, formato - *n(a,b,c,d) - onde n é o número do grupo
468       //       a=0 -> inativo, a=1 -> ativo, b=0 -> automático(fotocelula), b=1 ->programado(c e d só existem neste caso),
469       //       c -> hora liga (formato: h.m h->horas, m->minutos/100 guardado como float h+m), d -> hora desliga (idem)
470
471       // Monta string com data e hora atual
472       //DateTime now = rtc.now();                                 // Pega data e hora atual
473       tempo = String(now.day()) + '/' + String(now.month()) + '/' + String(now.year()) + ' ' + String(now.hour())
474       + ':' + String(now.minute()) +  ':' + String(now.second());
475
476       while (Serial2.available() > 0)                           // Enquanto tiver dados no buffer
477       {
478         par = Serial2.read();
479         switch (par)
480         {
```

```
481           case '!':                                          // Pedido para toggle de um grupo
482             if (!toggle_grupo_refletores(Serial2.parseInt())) Serial.print("\nToggle nao executado - grupo inativo");
483             else Serial.print("\nToggle executado - grupo ativo");
484             ACK;                                             // Confirma OK para P2
485             break;
486           case '$':                                          // Solicitação de data/hora, dados INAs
487             switch (Serial2.parseInt())
488             {
489               case 1:                                        // Solicitação de data/hora
490                 Serial2.print(tempo.c_str());
491                 break;
492               case 2:                                        // Solicitação de dados INAs
493                 Serial2.print("v1("); Serial2.print(vina1,2); Serial2.print("),");
494                 Serial2.print("c1("); Serial2.print(cina1,2); Serial2.print("),");
495                 Serial2.print("p1("); Serial2.print(pina1,2); Serial2.print("),");
496                 Serial2.print("v2("); Serial2.print(vina2,2); Serial2.print("),");
497                 Serial2.print("c2("); Serial2.print(cina2,2); Serial2.print("),");
498                 Serial2.print("p2("); Serial2.print(pina2,2); Serial2.print("),");
499                 Serial2.print("v3("); Serial2.print(vina3,2); Serial2.print("),");
500                 Serial2.print("c3("); Serial2.print(cina3,2); Serial2.print("),");
501                 Serial2.print("p3("); Serial2.print(pina3,2); Serial2.print("),");
502                 Serial2.print("ei("); Serial2.print(etotalin,2); Serial2.print("),");
503                 Serial2.print("eo("); Serial2.print(eeprom.einversor,2); Serial2.print(")");
504                 break;
505               case 3:                                        // Solicitação do status de grupos e refletores
506                 for (j=0;j<=12;j++)                          // Composição dos grupos
507                 {
508                   str = "\nGrupo " + String(j) + " => ";     // Prepara linha do grupo
509                   for (jj=1;jj<=grupo_rf[j][0];jj++)         // Acrescenta os refletores
510                   {
511                     str += String(grupo_rf[j][jj]);
512                     if (jj < grupo_rf[j][0]) str += ",";
513                     if (jj == grupo_rf[j][0]) str += "#";    // Encerra grupo
514                   }
515                   Serial2.print(str.c_str());                // Envia grupo para P2
516                 }
517                 for (j=1;j<=12;j++)                          // Status dos refletores
518                 {
519                   str = "\nRefletor " + String(j) + " => ";  // Prepara linha do status
520                   if (refletor_ligado[j]) str += "ligado#";
```

```
521                 else str += "desligado#";
522                 Serial2.print(str.c_str());                 // Envia status para P2
523               }
524               break;
525           }
526         break;
527         case '#':
528           eeRead(0,eeprom);                                 // Le eeprom
529           delay(30);
530           switch (Serial2.parseInt())                       // Configuraçãoi de ciclos
531           {
532             case 1:                                         // É interval1_p1
533               eeprom.inter1 = Serial2.parseInt();
534               break;
535             case 2:                                         // É interval2_p1
536               eeprom.inter2 = Serial2.parseInt();
537               break;
538             case 3:                                         // É cod_monitor_p1
539               eeprom.cod_mon = Serial2.parseInt();
540               break;
541           }
542           eeWrite(0,eeprom);                                // Grava eeprom
543           delay (30);
544
545           // Avisa no display Oled
546           display.clear();
547           display.setTextAlignment(TEXT_ALIGN_LEFT);
548           display.setFont(ArialMT_Plain_10);
549           display.drawString(1, 5, tempo);
550           display.drawString(1, 15, "nova configuracao gravada");
551           display.drawString(1, 25, "interval1       " + String(eeprom.inter1));
552           display.drawString(1, 35, "interval2       " + String(eeprom.inter2));
553           display.drawString(1, 45, "cod_monitor  " + String(eeprom.cod_mon));
554           display.display();
555           delay(5000);
556
557           // Avisa no monitor serial
558           Serial.print("\nnova configuracao gravada na eeprom");
559           Serial.print("\ninterval1    " + String(eeprom.inter1));
560           Serial.print("\ninterval2    " + String(eeprom.inter2));
```

