```
/*****************************************************************************************************************************
* This program is free software: you can redistribute it and/or modify it under the terms of the GNU Lesser General    *
* Public License as published by the Free Software Foundation, either version 3 of the License, or any later version.  *
* This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied   *
* warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU Lesser General Public License for more  *
* details. You should have received a copy of the GNU Lesser General Public License along with this program.           *
* If not, see <http://www.gnu.org/licenses/>.                                                                           *
*                                                                                                                       *
* © Copyright 2019-2020 Filippo Pardini - filippo@robotica.eng.br - http://blog.robotica.eng.br/                        *
*****************************************************************************************************************************/
/*
  Projeto CoMoSisFog V4.0.0 (23/10/2020)
  * Projeto para Controle e Monitoramento de um Sistema Fotovoltaico off grid para iluminacao de um jardim

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

  Parte 1 - Quiosque (onde estão os paineis solares)

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

  Parte 2 - Laboratorio (na residência)

  ESP32 remoto - controlador
  * Esta parte utiliza uma placa DOIT ESP32 DEVKIT V1, uma eeprom At24c256 - 32KB, um Nanoshield ADC circuitar, um
```

```
47    * display LCD03 com keypad, um buzzer, dois botões NA, um Xbee, um PC/keyboard/mouse, uma fonte 5Vdc com bateria
48    * ion-litio 3S e um celular. Ela atua como servidor AP para comunicação com o celular (o qual envia comandos
49    * para acionar os grupos de refletores) e como entrada de configurações, solicitações e comandos, via PC, para a
50    * Parte 1 via Xbee. Além disso, faz o display, no LCD03, dos dados enviados pela parte 1, comanda os grupos de
51    * refletores pelo keypad e envia os dados para a nuvem via MQTT (cloudMQTT) e ThingSpeak (para analise com MATLAB).
52  */
53
54  //*************************************************************************************************************************
55  //*************************************** Este é o sketch da Parte 1 ******************************************************
56  //************************************************************************************************************************* Bibliotecas
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  //_____________________________________________________________________________________________________________________ Pinos I2C
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  //_____________________________________________________________________________________________________________________ template
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 //_____________________________________________________________________________________________________ Instâncias
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  //_____________________________________________________________________________________________________ Variáveis e constantes
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 = "V4.0.0";                          // ************* 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;
```

```
unsigned int inicio,fim,datseq;
File arquivo;

float vina1_max_dia;                                        // Voltagem máxima INA1 no dia
float vina1_min_dia;                                        // Voltagem mínima INA1 no dia
float vina2_max_dia;                                        // Voltagem máxima INA2 no dia
float vina2_min_dia;                                        // Voltagem mínima INA2 no dia
float vina3_max_dia;                                        // Voltagem máxima INA3 no dia
float vina3_min_dia;                                        // Voltagem mínima INA3 no dia

int tipo;
char comando;
int codigo;
long int parametros[13];
int numero_parametros;

//______________________________________________________________________________________________________________________ Declaração Funções

// Funções
void telainicial();
void configura_INAs(unsigned char ch);
void saidas_altas_pcf1_pcf2(void);
bool toggle_grupo_refletores(long int grp);
void desliga_todos_refletores(void);
void liga_todos_refletores(void);
void limpa_buffer_serial(uint8_t cmp);
bool ppe(uint8_t ncar,unsigned long intervalo,uint8_t deonde);
void liga_desliga_grupos_automaticos(uint8_t ld);
void liga_desliga_grupos_programados(void);
unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano);
bool SD_readStringUntil(const char car);
bool tem_algo_ligado(void);
void liga_inversor(void);
void desliga_inversor(void);
bool toggle_refletor(long int rf);
bool recebe_comando(int *tip,char *cmd,int *cod,int *numpar,long int *par);

//______________________________________________________________________________________________________________________
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@@
//_________________________________________________ Processa uma vez ___________________________________________________ setup

void setup()
{
  byte error,addr;
```

```
231     int nDevices;
232
233     // Definição pinos ____________________________________________________________________________________
234     pinMode(fotocelula, INPUT_PULLUP);                          // Modo do pino fotocelula
235     pinMode(inputPin, INPUT_PULLUP);                            // Modo do inputPin para eeprom default
236
237     // Inicia monitor serial ______________________________________________________________________________
238     Serial.begin(115200);                                       // Define velocidade
239     while (!Serial);                                            // aguarda
240
241     // Inicia I2C _________________________________________________________________________________________
242     Wire.begin(SDApin,SCLpin);                                  // Define pinos I2C
243
244     // ____________________________________________ Inicio varredura I2C ____________________________________
245
246     Serial.println("\nVarredura I2C");
247     Serial.println("Procurando...\n");
248     nDevices = 0;
249     ad1=ad2=ad3=ad4=ad5=ad6=ad7=ad8 = false;
250
251     for(addr = 1;addr < 127;addr++)
252     {
253       Wire.beginTransmission(addr);
254       error = Wire.endTransmission();
255       if (error == 0)
256       {
257         switch (addr)
258         {
259           case 0x57:
260             Serial.print("Encontrada eeprom 24C32 no endereço 0x57\n");
261             ad1 = true;
262             break;
263           case 0x20:
264             Serial.print("Encontrado PCF8574_1 no endereço 0x20\n");
265             ad2 = true;
266             break;
267           case 0x21:
268             Serial.print("Encontrado PCF8574_2 no endereço 0x21\n");
269             ad3 = true;
270             break;
271           case 0x3C:
272             Serial.print("Encontrado OLED no endereço 0x3C\n");
273             ad4 = true;
274             break;
275           case 0x68:
276             Serial.print("Encontrado RTC DS3231M no endereço 0x68\n");
```

```
277             ad5 = true;
278             break;
279           case 0x40:
280             Serial.print("Encontrado INA1 Tracer1 no endereço 0x40\n");
281             ad6 = true;
282             break;
283           case 0x41:
284             Serial.print("Encontrado INA2 Tracer2 no endereço 0x41\n");
285             ad7 = true;
286             break;
287           case 0x44:
288             Serial.print("Encontrado INA3 Inversor no endereço 0x44\n");
289             ad8 = true;
290             break;
291           default:
292             Serial.print("Endereço desconhecido\n");
293         }
294         nDevices++;
295       }
296       else if (error==4)
297       {
298         Serial.print("Erro desconhecido no endereço 0x");
299         if (addr<16) Serial.print("0");
300         Serial.println(addr,HEX);
301       }
302     }
303     if (nDevices == 0) Serial.println("Nenhum equipamento encontrado\n");
