```
/*
  Projeto CoMoSisFog V1.1 (21/08/2019)
  * Projeto para Controle e Monitoramento de um Sistema Fotovoltaico off grid para iluminacao de um jardim
  Bloco - Quiosque
  * Este Bloco e composto de duas partes
  Parte 1 - ESP32
  * Esta parte utiliza uma placa ESP32 TTGO com display Oled com pilha recarregavel anexa e uma eeprom At24c256.
  * 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, enviando para a
  * nuvem via MQTT e ThingSpeak (para analise com MATLAB)e enviando via serial para o ATxmega128A1 (Parte 2) que faz
  * o armazenamento em SD. O Xmega, além disso, faz o controle das luminarias via remota através de outro Xmega com
  * comunicacao via Xbee.
*/

//_____________________________________________________________________________________________________________________

// Bibliotecas
#include <Wire.h>                                       // I2C
#include <INA226.h>                                     // INA226
#include "RTClib.h"                                     // DS3231M
#include <elapsedMillis.h>                              // Intervalos de tempo
#include "SSD1306Wire.h"                                // OLED
#include <HardwareSerial.h>                             // UART
#include <WiFi.h>                                       // WiFi
#include <PubSubClient.h>                               // MQTT
#include "ThingSpeak.h"                                 // ThingSpeak
#include <ArduinoJson.h>                                // Json

//_____________________________________________________________________________________________________________________

// Para display no monitor serial das configuracoes dos INA226 descomentar o comando abaixo e todos os checkConfig(n)
//#include "checkConfig.h"
// Configuracao atual:
// Mode - "Shunt and Bus, Continuous"
// Averages - "64 samples"
```

```
37  // BusConversionTime - "1.100ms"
38  // ShuntConversionTime - "1.100ms"
39  // Rshunt = 0.1 ohm
40  // Max excepted current = 0.8 A
41
42  //_____________________________________________________________________________________________________________
43
44  // eeprom
45  #define i2c_addr_eeprom 0x50                                    // Endereco I2C da 24LC256
46
47  template <class T> int eeWrite(int ee, const T& value)
48  {
49    const byte* p = (const byte*)(const void*)&value;
50    int i;
51    Wire.beginTransmission(i2c_addr_eeprom);
52    Wire.write((int)(ee >> 8)); // MSB
53    Wire.write((int)(ee & 0xFF)); // LSB
54    for (i = 0; i < sizeof(value); i++)
55    Wire.write(*p++);
56    Wire.endTransmission();
57    return i;
58  }
59
60  template <class T> int eeRead(int ee, T& value)
61  {
62    byte* p = (byte*)(void*)&value;
63    int i;
64    Wire.beginTransmission(i2c_addr_eeprom);
65    Wire.write((int)(ee >> 8)); // MSB
66    Wire.write((int)(ee & 0xFF)); // LSB
67    Wire.endTransmission();
68    Wire.requestFrom(i2c_addr_eeprom,sizeof(value));
69    for (i = 0; i < sizeof(value); i++)
70    if(Wire.available())
71    *p++ = Wire.read();
72    return i;
```

```
  }

  struct eeprom
  {
    unsigned long valido;
    float etracer1;
    float etracer2;
    float einversor;
    float eetotalin;
    float eetotalout;
    unsigned long inter1;
    unsigned long inter2;
    unsigned int cod_mon;
  } eeprom;

  //_____________________________________________________________________________________________________________________

  // WiFi
  const char* ssid = "Pardini";                                   // Nome da rede WiFi
  const char* password = "sitiolacicala";                         // Senha da rede WiFi

  //_____________________________________________________________________________________________________________________

  // MQTT
  const char* BROKER_MQTT = "m16.cloudmqtt.com";                  // URL do broker MQTT
  int BROKER_PORT = 17962;                                        // Porta do Broker MQTT
  #define ID_MQTT  "lrffixsc"                                     // User ID
  #define PSW_MQTT "JUp-9gCLKD2X"                                 // Password
  #define tracer1 "comosisfog/tracer1"                            // Topico tracer1