```
561             Serial.print("\ncod_monitor  " + String(eeprom.cod_mon));
562
563             ACK;                                          // Confirma OK para P2
564             break;
565           case '&':                                       // Configuração da composição dos grupos
566             // Nota: o grupo 0 deve conter todos os refletores para comando total
567             grupo = Serial2.parseInt();
568             j = 1;
569             while (Serial2.available() > 0)               // Enquanto tiver dados no buffer
570             {
571               grupo_rf[grupo][j] = Serial2.parseInt();    // Armazena refletor
572               j++;
573             }
574             grupo_rf[grupo][0] = j;                       // Número de refletores no grupo
575             ACK;                                          // Confirma OK para P2
576             break;
577           case '*':                                       // Configuração do modo dos grupos
578             grupo = Serial2.parseInt();
579             grupo_am[grupo][0] = Serial2.parseInt();
580             grupo_am[grupo][1] = Serial2.parseInt();
581             if (grupo_am[grupo][1] == 1)                  // É programado
582             {
583               grupo_if[grupo][0] = Serial2.parseFloat();  // Liga h.m (h - hora(0-24), m - minutos(0-60))
584               grupo_if[grupo][1] = Serial2.parseFloat();  // Desliga h.m (h - hora(0-24), m - minutos(0-60))
585             }
586             ACK;                                          // Confirma OK para P2
587             break;
588           case '@':                                       // Solicitação de dados gravado0s no SD
589             inicio = Serial2.parseInt();
590             fim = Serial2.parseInt();
591             if (inicio > fim)
592             {
593               Serial.print("\nData de inicio maior que data de fim");
594               break;
595             }
596             arquivo.close();
597             arquivo = SD.open("/dados.txt");
598             if(!arquivo)
599             {
600               //Serial.print("\nFalha na abertura do arquivo");
```

```
601               Serial2.print("<Falha no arquivo 2>");
602               while(1);
603             }
604             else
605             {
606               if (!SD_readStringUntil('<'))                    // Lê o primeiro registro
607               {
608                 //Serial.print("\nRegistro nao encontrado no SD - erro 1");
609                 Serial2.print("<Falha registro 1>");
610                 while(1);
611               }
612               // Encontrou o primeiro registro
613               datseq = arquivo.parseInt();                     // Vamos ler a datseq do primeiro registro
614
615               while (datseq < fim)                             // Vamos executar enquanto datseq < fim
616               {
617                 while (datseq < inicio)                        // Se o datseq é menor que o inicio,
618                 {
619                   if (!SD_readStringUntil('<'))                // vamos ler o arquivo até o próximo registro
620                   {
621                     //Serial.print("\nRegistro nao encontrado no SD - erro 2");
622                     Serial2.print("<Falha registro 2>");
623                     break;
624                   }
625                   datseq = arquivo.parseInt();                 // até acharmos um registro com datseq >= inicio
626                 }
627
628                 // Achamos um, vamos imprimir
629                 if (!SD_readStringUntil('>'))
630                 {
631                   //Serial.print("\nFim de registro nao encontrado no SD - erro 3");
632                   Serial2.print("<Falha registro 3>");
633                   break;
634                 }
635                 str = '<' + String(datseq)+ str;
636                 Serial.print("\n" + str);
637                 if (!SD_readStringUntil('<'))                  // ler o arquivo até o próximo registro
638                 {
639                   //Serial.print("\nRegistro nao encontrado no SD - erro 4");
640                   Serial2.print("<Falha registro 4>");
```