304     if (nDevices == 8) Serial.println("Todos os equipamentos presentes\n");
305
306     // ____________________________________________ Fim varredura I2C ____________________________________________
307
308     // Inicia Xbee ____________________________________________________________________________________________
309     Serial2.begin(9600, SERIAL_8N1, RXpin, TXpin);
310
311     // Inicia cartão SD _______________________________________________________________________________________
312     if(!SD.begin())
313     {
314       Serial.print("\nMontagem do cartao falhou");            // Montagem do cartão falhou
315       Serial2.print("<Montagem SD falhou>");                  // Envia para P2 para display no LCD03
316       while(1);
317     }
318     uint8_t cardType = SD.cardType();                         // Tipo do cartão
319     if(cardType == CARD_NONE){
320       Serial.print("\nSem SD inserido");                     // Sem cartão
321       Serial2.print("<Sem SD inserido>");                     // Envia para P2 para display no LCD03
322       while(1);
```

```
323     }
324     Serial.print("\nSD inicializado");                        // Tipo do cartão SD
325     Serial.print("\nSD tipo: ");
326     if(cardType == CARD_MMC) Serial.print("MMC");
327     else if(cardType == CARD_SD) Serial.print("SDSC");
328     else if(cardType == CARD_SDHC) Serial.print("SDHC");
329     else Serial.print("DESCONHECIDO");
330
331     uint64_t cardSize = SD.cardSize() / (1024 * 1024);        // Tamanho do cartão SD
332     Serial.printf("\nSD tamanho: %lluMB", cardSize);
333
334     //*********************************************************************************************************************
335     //SD.remove("/dados.txt");                                 // *******Só quando precisa reinicializar /dados.txt********
336     //*********************************************************************************************************************
337
338     arquivo = SD.open("/dados.txt", FILE_APPEND);             // Abre o arquivo para append
339     if(!arquivo)
340     {
341       Serial.print("\nFalha na abertura do arquivo");
342       Serial2.print("<Falha no arquivo (1)>");                // Envia para P2 para display no LCD03
343       while(1);
344     }
345
346     // Inicia display Oled ____________________________________________________________________________________________
347     display.init();
348
349     // Inicia rtc
350     if (!rtc.begin())
351     {
352       Serial.print("\nRTC inexistente");
353       while (1);
354     }
355
356     //rtc.adjust(DateTime(F(__DATE__), F(__TIME__)));           // Só para acertar o RTC, depois de carregar e executar,
357                                                               // comentar (//) e recarregar
358
359     // Inicializa os grupos de refletores _____________________________________________________________________________
360     desliga_todos_refletores();                               // Inicializa os PCF8574 com as saidas 0 (grupos desligados)
361
362     // Inicializa INAs 226 ____________________________________________________________________________________________
363     if (ad6 & ad7 & ad8)
364     {
365       configura_INAs(1);                                      // Configura INAs 1 - imprime dados 0 - nao imprime
366       ina1.enableOverPowerLimitAlert();                       // Habilita alerta de OverPower INA1 - Tracer1
367       ina2.enableOverPowerLimitAlert();                       // Habilita alerta de OverPower INA2 - Tracer2
368       ina3.enableBusUnderLimitAlert();                        // Habilita alerta de UnderLimitVoltage INA3 - Inversor
```

```
369       ina1.setPowerLimit(1084.0);                          // Configura potência limite INA1 - Tracer1
370       ina2.setPowerLimit(1084.0);                          // Configura potência limite INA2 - Tracer2
371       ina3.setBusVoltageLimit(22.2);                       // Configura Voltagem limite INA3 - Inversor
372     }
373
374     // Inicializa pcf1 e pcf2 com as saidas altas para que os relés iniciem desligados ____________________________________
375     saidas_altas_pcf1_pcf2();                              // Coloca as saidas dos pcf1 e pcf2 altas porque elas
376                                                            // iniciam baixas e os reles são ligados com baixa.
377                                                            // Inicialmente portanto, como precisamos que os relés
378                                                            // inciem desligados, temos que colocar as saidas
379                                                            // iniciais dos pcf1 e pcf2 altas
380
381     // Display da tela inicial no OLED ____________________________________________________________________________________
382     telainicial();
383
384     // Le eeprom __________________________________________________________________________________________________________
385     eeRead(0,eeprom);
386     delay(30);
387
388     // Se eeprom não válida ou reinicialização forçada via inputPin - define eeprom default
389     if ((eeprom.valido != key) || (digitalRead(inputPin) == LOW))
390     {
391       eeprom.etracer1 = 0.0;
392       eeprom.etracer2 = 0.0;
393       eeprom.einversor = 0.0;
394       eeprom.inter1 = 60000;                               // 1 minuto
395       eeprom.inter2 = 300000;                              // 5 minutos
396       eeprom.cod_mon = 1;                                  // Habilita/desabilita monitor serial (não usado no momento)
397       eeprom.valido = key;                                 // Valida a eeprom
398
399       eeWrite(0,eeprom);                                   // Grava eeprom
400       delay (30);
401
402       Serial.print("\neeprom default");
403       Serial.print("\ndeixar o pino inputPin aberto");
404     }
405     else  Serial.print("\neeprom valida");                  // Eeprom valida
406
407     coef1 = eeprom.inter1 / 3600000.0;                     // Para calculo da energia (Wh)
408
409     Serial.print("\nconfiguracao na eeprom");
410     Serial.print("\ninterval1    " + String(eeprom.inter1));
411     Serial.print("\ninterval2    " + String(eeprom.inter2));
412     Serial.print("\ncod_monitor  " + String(eeprom.cod_mon));
413
414     // 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 _________________________________________________ loop

void loop()
{
  int i,k,nn;

  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)
    timeElapsed24h = 0;
    etotalin_24h_inicial = etotalin;                          // Energia total de entrada até agora (05h)
    etotalout_24h_inicial = eeprom.einversor;                 // Energia total de saída até agora (05h)
    pula = true;
  }

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

```
461
462   //_____________________________________________________________________________________________________ recebe_comando
463
464   // Verifica se P2 enviou alguma coisa: comando, coniguração p1, configuração dos grupos de refletores, configuração
465   // de seus modos ou solicitação de dados
466   // Indicadores válidos:
467   // '>' - Comando para ligar todos os refletores - formato: >
468   // '<' - Comando para desligar todos os refletores - formato: <
469   // '%' - Comando toggle direto de refletores - formato - %(r) - onde r é o número do refletor (1 a 12)
470   //       %(0) -> solicitação de quais os refletores ligados
471   // '!' - Comando toggle dos grupos de refletores - formato - !(g) - onde g é o número do grupo. g = 0 => tudo
472   // '$' - Solicitação de dados - formato - $(i) - onde i=1 data-hora, i=2 dados INAs e energias, i=3 status grupos
473   //       e refletores
474   // '@' - Solicitação de dados - formato - @(di,df) - onde di - data de início em dias desde 01/01/2019, df - data
475   //       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
476   //       os dados relativos a esse período
477   // '#' - Configuração de ciclos em p1 - formato - #n(v) - onde n=1 interval1_p1, n=2 interval2_p1,
478   //       n=3 cod_monitor_p1, v = 0 ou 1 ou valor em minutos
479   // '&' - Configuração dos grupos de refletores - formato - &n(r,...,r) - onde n é o número do grupo
480   //       e r o número do refletor. grupo_rf[grupo][13] vai conter o número de refletores registrados no grupo.