  #define tracer2 "comosisfog/tracer2"                            // Topico tracer2
  #define inversor "comosisfog/inversor"                          // Topico inversor
  #define etotin "comosisfog/etotin"                              // Topico energia total acumulada
  #define etotout "comosisfog/etotout"                            // Topico energia total consumida
  #define aviso1 "comosisfog/aviso1"                              // Topico aviso1
  #define aviso2 "comosisfog/aviso2"                              // Topico aviso2

```

```
109  //_____________________________________________________________________________________________________________________
110
111  // Instancias
112  WiFiClient wificlient;                                          // Instancia WiFi MQTT
113  WiFiClient wifiTS;                                              // Instancia WiFi ThingSpeak
114  PubSubClient MQTT(BROKER_MQTT,BROKER_PORT,wificlient);          // Instancia MQTT passando o objeto wificlient
115  HardwareSerial SerialXmega(2);                                  // Instancia Serial
116  elapsedMillis timeElapsed1;                                     // Instancia elapsedMillis
117  elapsedMillis timeElapsed2;                                     // Instancia elapsedMillis
118  INA226 ina1;                                                    // Instancia INA226 Tracer1
119  INA226 ina2;                                                    // Instancia INA226 Tracer2
120  INA226 ina3;                                                    // Instancia INA226 Inversor
121  RTC_DS3231 rtc;                                                 // Instancia RTC_DS3231
122
123  //_____________________________________________________________________________________________________________________
124
125  // ThingSpeak
126  unsigned long myChannelNumber = 726200;                         // ThingSpeak ID do canal
127  const char * myWriteAPIKey = "MPVSYRDHBVLOCHJ8";                // ThingSpeak chave de escrita
128  String myStatus = "";                                           // ThingSpeak para teste
129
130  //_____________________________________________________________________________________________________________________
131
132  // Inicializa o display Oled
133  SSD1306Wire display(0x3C, 5, 4);                                // Para ESP32
134
135  //_____________________________________________________________________________________________________________________
136
137  // Variaveis e constantes
138  char daysOfTheWeek[7][12] = {"Domingo", "Segunda", "Terca", "Quarta", "Quinta", "Sexta", "Sabado"};
139  unsigned long interval1;
140  unsigned long interval2;
141  unsigned long cod_monitor;        // 1 - ativa monitor serial 0 - destiva monitor serial
142  long lastReconnectAttempt = 0;
143  float eina1 = 0.0, delta1, eina2 = 0.0, delta2, eina3 = 0.0, delta3, etotalin = 0.0, etotalout = 0.0, coef1;
144
```

```
145  float vina1;
146  float cina1;
147  float pina1;
148
149  float vina2;
150  float cina2;
151  float pina2;
152
153  float vina3;
154  float cina3;
155  float pina3;
156
157  char v_ina1[6];
158  char c_ina1[6];
159  char p_ina1[6];
160  char e_ina1[6];
161
162  char v_ina2[6];
163  char c_ina2[6];
164  char p_ina2[6];
165  char e_ina2[6];
166
167  char v_ina3[6];
168  char c_ina3[6];
169  char p_ina3[6];
170  char e_ina3[6];
171
172  String tempo;
173  String VB;
174  String AT;
175  String PT;
176  String WT;
177
178  char buffer[100];
179  char par;
180  unsigned int wr = 0;
```

```
181  unsigned int ii = 0;
182  unsigned int pedido = 0;
183  unsigned long key = 2863311530;
184  const int inputPin = 15;
185
186  //_____________________________________________________________________________________________________________
187
188  void setup()
189  {
190    Serial.begin(115200);                                  // Inicia monitor serial
191    SerialXmega.begin(115200, SERIAL_8N1, 16, 17);         // Inicia serial UART
192    Wire.begin(5,4);                                       // Inicia I2C para ESP32