```
                break;
              }
              datseq = arquivo.parseInt();
            }
            if (datseq == fim)                                // Achamos o último, vamos enviar para Parte 2
            {
              if (!SD_readStringUntil('>'))
              {
                //Serial.print("\nFim de registro nao encontrado no SD - erro 5");
                Serial2.print("<Falha registro 5>");
                while(1);
              }
              str = '<' + String(datseq) + str;
              Serial.print("\n" + str);
            }
            else;
            arquivo.close();
          }
          break;
      }
    }
  }

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

// **************************************************** 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");
      Serial2.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

  }

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

  if (timeElapsed1 >= eeprom.inter1)                    // Executa a cada ciclo de duracao inter1
  {                                                     // Default 1 minuto
    // Lê os sensores INA226
    if (ad6 & ad7 & ad8)
    {
      vina1 = ina1.readBusVoltage();                    // Voltagem INA1
```

```
721         cina1 = ina1.readShuntCurrent();                    // Corrente INA1
722         pina1 = ina1.readBusPower();                        // Potência INA1
723         delta1 = pina1 * coef1;                             // Delta energia
724         vina2 = ina2.readBusVoltage();                      // Voltagem INA2
725         cina2 = ina2.readShuntCurrent();                    // Corrente INA2
726         pina2 = ina2.readBusPower();                        // Potência INA2
727         delta2 = pina2 * coef1;                             // Delta energia
728         vina3 = ina3.readBusVoltage();                      // Voltagem INA3
729         cina3 = ina3.readShuntCurrent();                    // Corrente INA3
730         pina3 = ina3.readBusPower();                        // Potência INA3
731         delta3 = pina3 * coef1;                             // Delta energia
732         eeprom.etracer1 = eeprom.etracer1 + delta1;         // Integra energia
733         eeprom.etracer2 = eeprom.etracer2 + delta2;         // Integra energia
734         eeprom.einversor = eeprom.einversor + delta3;       // Integra energia
735         etotalin = eeprom.etracer1 + eeprom.etracer2;       // Integra energia total acumulada
736       }
737
738       // Imprime data e hora atual
739       Serial.println("\n");
740       Serial.print(tempo);
741       Serial.print("\n");
742       // Imprime leituras relativas ao Tracer1
743       Serial.print("\nINA226-1 Tracer1");
744       Serial.print("\n________________");
745       Serial.print("\nBus voltage:   ");
746       Serial.print(vina1, 2);
747       Serial.print("V");
748       Serial.print("\nShunt current: ");
749       Serial.print(cina1, 2);
750       Serial.print("A");
751       Serial.print("\nBus power:     ");
752       Serial.print(pina1, 2);
753       Serial.print("W");
754       Serial.print("\nBus energy:    ");
755       Serial.print(delta1, 2);
756       Serial.print("Wh");
757       // Imprime leituras relativas ao Tracer2
758       Serial.print("\n");
759       Serial.print("\nINA226-2 Tracer2");
760       Serial.print("\n________________");
```

```
761       Serial.print("\nBus voltage:   ");
762       Serial.print(vina2, 2);
763       Serial.print("V");
764       Serial.print("\nShunt current: ");
765       Serial.print(cina2, 2);
766       Serial.print("A");
767       Serial.print("\nBus power:     ");
768       Serial.print(pina2, 2);
769       Serial.print("W");
770       Serial.print("\nBus energy:    ");
771       Serial.print(delta2, 2);
772       Serial.print("Wh");
773       // Imprime leituras relativas ao Inversor
774       Serial.print("\n");
775       Serial.print("\nINA226-3 Inversor");
776       Serial.print("\n_________________");
777       Serial.print("\nBus voltage:   ");
778       Serial.print(vina3, 2);
779       Serial.print("V");
780       Serial.print("\nShunt current: ");
781       Serial.print(cina3, 2);
782       Serial.print("A");
783       Serial.print("\nBus power:     ");
784       Serial.print(pina3, 2);
785       Serial.print("W");
786       Serial.print("\nBus energy:    ");
787       Serial.print(delta3, 2);
788       Serial.print("Wh");
789       Serial.print("\n");
790
791       //_____________________________________________________________________________________________________________
792
793       // Para display na OLED
794       dtostrf(vina1, 5, 2, v_ina1);
795       dtostrf(cina1, 5, 2, c_ina1);
796       dtostrf(pina1, 5, 2, p_ina1);
797       dtostrf(eeprom.etracer1, 5, 2, e_ina1);
798
799       dtostrf(vina2, 5, 2, v_ina2);
800       dtostrf(cina2, 5, 2, c_ina2);
```