481   //       O grupo_rf[grupo][0] contém o número de refletores do grupo. O grupo_rf[0][refletores]tem que ser definido
482   //       com todos os refletores
483   // '=' - Configuração dos horarios liga/desliga dos grupos e modo de operação de cada grupo
484   //       grupo_am[grupo][0] - modo de operação: ativo -> sim(1)/não(0), grupo_am[grupo][1] - automatico(0) ->
485   //       fotocelula, programado(1) -> hora liga/desliga, formato - =n(a,b,c,d,e,f) - onde n é o número do grupo
486   //       a=0 -> inativo, a=1 -> ativo, b=0 -> automático(fotocelula), b=1 ->programado(c,d,e,f só existem neste caso),
487   //       c -> hora liga d -> minutos liga e -> hora desliga d -> minutos desliga
488
489   if(recebe_comando(&tipo,&comando,&codigo,&numero_parametros,parametros))
490   {
491     // Tipos de comandos:
492     // Tipo 1 - 1 caractere - ex: &
493     // Tipo 2 - 1 caractere seguido de um inteiro - ex: &32
494     // Tipo 3 - 1 caractere seguido de '(' seguido de vários inteiros separados por virgula e encerrado por ')'
495     //          ex: &(23,350,5)
496     // Tipo 4 - 1 caractere seguido de um inteiro, seguido de '(' seguido de vários inteiros separados por virgula e
497     //          encerrado por ')' - ex: &32(23,350,5)
498
499     switch (tipo)
500     {
501       case 1:                                          // É comando tipo 1
502         Serial.print("\n");
503         Serial.print(comando);
504         switch (comando)
505         {
506           case '>':                                    // É comando +
```

```
507             // Liga todos os refletores
508             liga_todos_refletores();
509             Serial.print("\nTodos os refletores ligados");
510             break;
511           case '<':                                     // É comando !
512             // Desliga todos os refletores
513             desliga_todos_refletores();
514             Serial.print("\nTodos os refletores desligados");
515             break;
516         }
517         ACK;
518         break;
519       case 2:                                           // É comando tipo 2
520         Serial.print("\n");
521         Serial.print(comando); Serial.print(codigo);
522         break;
523       case 3:                                           // É comando tipo 3
524         Serial.print("\n");
525         Serial.print(comando); Serial.print('(');
526         for (i=1;i<=numero_parametros;i++)
527         {
528           Serial.print(parametros[i]);
529           if (i == numero_parametros) Serial.print(')');
530           else Serial.print(',');
531         }
532         switch (comando)
533         {
534           case '%':                                     // É comando %
535             nn = 0;
536             if (parametros[1] == 0)                     // Está solicitando quais os refletores ligados
537             {
538               str = "<";
539               for (k=1;k<=12;k++)
540               {
541                 if (refletor_ligado[k])
542                 {
543                   if (nn == 0);
544                   else str += String(',');
545                   str += String(k);
546                   nn++;
547                 }
548                 else;
549               }
550               if (nn >= 1)
551               {
552                 str += String('>');
```

```
553                 Serial2.print(str.c_str());                 // Envia para P2 para display no LCD03
554               }
555               else Serial2.print("<Nada ligado>");
556             }
557             else
558             {
559               for (k=1;k<=numero_parametros;k++)
560               {
561                 if (!toggle_refletor(parametros[k]))
562                 {
563                   Serial.print("\nToggle nao executado - ");
564                   Serial.print(parametros[k]);
565                 }
566                 else
567                 {
568                   Serial.print("\nToggle executado - ");
569                   Serial.print(parametros[k]);
570                 }
571               }
572             }
573             break;
574           case '!':                                           // É comando !
575             for (k=1;k<=numero_parametros;k++)
576             {
577               if (!toggle_grupo_refletores(parametros[k]))
578               {
579                 Serial.print("\nToggle nao executado, grupo inativo - ");
580                 Serial.print(parametros[k]);
581               }
582               else
583               {
584                 Serial.print("\nToggle executado - ");
585                 Serial.print(parametros[k]);
586               }
587             }
588             break;
589           case '$':                                           // É comando $
590             switch (parametros[1])
591             {
592               case 1:                                         // Solicitação de data/hora
593                 Serial2.print(tempo.c_str());
594                 break;
595               case 2:                                         // Solicitação de dados INAs
596                 Serial2.print("v1("); Serial2.print(vina1,2); Serial2.print("),");
597                 Serial2.print("c1("); Serial2.print(cina1,2); Serial2.print("),");
598                 Serial2.print("p1("); Serial2.print(pina1,2); Serial2.print("),");
```

```
                Serial2.print("v2("); Serial2.print(vina2,2); Serial2.print("),");
                Serial2.print("c2("); Serial2.print(cina2,2); Serial2.print("),");
                Serial2.print("p2("); Serial2.print(pina2,2); Serial2.print("),");
                Serial2.print("v3("); Serial2.print(vina3,2); Serial2.print("),");
                Serial2.print("c3("); Serial2.print(cina3,2); Serial2.print("),");
                Serial2.print("p3("); Serial2.print(pina3,2); Serial2.print("),");
                Serial2.print("ei("); Serial2.print(etotalin,2); Serial2.print("),");
                Serial2.print("eo("); Serial2.print(eeprom.einversor,2); Serial2.print(")");
                break;
              case 3:                                       // Solicitação do status de grupos e refletores
                for (j=0;j<=12;j++)                         // Composição dos grupos
                {
                  str = "\nGrupo " + String(j) + " => ";    // Prepara linha do grupo
                  for (jj=1;jj<=grupo_rf[j][0];jj++)        // Acrescenta os refletores
                  {
                    str += String(grupo_rf[j][jj]);
                    if (jj < grupo_rf[j][0]) str += ",";
                    if (jj == grupo_rf[j][0]) str += "#";   // Encerra grupo
                  }
                  Serial2.print(str.c_str());               // Envia grupo para P2
                }
                for (j=1;j<=12;j++)                         // Status dos refletores
                {
                  str = "\nRefletor " + String(j) + " => "; // Prepara linha do status
                  if (refletor_ligado[j]) str += "ligado#";
                  else str += "desligado#";
                  Serial2.print(str.c_str());               // Envia status para P2
                }
                break;
            }
            break;
          case '@':                                         // É comando @
            inicio = parametros[1];
            fim = parametros[2];
            if (inicio > fim)
            {
              Serial.print("\nData de inicio maior que data de fim");
              break;
            }
            arquivo.close();
            arquivo = SD.open("/dados.txt");
            if(!arquivo)