193    ThingSpeak.begin(wifiTS);                              // Inicia ThingSpeak
194    display.init();                                        // Inicia display Oled
195    pinMode(inputPin, INPUT_PULLUP);                       // Modo do inputPin
196
197    configura_INAs(1);                                     // Configura INAs 1 - imprime dados 0 - nao imprime
198    telainicial();                                         // Tela inicial no OLED
199    eeRead(0,eeprom);                                      // Le eeprom
200    delay(10000);                                          // Da um tempo
201
202    cod_monitor = eeprom.cod_mon;
203
204    int val = digitalRead(inputPin);                       // Le o valor do pino
205
206    if ((val == LOW) | (eeprom.valido != key))             // Forca reinicializacao default
207    {
208      eeprom.etracer1 = 0.0;
209      eeprom.etracer2 = 0.0;
210      eeprom.einversor = 0.0;
211      eeprom.eetotalin = 0.0;
212      eeprom.eetotalout = 0.0;
213      eeprom.inter1 = 60000;                               // 1 minuto
214      eeprom.inter2 = 300000;                              // 5 minutos
215      eeprom.cod_mon = 1;                                  // Habilita monitor serial
216      eeprom.valido = key;
```

```
217
218       eeWrite(0,eeprom);
219       delay (30);
220
221       cod_monitor = eeprom.cod_mon;
222
223       if (cod_monitor == 1)
224       {
225         Serial.println("");
226         Serial.println("eeprom default");
227         Serial.println("deixar o pino inputPin flutuante");
228         Serial.println("");
229       }
230     }
231     else
232     {
233       if (cod_monitor == 1)
234       {
235         Serial.println("");
236         Serial.println("eeprom valida");
237         Serial.println("");
238       }
239     }
240     eina1 = eeprom.etracer1;
241     eina2 = eeprom.etracer2;
242     eina3 = eeprom.einversor;
243     interval1 = eeprom.inter1;
244     interval2 = eeprom.inter2;
245     cod_monitor = eeprom.cod_mon;
246     coef1 = interval1 / 3600000.0;                              // Para calculo da energia
247
248     if (cod_monitor == 1)
249     {
250       Serial.println("configuracao na eeprom");
251       Serial.println("interval1    " + String(interval1));
252       Serial.println("interval2    " + String(interval2));
```

```
253       Serial.println("cod_monitor  " + String(cod_monitor));
254       Serial.println("");
255     }
256
257     // Avisa no display Oled
258     DateTime now = rtc.now();                                  // Pega data e hora atual
259     // Monta string com data e hora atual
260     tempo = String(now.day()) + '/' + String(now.month()) + '/' + String(now.year()) + "-" + String(now.hour())
261     + ':' + String(now.minute()) +  ':' + String(now.second());
262     display.clear();
263     display.setTextAlignment(TEXT_ALIGN_LEFT);
264     display.setFont(ArialMT_Plain_10);
265     display.drawString(1, 5, tempo);
266     display.drawString(1, 15, "configuracao da eeprom");
267     display.drawString(1, 25, "interval1       " + String(interval1));
268     display.drawString(1, 35, "interval2       " + String(interval2));
269     display.drawString(1, 45, "cod_monitor  " + String(cod_monitor));
270     display.display();
271     delay(5000);
272
273     conecta_wifi(1);                                           // Conecta WiFi 1 - imprime dados 0 - nao imprime
274   }
275
276   //_____________________________________________________________________________________________________________
277
278   void loop()
279   {
280     conecta_MQTT();                                            // Conecta com o Broker MQTT
281     MQTT.loop();                                               // Mantem MQTT connectado
282
283     //_____________________________________________________________________________________________________________
284
285     // Verifica se tem comunicacao vindo do Xmega para solicitacao e/ou configuracao
286     if (SerialXmega.available() > 0)
287     {
288       // Tem dados esperando no buffer. Formato: 'letra'numero,...,'letra'numero ex: a1,d1 ou d1,b5,c10 etc...