```
801       dtostrf(pina2, 5, 2, p_ina2);
802       dtostrf(eeprom.etracer2, 5, 2, e_ina2);
803
804       dtostrf(vina3, 5, 2, v_ina3);
805       dtostrf(cina3, 5, 2, c_ina3);
806       dtostrf(pina3, 5, 2, p_ina3);
807       dtostrf(eeprom.einversor, 5, 2, e_ina3);
808
809       ii++;
810       if (ii > 3) ii = 1;
811       switch (ii)
812       {
813         case 1:
814         VB = "V-Banco     " + String(v_ina1);
815         AT = "A-Tracer1   " + String(c_ina1);
816         PT = "P-Tracer1   " + String(p_ina1);
817         WT = "Wh-Tracer1 " + String(e_ina1);
818         break;
819         case 2:
820         VB = "V-Banco     " + String(v_ina2);
821         AT = "A-Tracer2   " + String(c_ina2);
822         PT = "P-Tracer2   " + String(p_ina2);
823         WT = "Wh-Tracer2 " + String(e_ina2);
824         break;
825         case 3:
826         VB = "V-Banco      " + String(v_ina3);
827         AT = "A-Inversor   " + String(c_ina3);
828         PT = "P-Inversor   " + String(p_ina3);
829         WT = "Wh-Inversor " + String(e_ina3);
830         break;
831       }
832
833       display.clear();
834       display.setTextAlignment(TEXT_ALIGN_LEFT);
835       display.setFont(ArialMT_Plain_10);
836       display.drawString(1, 5, tempo);
837       display.drawString(1, 15, VB);
838       display.drawString(1, 25, AT);
839       display.drawString(1, 35, PT);
840       display.drawString(1, 45, WT);
```

```
841       display.display();
842
843       // Verifica se tem alertas
844       if (ad6 & ad7 & ad8)
845       {
846         if (ina1.isAlert())
847         {
848           Serial.print("\nALERTA!!! excesso de potencia - Tracer 1 - ");
849           Serial.print(pina1, 2);
850           Serial.print("W");
851           Serial2.print("<ALERTA-POL Tracer1>");
852         }
853         if (ina2.isAlert())
854         {
855           Serial.print("\nALERTA!!! excesso de potencia - Tracer 2");
856           Serial.print(pina1, 2);
857           Serial.print("W");
858           Serial2.print("<ALERTA-POL Tracer2>");
859         }
860         if (ina3.isAlert())
861         {
862           Serial.print("\nALERTA!!! Voltagem baixa no banco de baterias");
863           Serial.print(vina3, 2);
864           Serial.print("V");
865           Serial2.print("<ALERTA-BUL Baterias>");
866         }
867       }
868
869     //_____________________________________________________________________________________________________________
870
871     // Para verificar se a fotocelula está ativa e quais grupos de refletores devem ser ligados
872
873     noite = digitalRead(fotocelula);                          // Lê o pino da fotocélula
874
875     if ((noite == 1) && (!foinoite))                          // Iniciou a noite, liga
876     {
877       // Vamos ligar os grupos que tem o modo automático
878       liga_desliga_grupos_automaticos(1);
879       foinoite = true;
880     }
```