            {
              //Serial.print("\nFalha na abertura do arquivo");
              Serial2.print("<Falha no arquivo 2>");
              while(1);
```

```
645             }
646             else
647             {
648               if (!SD_readStringUntil('<'))                // Lê o primeiro registro
649               {
650                 //Serial.print("\nRegistro nao encontrado no SD - erro 1");
651                 Serial2.print("<Falha registro 1>");
652                 while(1);
653               }
654               // Encontrou o primeiro registro
655               datseq = arquivo.parseInt();                 // Vamos ler a datseq do primeiro registro
656
657               while (datseq < fim)                         // Vamos executar enquanto datseq < fim
658               {
659                 while (datseq < inicio)                    // Se o datseq é menor que o inicio,
660                 {
661                   if (!SD_readStringUntil('<'))            // vamos ler o arquivo até o próximo registro
662                   {
663                     //Serial.print("\nRegistro nao encontrado no SD - erro 2");
664                     Serial2.print("<Falha registro 2>");
665                     break;
666                   }
667                   datseq = arquivo.parseInt();             // até acharmos um registro com datseq >= inicio
668                 }
669
670                 // Achamos um, vamos imprimir
671                 if (!SD_readStringUntil('>'))
672                 {
673                   //Serial.print("\nFim de registro nao encontrado no SD - erro 3");
674                   Serial2.print("<Falha registro 3>");
675                   break;
676                 }
677                 str = '<' + String(datseq)+ str;
678                 Serial.print("\n" + str);
679                 if (!SD_readStringUntil('<'))              // ler o arquivo até o próximo registro
680                 {
681                   //Serial.print("\nRegistro nao encontrado no SD - erro 4");
682                   Serial2.print("<Falha registro 4>");
683                   break;
684                 }
685                 datseq = arquivo.parseInt();
686               }
687               if (datseq == fim)                           // Achamos o último, vamos enviar para Parte 2
688               {
689                 if (!SD_readStringUntil('>'))
690                 {
```

```
691                   //Serial.print("\nFim de registro nao encontrado no SD - erro 5");
692                   Serial2.print("<Falha registro 5>");
693                   while(1);
694                 }
695                 str = '<' + String(datseq) + str;
696                 Serial.print("\n" + str);
697               }
698               else;
699               arquivo.close();
700             }
701            break;
702         }
703         ACK;
704         break;
705       case 4:                                          // É comando tipo 4
706         Serial.print("\n");
707         Serial.print(comando); Serial.print(codigo); Serial.print('(');
708         for (i=1;i<=numero_parametros;i++)
709         {
710           Serial.print(parametros[i]);
711           if (i == numero_parametros) Serial.print(')');
712           else Serial.print(',');
713         }
714
715         switch (comando)
716         {
717           case '#':                                    // É comando #
718             eeRead(0,eeprom);                          // Le eeprom
719             delay(30);
720             switch (codigo)                            // Configuração de ciclos
721             {
722               case 1:                                  // É interval1_p1
723                 eeprom.inter1 = parametros[1];
724                 break;
725               case 2:                                  // É interval2_p1
726                 eeprom.inter2 = parametros[1];
727                 break;
728               case 3:
729                 eeprom.cod_mon = parametros[1];
730                 break;
731             }
732             eeWrite(0,eeprom);                         // Grava eeprom
733             delay (30);
734
735             // Avisa no display Oled
736             display.clear();
```

```
            display.setTextAlignment(TEXT_ALIGN_LEFT);
            display.setFont(ArialMT_Plain_10);
            display.drawString(1, 5, tempo);
            display.drawString(1, 15, "nova configuracao gravada");
            display.drawString(1, 25, "interval1       " + String(eeprom.inter1));
            display.drawString(1, 35, "interval2       " + String(eeprom.inter2));
            display.drawString(1, 45, "cod_monitor  " + String(eeprom.cod_mon));
            display.display();
            delay(5000);

            // Avisa no monitor serial
            Serial.print("\nnova configuracao gravada na eeprom");
            Serial.print("\ninterval1    " + String(eeprom.inter1));
            Serial.print("\ninterval2    " + String(eeprom.inter2));
            Serial.print("\ncod_monitor   " + String(eeprom.cod_mon));
            break;
          case '&':                                         // É comando &
            // Nota: o grupo 0 deve conter todos os refletores para comando total
            grupo = codigo;
            for (j=1;j<=numero_parametros;j++)
            {
              grupo_rf[grupo][j] = parametros[j];           // Armazena refletor
            }
            grupo_rf[grupo][0] = numero_parametros;         // Número de refletores no grupo
            break;
          case '=':                                         // É comando *
            grupo = codigo;
            grupo_am[grupo][0] = parametros[1];
            grupo_am[grupo][1] = parametros[2];
            if (grupo_am[grupo][1] == 1)                    // É programado
            {
              // Liga h.m (h - hora(0-24), m - minutos(0-60))
              grupo_if[grupo][0] = (float)parametros[3] + (float)parametros[4]/100.;
              // Desliga h.m (h - hora(0-24), m - minutos(0-60))
              grupo_if[grupo][1] = (float)parametros[5] + (float)parametros[6]/100.;
            }
            break;
        }
        ACK;                                                // Confirma OK para P2
        break;
    }
  }

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

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

  }

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

```
829  //_____________________________________ Ciclo de tempo timeElapsed1 - default 1 minuto _____________________________________ timeElapsed1
830
831    if (timeElapsed1 >= eeprom.inter1)                    // Executa a cada ciclo de duracao inter1
832    {                                                     // Default 1 minuto
833      // Lê os sensores INA226
834      if (ad6 & ad7 & ad8)
835      {
836        vina1 = ina1.readBusVoltage();                    // Voltagem INA1
837        cina1 = ina1.readShuntCurrent();                  // Corrente INA1
838        pina1 = ina1.readBusPower();                      // Potência INA1
839        delta1 = pina1 * coef1;                           // Delta energia
840        vina2 = ina2.readBusVoltage();                    // Voltagem INA2
841        cina2 = ina2.readShuntCurrent();                  // Corrente INA2
842        pina2 = ina2.readBusPower();                      // Potência INA2
843        delta2 = pina2 * coef1;                           // Delta energia
844        vina3 = ina3.readBusVoltage();                    // Voltagem INA3
845        cina3 = ina3.readShuntCurrent();                  // Corrente INA3