```

```
289       // a - digitos indicando solicitacao do Xmega
290       // b - digitos indicando minutos (interval1)
291       // c - digitos indicando minutos (interval2)
292       // d - digitos indicando: 1 - imprime no monitor serial, 0 - nao imprime no monitor serial
293
294       eeRead(0,eeprom);                                       // Le eeprom
295       delay(30);
296
297       while (SerialXmega.available() > 0)
298       {
299         par = SerialXmega.read();
300         switch (par)
301         {
302           case 'a':
303             pedido = SerialXmega.parseInt();
304             break;
305           case 'b':
306             eeprom.inter1 = SerialXmega.parseInt()  * 60000;    // Calcula em milisegundos
307             interval1 = eeprom.inter1;
308             wr = 1;
309             break;
310           case 'c':
311             eeprom.inter2 = SerialXmega.parseInt()  * 60000;    // Calcula em milisegundos
312             interval2 = eeprom.inter2;
313             wr = 1;
314             break;
315           case 'd':
316             eeprom.cod_mon = SerialXmega.parseInt();
317             cod_monitor = eeprom.cod_mon;
318             wr = 1;
319             break;
320         }
321       }
322
323       if (wr == 1)
324       {
```

```
325         eeWrite(0,eeprom);                                    // Grava eeprom
326         delay (30);
327         wr = 0;
328       }
329
330       DateTime now = rtc.now();                               // Pega data e hora atual
331       // Monta string com data e hora atual
332       tempo = String(now.day()) + '/' + String(now.month()) + '/' + String(now.year()) + "-" + String(now.hour())
333       + ':' + String(now.minute()) +  ':' + String(now.second());
334
335       if (pedido == 1)
336       {
337         SerialXmega.println(tempo + '#');                     // Se Xmega pediu, vamos mandar data e hora
338         pedido = 0;
339       }
340
341       // Avisa no display Oled
342       display.clear();
343       display.setTextAlignment(TEXT_ALIGN_LEFT);
344       display.setFont(ArialMT_Plain_10);
345       display.drawString(1, 5, tempo);
346       display.drawString(1, 15, "nova configuracao gravada");
347       display.drawString(1, 25, "interval1       " + String(interval1));
348       display.drawString(1, 35, "interval2       " + String(interval2));
349       display.drawString(1, 45, "cod_monitor  " + String(cod_monitor));
350       display.display();
351       delay(5000);
352
353       MQTT.publish(aviso2,"eeprom configurada");              // Eeprom configurada
354
355       if (cod_monitor == 1)
356       {
357         Serial.println("");
358         Serial.println("nova configuracao gravada na eeprom");
359         Serial.println("interval1    " + String(interval1));
360         Serial.println("interval2    " + String(interval2));
```

```
361         Serial.println("cod_monitor  " + String(cod_monitor));
362         Serial.println("");
363       }
364     }
365
366     //_____________________________________________________________________________________________________________
367
368     DateTime now = rtc.now();                                   // Pega data e hora atual
369     // Monta string com data e hora atual
370     tempo = String(now.day()) + '/' + String(now.month()) + '/' + String(now.year()) + "-" + String(now.hour())
371     + ':' + String(now.minute()) +  ':' + String(now.second());
372
373     //_____________________________________________________________________________________________________________
374
375     if (timeElapsed1 > interval1)                               // Executa quando completa um ciclo de duracao interval1
376     {
377       vina1 = ina1.readBusVoltage();
378       cina1 = ina1.readShuntCurrent();
379       pina1 = ina1.readBusPower();
380       delta1 = pina1 * coef1;
381       vina2 = ina2.readBusVoltage();
382       cina2 = ina2.readShuntCurrent();
383       pina2 = ina2.readBusPower();
384       delta2 = pina2 * coef1;
385       vina3 = ina3.readBusVoltage();
386       cina3 = ina3.readShuntCurrent();
387       pina3 = ina3.readBusPower();
388       delta3 = pina3 * coef1;
389       eina1 = eina1 + delta1;                                   // Integra energia
390       eina2 = eina2 + delta2;                                   // Integra energia
391       eina3 = eina3 + delta3;                                   // Integra energia
392       etotalin = eina1 + eina2;                                 // Integra energia total acumulada