```
881     else if ((noite == 0) && foinoite)                      // Acabou a noite, desliga
882     {
883       // Vamos desligar os grupos que tem o modo automático
884       liga_desliga_grupos_automaticos(0);
885       foinoite = false;
886     }
887     else;                                                   // Continua noite ou continua dia
888
889     //_____________________________________________________________________________________________________________
890
891     // Para verificar horario de ligar ou desligar os grupos de refletores programados
892
893     liga_desliga_grupos_programados();
894
895     //_____________________________________________________________________________________________________________
896
897       timeElapsed1 = 0;                                     // Reset do contador
898     }
899
900   //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
901   //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
902   //@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
903   //_____________________________________ Ciclo de tempo timeElapsed2 - default 5 minutos _____________________________________
904
905     if (timeElapsed2 >= eeprom.inter2)                      // Executa quando completa um ciclo de duracao inter2
906     {                                                       // Default 5 minutos
907       temperatura = rtc.getTemperature();                   // Pega temperatura
908
909       // Vamos pegar máximos e mínimos da voltagem das INAs (no dia)
910       if (vina1 > vina1_max_dia) vina1_max_dia = vina1;
911       else if (vina1 < vina1_min_dia) vina1_min_dia = vina1;
912       else;
913       if (vina2 > vina2_max_dia) vina2_max_dia = vina2;
914       else if (vina2 < vina2_min_dia) vina2_min_dia = vina2;
915       else;
916       if (vina3 > vina3_max_dia) vina3_max_dia = vina3;
917       else if (vina3 < vina3_min_dia) vina3_min_dia = vina3;
918       else;
919
920       // Imprime data e hora atual
```

```
    Serial.println("\n");
    Serial.print(tempo);
    Serial.print("\nTemperatura: ");
    Serial.print(temperatura);
    Serial.print(" C°\n");
    Serial.print("\nEnergia Tracer1:   ");
    Serial.print(eeprom.etracer1, 2);
    Serial.print("Wh");
    Serial.print("\nEnergia Tracer2:   ");
    Serial.print(eeprom.etracer2, 2);
    Serial.print("Wh");
    Serial.print("\nEnergia Inversor:  ");
    Serial.print(eeprom.einversor, 2);
    Serial.print("Wh");
    Serial.print("\nEnergia acumulada: ");
    Serial.print(etotalin, 2);
    Serial.print("Wh");
    Serial.print("\nEnergia consumida: ");
    Serial.print(eeprom.einversor, 2);
    Serial.print("Wh");

    // Grava eeprom
    eeWrite(0,eeprom);
    delay (30);

    timeElapsed2 = 0;                                       // Reset do contador
  }
}

//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//______________________________________________________ Funções _______________________________________________________

void telainicial()                                          // Tela inicial do OLED
{
  str = "CoMoSisFog " + versao;
  display.clear();
  display.setTextAlignment(TEXT_ALIGN_CENTER);
  display.setFont(ArialMT_Plain_10);
```

```
961     display.drawString(63, 5, str);
962     display.drawString(63, 25, "ESP32");
963     display.drawString(63, 45, "Filippo Pardini");
964     display.display();
965   }
966
967   //_____________________________________________________________________________________________________________
968
969   void configura_INAs(unsigned char ch)
970   {
971     // Inicia INA1 endereco 0x40 - Tracer 1 - shunt 30A - 75mV
972     ina1.begin(0x40);
973     // Configura
974     ina1.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
975     INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
976     // Calibra INA226 - shunt = 0.0025 ohm, corrente maxima esperada = 20A
977     ina1.calibrate(0.0025, 20);
978
979     // Inicia INA2 endereco 0x41 - Tracer 2 - shunt 30A - 75mV
980     ina2.begin(0x41);
981     // Configura
982     ina2.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
983     INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
984     // Calibra INA226 - shunt = 0.0025 ohm, corrente maxima esperada = 20A
985     ina2.calibrate(0.0025, 20);
986
987     // Inicia INA3 endereco 0x44 - Inversor - shunt 50A - 75mV
988     ina3.begin(0x44);
989     // Configura
990     ina3.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
991     INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
992     // Calibra INA226 - shunt = 0.0015 ohm, corrente maxima esperada = 30A
993     ina3.calibrate(0.0015, 30);
994
995   /*
996     if (ch == 1)
997     {
998       // Display das configuracoes
999       checkConfig(1);
1000      checkConfig(2);
```