846        pina3 = ina3.readBusPower();                      // Potência INA3
847        delta3 = pina3 * coef1;                           // Delta energia
848        eeprom.etracer1 = eeprom.etracer1 + delta1;       // Integra energia
849        eeprom.etracer2 = eeprom.etracer2 + delta2;       // Integra energia
850        eeprom.einversor = eeprom.einversor + delta3;     // Integra energia
851        etotalin = eeprom.etracer1 + eeprom.etracer2;     // Integra energia total acumulada
852      }
853
854      // Imprime data e hora atual
855      Serial.println("\n");
856      Serial.print(tempo);
857      Serial.print("\n");
858      // Imprime leituras relativas ao Tracer1
859      Serial.print("\nINA226-1 Tracer1");
860      Serial.print("\n________________");
861      Serial.print("\nBus voltage:   ");
862      Serial.print(vina1, 2);
863      Serial.print("V");
864      Serial.print("\nShunt current: ");
865      Serial.print(cina1, 2);
866      Serial.print("A");
867      Serial.print("\nBus power:     ");
868      Serial.print(pina1, 2);
869      Serial.print("W");
870      Serial.print("\nBus energy:    ");
871      Serial.print(delta1, 2);
872      Serial.print("Wh");
873      // Imprime leituras relativas ao Tracer2
874      Serial.print("\n");
```

```
875       Serial.print("\nINA226-2 Tracer2");
876       Serial.print("\n_________________");
877       Serial.print("\nBus voltage:   ");
878       Serial.print(vina2, 2);
879       Serial.print("V");
880       Serial.print("\nShunt current: ");
881       Serial.print(cina2, 2);
882       Serial.print("A");
883       Serial.print("\nBus power:     ");
884       Serial.print(pina2, 2);
885       Serial.print("W");
886       Serial.print("\nBus energy:    ");
887       Serial.print(delta2, 2);
888       Serial.print("Wh");
889       // Imprime leituras relativas ao Inversor
890       Serial.print("\n");
891       Serial.print("\nINA226-3 Inversor");
892       Serial.print("\n_________________");
893       Serial.print("\nBus voltage:   ");
894       Serial.print(vina3, 2);
895       Serial.print("V");
896       Serial.print("\nShunt current: ");
897       Serial.print(cina3, 2);
898       Serial.print("A");
899       Serial.print("\nBus power:     ");
900       Serial.print(pina3, 2);
901       Serial.print("W");
902       Serial.print("\nBus energy:    ");
903       Serial.print(delta3, 2);
904       Serial.print("Wh");
905       Serial.print("\n");
906
907       //_____________________________________________________________________________________________________ OLED
908
909       // Para display na OLED
910       dtostrf(vina1, 5, 2, v_ina1);
911       dtostrf(cina1, 5, 2, c_ina1);
912       dtostrf(pina1, 5, 2, p_ina1);
913       dtostrf(eeprom.etracer1, 5, 2, e_ina1);
914
915       dtostrf(vina2, 5, 2, v_ina2);
916       dtostrf(cina2, 5, 2, c_ina2);
917       dtostrf(pina2, 5, 2, p_ina2);
918       dtostrf(eeprom.etracer2, 5, 2, e_ina2);
919
920       dtostrf(vina3, 5, 2, v_ina3);
```

```
921       dtostrf(cina3, 5, 2, c_ina3);
922       dtostrf(pina3, 5, 2, p_ina3);
923       dtostrf(eeprom.einversor, 5, 2, e_ina3);
924
925       ii++;
926       if (ii > 3) ii = 1;
927       switch (ii)
928       {
929         case 1:
930         VB = "V-Banco     " + String(v_ina1);
931         AT = "A-Tracer1   " + String(c_ina1);
932         PT = "P-Tracer1   " + String(p_ina1);
933         WT = "Wh-Tracer1 " + String(e_ina1);
934         break;
935         case 2:
936         VB = "V-Banco     " + String(v_ina2);
937         AT = "A-Tracer2   " + String(c_ina2);
938         PT = "P-Tracer2   " + String(p_ina2);
939         WT = "Wh-Tracer2 " + String(e_ina2);
940         break;
941         case 3:
942         VB = "V-Banco      " + String(v_ina3);
943         AT = "A-Inversor   " + String(c_ina3);
944         PT = "P-Inversor   " + String(p_ina3);
945         WT = "Wh-Inversor " + String(e_ina3);
946         break;
947       }
948
949       display.clear();
950       display.setTextAlignment(TEXT_ALIGN_LEFT);
951       display.setFont(ArialMT_Plain_10);
952       display.drawString(1, 5, tempo);
953       display.drawString(1, 15, VB);
954       display.drawString(1, 25, AT);
955       display.drawString(1, 35, PT);
956       display.drawString(1, 45, WT);
957       display.display();
958
959       // Verifica se tem alertas
960       if (ad6 & ad7 & ad8)
961       {
962         if (ina1.isAlert())
963         {
964           Serial.print("\nALERTA!!! excesso de potencia - Tracer 1 - ");
965           Serial.print(pina1, 2);
966           Serial.print("W");
```

```
967           Serial2.print("<ALERTA-POL Tracer1>");
968         }
969         if (ina2.isAlert())
970         {
971           Serial.print("\nALERTA!!! excesso de potencia - Tracer 2");
972           Serial.print(pina1, 2);
973           Serial.print("W");
974           Serial2.print("<ALERTA-POL Tracer2>");
975         }
976         if (ina3.isAlert())
977         {
978           Serial.print("\nALERTA!!! Voltagem baixa no banco de baterias");
979           Serial.print(vina3, 2);
980           Serial.print("V");
981           Serial2.print("<ALERTA-BUL Baterias>");
982         }
983       }
984
985     //_____________________________________________________________________________________________________
986
987     // Para verificar se a fotocelula está ativa e quais grupos de refletores devem ser ligados
988
989     noite = digitalRead(fotocelula);                          // Lê o pino da fotocélula
990
991     if ((noite == 1) && (!foinoite))                          // Iniciou a noite, liga
992     {
993       // Vamos ligar os grupos que tem o modo automático
994       liga_desliga_grupos_automaticos(1);
995       foinoite = true;
996     }
997     else if ((noite == 0) && foinoite)                        // Acabou a noite, desliga
998     {
999       // Vamos desligar os grupos que tem o modo automático
1000      liga_desliga_grupos_automaticos(0);
1001      foinoite = false;
1002    }
1003    else;                                                     // Continua noite ou continua dia
1004
1005    //_____________________________________________________________________________________________________
1006
1007    // Para verificar horario de ligar ou desligar os grupos de refletores programados
1008
1009    liga_desliga_grupos_programados();
1010
1011    //_____________________________________________________________________________________________________
1012
```

```
    timeElapsed1 = 0;                                       // Reset do contador
  }

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

  if (timeElapsed2 >= eeprom.inter2)                        // Executa quando completa um ciclo de duracao inter2
  {                                                         // Default 5 minutos
    temperatura = rtc.getTemperature();                     // Pega temperatura

    // Vamos pegar máximos e mínimos da voltagem das INAs (no dia)
    if (vina1 > vina1_max_dia) vina1_max_dia = vina1;
    else if (vina1 < vina1_min_dia) vina1_min_dia = vina1;
    else;
    if (vina2 > vina2_max_dia) vina2_max_dia = vina2;
    else if (vina2 < vina2_min_dia) vina2_min_dia = vina2;
    else;
    if (vina3 > vina3_max_dia) vina3_max_dia = vina3;
    else if (vina3 < vina3_min_dia) vina3_min_dia = vina3;
    else;

    // 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 _______________________________________________________ telainicial

void telainicial()                                        // Tela inicial do OLED
{
  str = "CoMoSisFog " + versao;