393       etotalout = eina3;                                        // Integra energia total consumida
394
395       if (cod_monitor == 1)
396       {
```

```
397         // Imprime data e hora atual
398         Serial.println("");
399         Serial.println(tempo);
400         Serial.println("");
401         // Imprime leituras relativas ao Tracer1
402         Serial.println("INA226-1 Tracer1");
403         Serial.println("-----------------------------------------------");
404         Serial.print("Bus voltage:   ");
405         Serial.print(vina1, 2);
406         Serial.println(" V");
407         Serial.print("Shunt current: ");
408         Serial.print(cina1, 2);
409         Serial.println(" A");
410         Serial.print("Bus power:     ");
411         Serial.print(pina1, 2);
412         Serial.println(" W");
413         Serial.print("Bus energy:    ");
414         Serial.print(delta1, 2);
415         Serial.println(" Wh");
416         Serial.println("-----------------------------------------------");
417         Serial.println("");
418         // Imprime leituras relativas ao Tracer2
419         Serial.println("INA226-2 Tracer2");
420         Serial.println("-----------------------------------------------");
421         Serial.print("Bus voltage:   ");
422         Serial.print(vina2, 2);
423         Serial.println(" V");
424         Serial.print("Shunt current: ");
425         Serial.print(cina2, 2);
426         Serial.println(" A");
427         Serial.print("Bus power:     ");
428         Serial.print(pina2, 2);
429         Serial.println(" W");
430         Serial.print("Bus energy:    ");
431         Serial.print(delta2, 2);
432         Serial.println(" Wh");
```

```
433         Serial.println("-------------------------------------------------");
434         Serial.println("");
435         // Imprime leituras relativas ao Inversor
436         Serial.println("INA226-3 Inversor");
437         Serial.println("-------------------------------------------------");
438         Serial.print("Bus voltage:   ");
439         Serial.print(vina3, 2);
440         Serial.println(" V");
441         Serial.print("Shunt current: ");
442         Serial.print(cina3, 2);
443         Serial.println(" A");
444         Serial.print("Bus power:     ");
445         Serial.print(pina3, 2);
446         Serial.println(" W");
447         Serial.print("Bus energy:    ");
448         Serial.print(delta3, 2);
449         Serial.println(" Wh");
450         Serial.println("-------------------------------------------------");
451       }
452
453       //_____________________________________________________________________________________________________________________
454
455       // Para display na OLED
456       dtostrf(vina1, 5, 2, v_ina1);
457       dtostrf(cina1, 5, 2, c_ina1);
458       dtostrf(pina1, 5, 2, p_ina1);
459       dtostrf(eina1, 5, 2, e_ina1);
460
461       dtostrf(vina2, 5, 2, v_ina2);
462       dtostrf(cina2, 5, 2, c_ina2);
463       dtostrf(pina2, 5, 2, p_ina2);
464       dtostrf(eina2, 5, 2, e_ina2);
465
466       dtostrf(vina3, 5, 2, v_ina3);
467       dtostrf(cina3, 5, 2, c_ina3);
468       dtostrf(pina3, 5, 2, p_ina3);
```

```
469       dtostrf(eina3, 5, 2, e_ina3);
470
471       ii++;
472       if (ii > 3) ii = 1;
473       switch (ii)
474       {
475         case 1:
476         VB = "V-Banco     " + String(v_ina1);
477         AT = "A-Tracer1   " + String(c_ina1);
478         PT = "P-Tracer1   " + String(p_ina1);
479         WT = "Wh-Tracer1 " + String(e_ina1);
480         break;
481         case 2:
482         VB = "V-Banco     " + String(v_ina2);
483         AT = "A-Tracer2   " + String(c_ina2);
484         PT = "P-Tracer2   " + String(p_ina2);
485         WT = "Wh-Tracer2 " + String(e_ina2);
486         break;
487         case 3:
488         VB = "V-Banco      " + String(v_ina3);
489         AT = "A-Inversor   " + String(c_ina3);
490         PT = "P-Inversor   " + String(p_ina3);
491         WT = "Wh-Inversor " + String(e_ina3);
492         break;
493       }
494
495       display.clear();
496       display.setTextAlignment(TEXT_ALIGN_LEFT);
497       display.setFont(ArialMT_Plain_10);
498       display.drawString(1, 5, tempo);
499       display.drawString(1, 15, VB);
500       display.drawString(1, 25, AT);
501       display.drawString(1, 35, PT);
502       display.drawString(1, 45, WT);
503       display.display();
504
```

```
505       //_____________________________________________________________________________________________________________
506
507       // Envia dados via serial ao Xmega