```
1001      checkConfig(3);
1002    }
1003  */
1004  }
1005
1006  //_____________________________________________________________________________________________________________
1007
1008  void saidas_altas_pcf1_pcf2(void)
1009  {
1010    Wire.beginTransmission(pcf1);                    // Inicia transmissão I2C para o Pcf1
1011    Wire.write(B11111111);                           // Grava novo byte
1012    Wire.endTransmission();                          // Encerra transmissão I2C
1013
1014    Wire.beginTransmission(pcf2);                    // Inicia transmissão I2C para o Pcf2
1015    Wire.write(B11111111);                           // Grava novo byte
1016    Wire.endTransmission();                          // Encerra transmissão I2C
1017  }
1018
1019  //_____________________________________________________________________________________________________________
1020
1021  bool toggle_grupo_refletores(uint8_t grp)
1022  {
1023    // Faz o toggle (inversão) de um grupo de refletores: se estava ligado -> desliga e se estava desligado -> liga
1024    // Nota: o grupo 0 por definição contém todos os refletores, portanto liga/desliga todos
1025
1026    int adr,ref,rf,m;
1027    byte atual,tmp;
1028
1029    // Vamos verificar se é um grupo ativo
1030    if (grupo_am[grp][0] != 1) return false;         // grupo inativo, retorna
1031
1032    // É ativo, faz o toggle dos refletores do grupo
1033    for (m=1;m<=grupo_rf[grp][0];m++)                // Varre os refletores do grupo
1034    {
1035      rf = grupo_rf[grp][m];                         // refletor (entre 1 e 12)
1036      if (rf <= 8) {adr = pcf1; ref = rf;}           // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1037      else {adr = pcf2; ref = rf - 8;}               // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1038
1039      Wire.requestFrom(adr,1);                       // Solicita 1 byte ao Pcf correto
1040      while (Wire.available() == 0);                 // Aguarda estar no buffer
```

```
1041      atual = Wire.read();                                  // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1042                                                            // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1043                                                            // -> refletor 12
1044
1045      tmp = atual;                                          // Temporario
1046      tmp <<= (8 - ref);                                    // Pega o bit do refletor
1047      tmp >>= 7;                                            // Pega o bit do refletor
1048      if (tmp == 0)                                         // Se refletor ligado -> desliga
1049      {
1050        atual |= 1 << (ref - 1);                            // Toggle do refletor rf
1051        refletor_ligado[rf] = false;                        // Marca como refletor desligado
1052        if (!tem_algo_ligado()) desliga_inversor();         // Se não tem nada ligado, desliga inversor
1053      }
1054      else                                                  // Se refletor desligado -> liga
1055      {
1056        if (!inversor_ligado) liga_inversor();              // Se inversor desligado, liga inversor
1057        atual &= ~(1 << (ref - 1));                         // Toggle do refletor rf
1058        refletor_ligado[rf] = false;                        // Marca como refletor desligado
1059      }
1060
1061      // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1062      Wire.beginTransmission(adr);                          // Inicia transmissão I2C para o Pcf correto
1063      Wire.write(atual);                                    // Novo byte de acionamento dos refletores do grupo
1064      Wire.endTransmission();                               // Encerra transmissão I2C
1065    }
1066    if (grupo_am[grp][2] == 0) grupo_am[grp][2] = 1;        // Marca grupo como ligado
1067    else grupo_am[grp][2] = 0;                              // Marca grupo como desligado
1068    return true;
1069  }
1070
1071  //_____________________________________________________________________________________________________________
1072
1073  void inicializa_grupos_refletores(void)
1074  {
1075    // Inicializa todos os grupos de refletores e refletores como desligados
1076    uint8_t m;
1077
1078    Wire.beginTransmission(pcf1);
1079    Wire.write(B11111111);                                  // Os relés ligam com 0, então para desligar 1
1080    Wire.endTransmission();
```

```
1081    Wire.beginTransmission(pcf2);
1082    Wire.write(B11111111);                                  // Os relés ligam com 0, então para desligar 1
1083    Wire.endTransmission();
1084
1085    for (m=1;m<=12;m++) refletor_ligado[m] = false;
1086  }
1087
1088  //_____________________________________________________________________________________________________________
1089
1090  void limpa_buffer_serial(uint8_t cmp)
1091  {
1092    uint8_t i = 1;
1093
1094    while (i <= cmp)
1095    {
1096      Serial2.read();
1097      i++;
1098    }
1099    delay(1000);
1100  }
1101
1102  //_____________________________________________________________________________________________________________
1103
1104  void liga_desliga_grupos_automaticos(uint8_t ld)
1105  {
1106    int adr,ref,rf,n,m;
1107    byte atual;
1108
1109    for (n=1;n<=12;n++)                                     // Varre os grupos
1110    {
1111      if ((grupo_am[n][0] == 1) && (grupo_am[n][1] == 0))   // Grupo ativo e automático, liga ou desliga (ld)
1112      {
1113        for (m=1;m<=grupo_rf[n][0];m++)                     // Varre os refletores do grupo
1114        {
1115          rf = grupo_rf[n][m];                              // refletor (entre 1 e 12)
1116          if (rf <= 8) {adr = pcf1; ref = rf;}              // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1117          else {adr = pcf2; ref = rf - 8;}                  // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1118
1119          Wire.requestFrom(adr,1);                          // Solicita 1 byte ao Pcf correto
1120          while (Wire.available() == 0);                    // Aguarda estar no buffer
```