  display.clear();
  display.setTextAlignment(TEXT_ALIGN_CENTER);
  display.setFont(ArialMT_Plain_10);
  display.drawString(63, 5, str);
  display.drawString(63, 25, "ESP32");
  display.drawString(63, 45, "Filippo Pardini");
  display.display();
}

//______________________________________________________________________________________________________________________ configura_INAs

void configura_INAs(unsigned char ch)
{
  // Inicia INA1 endereco 0x40 - Tracer 1 - shunt 30A - 75mV
  ina1.begin(0x40);
  // Configura
  ina1.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
  INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
  // Calibra INA226 - shunt = 0.0025 ohm, corrente maxima esperada = 20A
  ina1.calibrate(0.0025, 20);

  // Inicia INA2 endereco 0x41 - Tracer 2 - shunt 30A - 75mV
  ina2.begin(0x41);
  // Configura
  ina2.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
  INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
  // Calibra INA226 - shunt = 0.0025 ohm, corrente maxima esperada = 20A
  ina2.calibrate(0.0025, 20);

  // Inicia INA3 endereco 0x44 - Inversor - shunt 50A - 75mV
  ina3.begin(0x44);
```

```
1105    // Configura
1106    ina3.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
1107    INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
1108    // Calibra INA226 - shunt = 0.0015 ohm, corrente maxima esperada = 30A
1109    ina3.calibrate(0.0015, 30);
1110
1111  /*
1112    if (ch == 1)
1113    {
1114      // Display das configuracoes
1115      checkConfig(1);
1116      checkConfig(2);
1117      checkConfig(3);
1118    }
1119  */
1120  }
1121
1122  //_____________________________________________________________________________________________________ saidas_altas_pcf1_pcf2
1123
1124  void saidas_altas_pcf1_pcf2(void)
1125  {
1126    Wire.beginTransmission(pcf1);                    // Inicia transmissão I2C para o Pcf1
1127    Wire.write(B11111111);                           // Grava novo byte
1128    Wire.endTransmission();                          // Encerra transmissão I2C
1129
1130    Wire.beginTransmission(pcf2);                    // Inicia transmissão I2C para o Pcf2
1131    Wire.write(B11111111);                           // Grava novo byte
1132    Wire.endTransmission();                          // Encerra transmissão I2C
1133  }
1134
1135  //_____________________________________________________________________________________________________ toggle_grupo_refletores
1136
1137  bool toggle_grupo_refletores(long int grp)
1138  {
1139    // Faz o toggle (inversão) de um grupo de refletores: se estava ligado -> desliga e se estava desligado -> liga
1140    // Nota: o grupo 0 por definição contém todos os refletores, portanto liga/desliga todos
1141
1142    int adr,ref,rf,m;
1143    byte atual,tmp;
1144
1145    // Vamos verificar se é um grupo ativo
1146    if (grupo_am[grp][0] != 1) return false;         // grupo inativo, retorna
1147
1148    // É ativo, faz o toggle dos refletores do grupo
1149    for (m=1;m<=grupo_rf[grp][0];m++)                // Varre os refletores do grupo
1150    {
```

```
1151      rf = grupo_rf[grp][m];                                  // refletor (entre 1 e 12)
1152      if (rf <= 8) {adr = pcf1; ref = rf;}                    // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1153      else {adr = pcf2; ref = rf - 8;}                        // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1154
1155      Wire.requestFrom(adr,1);                                // Solicita 1 byte ao Pcf correto
1156      while (Wire.available() == 0);                          // Aguarda estar no buffer
1157      atual = Wire.read();                                    // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1158                                                              // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1159                                                              // -> refletor 12
1160
1161      tmp = atual;                                            // Temporario
1162      tmp <<= (8 - ref);                                      // Pega o bit do refletor
1163      tmp >>= 7;                                              // Pega o bit do refletor
1164      if (tmp == 0)                                           // Se refletor ligado -> desliga
1165      {
1166        atual |= 1 << (ref - 1);                              // Toggle do refletor rf
1167        refletor_ligado[rf] = false;                          // Marca como refletor desligado
1168        if (!tem_algo_ligado()) desliga_inversor();           // Se não tem nada ligado, desliga inversor
1169      }
1170      else                                                    // Se refletor desligado -> liga
1171      {
1172        if (!inversor_ligado) liga_inversor();                // Se inversor desligado, liga inversor
1173        atual &= ~(1 << (ref - 1));                           // Toggle do refletor rf
1174        refletor_ligado[rf] = true;                           // Marca como refletor ligado
1175      }
1176
1177      // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1178      Wire.beginTransmission(adr);                            // Inicia transmissão I2C para o Pcf correto
1179      Wire.write(atual);                                      // Novo byte de acionamento dos refletores do grupo
1180      Wire.endTransmission();                                 // Encerra transmissão I2C
1181    }
1182    if (grupo_am[grp][2] == 0) grupo_am[grp][2] = 1;          // Marca grupo como ligado
1183    else grupo_am[grp][2] = 0;                                // Marca grupo como desligado
1184    return true;
1185  }
1186
1187  //_____________________________________________________________________________________________________________ desliga_todos_refletores
1188
1189  void desliga_todos_refletores(void)
1190  {
1191    // Inicializa todos os refletores como desligados
1192    uint8_t m;
1193
1194    Wire.beginTransmission(pcf1);
1195    Wire.write(B11111111);                                    // Os relés ligam com 0, então para desligar 1
1196    Wire.endTransmission();
```

```
1197    Wire.beginTransmission(pcf2);
1198    Wire.write(B11111111);                          // Os relés ligam com 0, então para desligar 1
1199    Wire.endTransmission();
1200
1201    for (m=1;m<=12;m++) refletor_ligado[m] = false;
1202    inversor_ligado = false;
1203  }
1204
1205  //_____________________________________________________________________________________________________ liga todos os refletores
1206
1207  void liga_todos_refletores(void)
1208  {
1209    // Inicializa todos os refletores como ligados
1210    uint8_t m;
1211
1212    Wire.beginTransmission(pcf1);
1213    Wire.write(B00000000);                          // Os relés ligam com 0, então para desligar 1
1214    Wire.endTransmission();
1215    Wire.beginTransmission(pcf2);
1216    Wire.write(B00000000);                          // Os relés ligam com 0, então para desligar 1
1217    Wire.endTransmission();
1218
1219    for (m=1;m<=12;m++) refletor_ligado[m] = true;
1220    inversor_ligado = true;
1221  }
1222
1223  //_____________________________________________________________________________________________________ limpa_buffer_serial
1224
1225  void limpa_buffer_serial(uint8_t cmp)
1226  {
1227    uint8_t i = 1;
1228
1229    while (i <= cmp)
1230    {
1231      Serial2.read();
1232      i++;
1233    }
1234    delay(1000);
1235  }
1236
1237  //_____________________________________________________________________________________________________ liga_desliga_grupos_automaticos
1238
1239  void liga_desliga_grupos_automaticos(uint8_t ld)
1240  {
1241    int adr,ref,rf,n,m;
1242    byte atual;
```

```
1243
1244    for (n=1;n<=12;n++)                                   // Varre os grupos
1245    {
1246      if ((grupo_am[n][0] == 1) && (grupo_am[n][1] == 0))  // Grupo ativo e automático, liga ou desliga (ld)
1247      {
1248        for (m=1;m<=grupo_rf[n][0];m++)                   // Varre os refletores do grupo
1249        {
1250          rf = grupo_rf[n][m];                            // refletor (entre 1 e 12)
1251          if (rf <= 8) {adr = pcf1; ref = rf;}            // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1252          else {adr = pcf2; ref = rf - 8;}                // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1253
1254          Wire.requestFrom(adr,1);                        // Solicita 1 byte ao Pcf correto
1255          while (Wire.available() == 0);                  // Aguarda estar no buffer
1256          atual = Wire.read();                            // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1257                                                          // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1258                                                          // -> refletor 12
1259          if (ld == 1)                                    // Liga refletor do grupo
1260          {
1261            atual &= ~(1 << (ref - 1));                   // ld = 1 => liga ( bit do pcf = 0)
1262            refletor_ligado[rf] = true;                   // Marca como refletor ligado
1263          }
1264          else                                            // Desliga refletores do grupo
1265          {
1266            atual |= 1 << (ref - 1);                      // ld = 0 => desliga ( bit do pcf = 1)
1267            refletor_ligado[rf] = false;                  // Marca como refletor desligado
1268          }
1269
1270          // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1271          Wire.beginTransmission(adr);                    // Inicia transmissão I2C para o Pcf correto
1272          Wire.write(atual);                              // Novo byte de acionamento dos refletores do grupo
1273          Wire.endTransmission();                         // Encerra transmissão I2C
1274        }
1275        if (ld == 1) grupo_am[n][2] = 1;                  // Marca grupo como ligado
1276        else grupo_am[n][2] = 0;                          // Marca grupo como desligado
1277      }
1278    }
1279  }
1280
1281  //_____________________________________________________________________________________________________ liga_desliga_grupos_programados
1282
1283  void liga_desliga_grupos_programados(void)
1284  {
1285    int adr,ref,rf,n,m,ld;
1286    byte atual;
1287    float agora;
1288
```

```
1289    DateTime now = rtc.now();
1290    agora = (float)now.hour() + ((float)now.minute() / 100.0);
1291
1292    for (n=1;n<=12;n++)                                    // Varre os grupos
1293    {
1294      if ((grupo_am[n][0] == 1) && (grupo_am[n][1] == 1))  // Grupo ativo e programado
1295      {
1296        // Vamos verificar se está na hora de ligar ou desligar grupo
1297        if ((agora >= grupo_if[n][0]) && (grupo_am[n][2] == 0)) ld = 1; // Hora de ligar
1298        else if ((agora >= grupo_if[n][1]) && (agora < grupo_if[n][0]) && (grupo_am[n][2] == 1)) ld = 0;  // Hora de
1299                                                                                                         // desligar
1300        else;
1301
1302        for (m=1;m<=grupo_rf[n][0];m++)                    // Varre os refletores do grupo
1303        {
1304          rf = grupo_rf[n][m];                             // refletor (entre 1 e 12)
1305          if (rf <= 8) {adr = pcf1; ref = rf;}             // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1306          else {adr = pcf2; ref = rf - 8;}                 // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1307
1308          Wire.requestFrom(adr,1);                         // Solicita 1 byte ao Pcf correto
1309          while (Wire.available() == 0);                   // Aguarda estar no buffer
1310          atual = Wire.read();                             // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1311                                                           // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1312                                                           // -> refletor 12
1313          if (ld == 1)                                     // Liga refletor do grupo
1314          {
1315            atual &= ~(1 << (ref - 1));                    // ld = 1 => liga (bit = 0 no pcf)
1316            refletor_ligado[rf] = true;                    // Marca como refletor ligado
1317          }
1318          else                                             // Desliga refletores do grupo
1319          {
1320            atual |= 1 << (ref - 1);                       // ld = 0 => desliga (bit = 1 no pcf)
1321            refletor_ligado[rf] = false;                   // Marca como refletor desligado
1322          }
1323
1324          // Atualiza Pcf para acionar refletor do grupo nesse Pcf
1325          Wire.beginTransmission(adr);                     // Inicia transmissão I2C para o Pcf correto
1326          Wire.write(atual);                               // Novo byte de acionamento dos refletores do grupo
1327          Wire.endTransmission();                          // Encerra transmissão I2C
1328        }
1329        if (ld == 1) grupo_am[n][2] = 1;                   // Marca grupo como ligado
1330        else grupo_am[n][2] = 0;                           // Marca grupo como desligado
1331      }
1332    }
1333  }
1334
```

```
1335  //_____________________________________________________________________________________________ dias_desde_01_01_2019
1336
1337  unsigned int dias_desde_01_01_2019(unsigned int dia,unsigned int mes,unsigned int ano)
1338  {
1339    unsigned int an,nd = 0;
1340    unsigned int dd[14] = {0,0,31,59,90,120,151,181,212,243,273,304,334,365};
1341
1342    for (an=2019;an<ano;an++)
1343    {
1344      if ((an % 4) == 0) nd += 366;                          // É ano bissexto (vale até 2100)
1345      else nd += 365;
1346    }
1347    if (((an % 4) == 0) && (mes > 2)) nd += (dd[mes] + dia + 1);  // É ano bissexto (vale até 2100)
1348    else nd += (dd[mes] + dia);
1349    return nd;
1350  }
1351
1352  //_____________________________________________________________________________________________ SD_readStringUntil
1353
1354  bool SD_readStringUntil(const char car)
1355  {
1356    // Lê o string de caracteres até o primeiro caractere car encontrado inclusive
1357
1358    str = "";
1359    char cc;
1360
1361    while(1)
1362    {
1363      cc = arquivo.read();
1364      if ((uint8_t)cc == -1) return false;
1365      if (cc != car) str += cc;
1366      else
1367      {
1368        str += cc;
1369        return true;
1370      }
1371    }
1372  }
1373
1374  //_____________________________________________________________________________________________ tem_algo_ligado
1375
1376  bool tem_algo_ligado(void)
1377  {
1378    uint8_t kk;
1379    bool ret = false;
1380
```

```
1381    for (kk=1;kk<=12;kk++) ret |= refletor_ligado[kk];
1382    return ret;
1383  }
1384
1385  //_____________________________________________________________________________________________________ liga_inversor
1386
1387  void liga_inversor(void)
1388  {
1389    // O inversor tem que estar ligado no pino P7 (bit 7)do pcf2 através de um relé externo
1390    byte este;
1391