```
1121          atual = Wire.read();                                  // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1122                                                                // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1123                                                                // -> refletor 12
1124          if (ld == 1)                                          // Liga refletor do grupo
1125          {
1126            atual &= ~(1 << (ref - 1));                         // ld = 1 => liga ( bit do pcf = 0)
1127            refletor_ligado[rf] = true;                         // Marca como refletor ligado
1128          }
1129          else                                                  // Desliga refletores do grupo
1130          {
1131            atual |= 1 << (ref - 1);                            // ld = 0 => desliga ( bit do pcf = 1)
1132            refletor_ligado[rf] = false;                        // Marca como refletor desligado
1133          }
1134
1135          // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1136          Wire.beginTransmission(adr);                          // Inicia transmissão I2C para o Pcf correto
1137          Wire.write(atual);                                    // Novo byte de acionamento dos refletores do grupo
1138          Wire.endTransmission();                               // Encerra transmissão I2C
1139        }
1140        if (ld == 1) grupo_am[n][2] = 1;                        // Marca grupo como ligado
1141        else grupo_am[n][2] = 0;                                // Marca grupo como desligado
1142      }
1143    }
1144  }
1145
1146  //_____________________________________________________________________________________________________________________
1147
1148  void liga_desliga_grupos_programados(void)
1149  {
1150    int adr,ref,rf,n,m,ld;
1151    byte atual;
1152    float agora;
1153
1154    DateTime now = rtc.now();
1155    agora = (float)now.hour() + ((float)now.minute() / 100.0);
1156
1157    for (n=1;n<=12;n++)                                         // Varre os grupos
1158    {
1159      if ((grupo_am[n][0] == 1) && (grupo_am[n][1] == 1))       // Grupo ativo e programado
1160      {
```

```
1161        // Vamos verificar se está na hora de ligar ou desligar grupo
1162        if ((agora >= grupo_if[n][0]) && (grupo_am[n][2] == 0)) ld = 1; // Hora de ligar
1163        else if ((agora >= grupo_if[n][1]) && (agora < grupo_if[n][0]) && (grupo_am[n][2] == 1)) ld = 0;  // Hora de
1164                                                                                                          // desligar
1165        else;
1166
1167        for (m=1;m<=grupo_rf[n][0];m++)                    // Varre os refletores do grupo
1168        {
1169          rf = grupo_rf[n][m];                             // refletor (entre 1 e 12)
1170          if (rf <= 8) {adr = pcf1; ref = rf;}             // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1171          else {adr = pcf2; ref = rf - 8;}                 // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1172
1173          Wire.requestFrom(adr,1);                         // Solicita 1 byte ao Pcf correto
1174          while (Wire.available() == 0);                   // Aguarda estar no buffer
1175          atual = Wire.read();                             // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1176                                                           // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1177                                                           // -> refletor 12
1178          if (ld == 1)                                     // Liga refletor do grupo
1179          {
1180            atual &= ~(1 << (ref - 1));                    // ld = 1 => liga (bit = 0 no pcf)
1181            refletor_ligado[rf] = true;                    // Marca como refletor ligado
1182          }
1183          else                                             // Desliga refletores do grupo
1184          {
1185            atual |= 1 << (ref - 1);                       // ld = 0 => desliga (bit = 1 no pcf)
1186            refletor_ligado[rf] = false;                   // Marca como refletor desligado
1187          }
1188
1189          // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1190          Wire.beginTransmission(adr);                     // Inicia transmissão I2C para o Pcf correto
1191          Wire.write(atual);                               // Novo byte de acionamento dos refletores do grupo
1192          Wire.endTransmission();                          // Encerra transmissão I2C
1193        }
1194        if (ld == 1) grupo_am[n][2] = 1;                   // Marca grupo como ligado
1195        else grupo_am[n][2] = 0;                           // Marca grupo como desligado
1196      }
1197    }
1198  }
1199
1200  //_____________________________________________________________________________________________________________
```

```
1201
1202  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano)
1203  {
1204    unsigned int an,nd = 0;
1205    unsigned int dd[14] = {0,0,31,59,90,120,151,181,212,243,273,304,334,365};
1206
1207    for (an=2019;an<ano;an++)
1208    {
1209      if ((an % 4) == 0) nd += 366;                              // É ano bissexto (vale até 2100)
1210      else nd += 365;
1211    }
1212    if (((an % 4) == 0) && (mes > 2)) nd += (dd[mes] + dia + 1);  // É ano bissexto (vale até 2100)
1213    else nd += (dd[mes] + dia);
1214    return nd;
1215  }
1216
1217  //_____________________________________________________________________________________________________________
1218
1219  bool SD_readStringUntil(const char car)
1220  {
1221    // Lê o string de caracteres até o primeiro caractere car encontrado inclusive
1222
1223    str = "";
1224    char cc;
1225
1226    while(1)
1227    {
1228      cc = arquivo.read();
1229      if ((uint8_t)cc == -1) return false;
1230      if (cc != car) str += cc;
1231      else
1232      {
1233        str += cc;
1234        return true;
1235      }
1236    }
1237  }
1238
1239  //_____________________________________________________________________________________________________________
1240
```

```
1241  bool tem_algo_ligado(void)
1242  {
1243    uint8_t k;
1244    bool ret = false;
1245
1246    for (k=1;k<=12;k++) ret |= refletor_ligado[k];
1247    return ret;
1248  }
1249
1250  //_____________________________________________________________________________________________________________
1251
1252  void liga_inversor(void)
1253  {
1254    // O inversor tem que estar ligado no pino P7 (bit 7)do pcf2 através de um relé externo
1255    byte este;
1256
1257    Wire.requestFrom(pcf2,1);                                  // Solicita 1 byte ao Pcf2
1258    while (Wire.available() == 0);                             // Aguarda estar no buffer
1259    este = Wire.read();                                        // Lê o byte, bit 7 -> inversor
1260    este &= ~(1 << 7);                                         // Liga (bit 7 = 0)
1261    // Atualiza Pcf2 para ligar inversor
1262    Wire.beginTransmission(pcf2);                              // Inicia transmissão I2C para o Pcf2
1263    Wire.write(este);                                          // Novo byte de acionamento
1264    Wire.endTransmission();                                    // Encerra transmissão I2C
1265    inversor_ligado = true;
1266  }
1267
1268  //_____________________________________________________________________________________________________________
1269
1270  void desliga_inversor(void)
1271  {
1272    // O inversor tem que estar ligado no pino P7 (bit 7)do pcf2 através de um relé externo
1273    byte este;
1274
1275    Wire.requestFrom(pcf2,1);                                  // Solicita 1 byte ao Pcf2
1276    while (Wire.available() == 0);                             // Aguarda estar no buffer
1277    este = Wire.read();                                        // Lê o byte, bit 7 -> inversor
1278    este |= 1 << 7;                                            // Desliga (bit 7 = 1)
1279    // Atualiza Pcf2 para desligar inversor
1280    Wire.beginTransmission(pcf2);                              // Inicia transmissão I2C para o Pcf correto
```

```
1281    Wire.write(este);                                   // Novo byte de acionamento
1282    Wire.endTransmission();                             // Encerra transmissão I2C
1283    inversor_ligado = false;
1284  }
1285
1286  //_____________________________________________________________________________________________________________
1287
1288  bool ppe(uint8_t ncar,unsigned long intervalo,uint8_t deonde)
1289  {
1290    // Pedido para envio, verifica espaço no buffer
1291    timeElapsed = 0;
1292    while(timeElapsed < intervalo)                      // Executa enquanto não passar o intervalo
1293    {
1294      if (Serial2.available() < ncar);
1295      else return true;
1296    }
1297    Serial.print("\nTime Out esperando Serial.Xbee para enviar,");
1298    Serial.print(" chamada de ");
1299    Serial.print(deonde);
1300    return false;
1301  }
1302
1303  //_____________________________________________________________________________________________________________
1304
```