1392    Wire.requestFrom(pcf2,1);                          // Solicita 1 byte ao Pcf2
1393    while (Wire.available() == 0);                     // Aguarda estar no buffer
1394    este = Wire.read();                                // Lê o byte, bit 7 -> inversor
1395    este &= ~(1 << 7);                                 // Liga (bit 7 = 0)
1396    // Atualiza Pcf2 para ligar inversor
1397    Wire.beginTransmission(pcf2);                      // Inicia transmissão I2C para o Pcf2
1398    Wire.write(este);                                  // Novo byte de acionamento
1399    Wire.endTransmission();                            // Encerra transmissão I2C
1400    inversor_ligado = true;
1401  }
1402
1403  //_____________________________________________________________________________________________________ desliga_inversor
1404
1405  void desliga_inversor(void)
1406  {
1407    // O inversor tem que estar ligado no pino P7 (bit 7)do pcf2 através de um relé externo
1408    byte este;
1409
1410    Wire.requestFrom(pcf2,1);                          // Solicita 1 byte ao Pcf2
1411    while (Wire.available() == 0);                     // Aguarda estar no buffer
1412    este = Wire.read();                                // Lê o byte, bit 7 -> inversor
1413    este |= 1 << 7;                                    // Desliga (bit 7 = 1)
1414    // Atualiza Pcf2 para desligar inversor
1415    Wire.beginTransmission(pcf2);                      // Inicia transmissão I2C para o Pcf correto
1416    Wire.write(este);                                  // Novo byte de acionamento
1417    Wire.endTransmission();                            // Encerra transmissão I2C
1418    inversor_ligado = false;
1419  }
1420
1421  //_____________________________________________________________________________________________________ ppe
1422
1423  bool ppe(uint8_t ncar,unsigned long intervalo,uint8_t deonde)
1424  {
1425    // Pedido para envio, verifica espaço no buffer
1426    timeElapsed = 0;
```

```
1427    while(timeElapsed < intervalo)                          // Executa enquanto não passar o intervalo
1428    {
1429      if (Serial2.available() < ncar);
1430      else return true;
1431    }
1432    Serial.print("\nTime Out esperando Serial.Xbee para enviar,");
1433    Serial.print(" chamada de ");
1434    Serial.print(deonde);
1435    return false;
1436  }
1437
1438  //_____________________________________________________________________________________________________ toggle_refletor
1439
1440  bool toggle_refletor(long int rf)
1441  {
1442    // Faz o toggle (inversão) de um refletor: se estava ligado -> desliga e se estava desligado -> liga
1443    // Para até 12 refletores (12 relés)
1444
1445    int adr,ref,m;
1446    byte atual,tmp;
1447
1448    if (rf <= 8) {adr = pcf1; ref = rf;}                    // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1449    else if (rf <= 12) {adr = pcf2; ref = rf - 8;}          // Calcula qual Pcf (1 ou 2) endereço adr e refletor ref
1450    else return false;
1451
1452    Wire.requestFrom(adr,1);                                // Solicita 1 byte ao Pcf correto
1453    while (Wire.available() == 0);                          // Aguarda estar no buffer
1454    atual = Wire.read();                                    // Lê o byte, bit 0 -> refletor 1 até bit 7 -> refletor
1455                                                            // 8 se for Pcf1. No Pcf2, bit 0 -> refletor 9 até bit 3
1456                                                            // -> refletor 12
1457
1458    tmp = atual;                                            // Temporario
1459    tmp <<= (8 - ref);                                      // Pega o bit do refletor
1460    tmp >>= 7;                                              // Pega o bit do refletor
1461    if (tmp == 0)                                           // Se refletor ligado -> desliga
1462    {
1463      atual |= 1 << (ref - 1);                              // Toggle do refletor rf
1464      refletor_ligado[rf] = false;                          // Marca como refletor desligado
1465      // Atualiza Pcf para acionar refletor nesse Pcf
1466      Wire.beginTransmission(adr);                          // Inicia transmissão I2C para o Pcf correto
1467      Wire.write(atual);                                    // Novo byte de acionamento dos refletores do grupo
1468      Wire.endTransmission();                               // Encerra transmissão I2C
1469      if (!tem_algo_ligado()) desliga_inversor();           // Se não tem nada ligado, desliga inversor
1470    }
1471    else                                                    // Se refletor desligado -> liga
1472    {
```

```
1473      atual &= ~(1 << (ref - 1));                          // Toggle do refletor rf
1474      refletor_ligado[rf] = true;                          // Marca como refletor ligado
1475      // Atualiza Pcf para acionar refletor nesse Pcf
1476      Wire.beginTransmission(adr);                         // Inicia transmissão I2C para o Pcf correto
1477      Wire.write(atual);                                   // Novo byte de acionamento dos refletores do grupo
1478      Wire.endTransmission();                              // Encerra transmissão I2C
1479      if (!inversor_ligado) liga_inversor();               // Se inversor desligado, liga inversor
1480    }
1481    return true;
1482  }
1483
1484  //_____________________________________________________________________________________________________ recebe_comando
1485
1486  bool recebe_comando(int *tip,char *cmd,int *cod,int *numpar,long int *par)
1487  {
1488    // Tipos de comandos:
1489    // Tipo 1 - 1 caractere - ex: &
1490    // Tipo 2 - 1 caractere seguido de um inteiro - ex: &32
1491    // Tipo 3 - 1 caractere seguido de '(' seguido de vários inteiros separados por virgula e encerrado por ')'
1492    //          ex: &(23,350,5)
1493    // Tipo 4 - 1 caractere seguido de um inteiro, seguido de '(' seguido de vários inteiros separados por virgula e
1494    //          encerrado por ')' - ex: &32(23,350,5)
1495
1496    int i;
1497    int numero_parametros = 0;
1498
1499    if (Serial2.available() > 0)
1500    {
1501      *cmd = Serial2.read();
1502      if (Serial2.peek() == -1)                            // É tipo 1
1503      {
1504        *tip = 1;
1505        return true;
1506      }
1507      else                                                 // Pode ser tipo 2, 3 ou 4
1508      {
1509        if (Serial2.peek() != '(')                         // Pode ser tipo 2 ou 4
1510        {
1511          *cod = Serial2.parseInt();
1512          if (Serial2.peek() == -1)                        // É tipo 2
1513          {
1514            *tip = 2;
1515            return true;
1516          }
1517          else                                             // É tipo 4
1518          {
```

```
1519            i = 1;
1520            while (Serial2.peek() != ')') par[i++] = Serial2.parseInt();
1521            Serial2.read();
1522            *numpar = i - 1;
1523            *tip = 4;
1524            return true;
1525          }
1526        }
1527        else                                                    // É tipo 3
1528        {
1529          i = 1;
1530          while (Serial2.peek() != ')') par[i++] = Serial2.parseInt();
1531          Serial2.read();
1532          *numpar = i - 1;
1533          *tip = 3;
1534          return true;
1535        }
1536      }
1537    }
1538    else return false;
1539  }
1540
1541  //_____________________________________________________________________________________________________________
1542
```