508       SerialXmega.print("[dt="); SerialXmega.print(tempo);
509       SerialXmega.print(";v1="); SerialXmega.print(v_ina1);
510       SerialXmega.print(";c1="); SerialXmega.print(c_ina1);
511       SerialXmega.print(";p1="); SerialXmega.print(p_ina1);
512       SerialXmega.print(";e1="); SerialXmega.print(e_ina1);
513       SerialXmega.print(";v2="); SerialXmega.print(v_ina2);
514       SerialXmega.print(";c2="); SerialXmega.print(c_ina2);
515       SerialXmega.print(";p2="); SerialXmega.print(p_ina2);
516       SerialXmega.print(";e2="); SerialXmega.print(e_ina2);
517       SerialXmega.print(";v3="); SerialXmega.print(v_ina3);
518       SerialXmega.print(";c3="); SerialXmega.print(c_ina3);
519       SerialXmega.print(";p3="); SerialXmega.print(p_ina3);
520       SerialXmega.print(";e3="); SerialXmega.print(e_ina3);
521       SerialXmega.print(";ei="); SerialXmega.print(etotalin);
522       SerialXmega.print(";eo="); SerialXmega.print(etotalout); SerialXmega.print("]");
523
524       timeElapsed1 = 0;                                       // Reset do contador
525     }
526
527     //_____________________________________________________________________________________________________________
528
529     if (timeElapsed2 > interval2)                             // Executa quando completa um ciclo de duracao interval2
530     {
531       if (cod_monitor == 1)
532       {
533         Serial.println("-----------------------------------------------");
534         Serial.print("Energia Tracer1: ");
535         Serial.print(eina1, 2);
536         Serial.println(" Wh");
537         Serial.println("-----------------------------------------------");
538         Serial.println("");
539
540         Serial.println("-----------------------------------------------");
```

```
541         Serial.print("Energia Tracer2: ");
542         Serial.print(eina2, 2);
543         Serial.println(" Wh");
544         Serial.println("-----------------------------------------------");
545         Serial.println("");
546 
547         Serial.println("-----------------------------------------------");
548         Serial.print("Energia Inversor: ");
549         Serial.print(eina3, 2);
550         Serial.println(" Wh");
551         Serial.println("-----------------------------------------------");
552         Serial.println("");
553 
554         Serial.println("-----------------------------------------------");
555         Serial.print("Energia acumulada: ");
556         Serial.print(etotalin, 2);
557         Serial.println(" Wh");
558         Serial.println("-----------------------------------------------");
559         Serial.println("");
560 
561         Serial.println("-----------------------------------------------");
562         Serial.print("Energia consumida: ");
563         Serial.print(etotalout, 2);
564         Serial.println(" Wh");
565         Serial.println("-----------------------------------------------");
566         Serial.println("");
567       }
568 
569       //_____________________________________________________________________________________________________________
570 
571       // Documentos para Json
572       StaticJsonDocument<100> doc1;
573       StaticJsonDocument<100> doc2;
574       StaticJsonDocument<100> doc3;
575       StaticJsonDocument<100> doc4;
576       StaticJsonDocument<100> doc5;
```

```
577
578       doc1["data"] = tempo;
579       JsonArray ina1 = doc1.createNestedArray("tracer1");
580       ina1.add(String(vina1, 2));
581       ina1.add(String(cina1, 2));
582       //ina1.add(pina1);
583       ina1.add(String(eina1, 2));
584
585       serializeJson(doc1, buffer);
586
587       if (cod_monitor == 1)
588       {
589         Serial.println("");
590         Serial.println(buffer);
591       }
592
593       MQTT.publish(tracer1, buffer);                          // Publica MQTT tracer1
594
595       doc2["data"] = tempo;
596       JsonArray ina2 = doc2.createNestedArray("tracer2");
597       ina2.add(String(vina2, 2));
598       ina2.add(String(cina2, 2));
599       //ina2.add(pina2);
600       ina2.add(String(eina2, 2));
601
602       serializeJson(doc2, buffer);
603
604       if (cod_monitor == 1)
605       {
606         Serial.println("");
607         Serial.println(buffer);
608       }
609
610       MQTT.publish(tracer2, buffer);                          // Publica MQTT tracer2
611
612       doc3["data"] = tempo;
```

```
613       JsonArray ina3 = doc3.createNestedArray("inversor");
614       ina3.add(String(vina3, 2));
615       ina3.add(String(cina3, 2));
616       //ina3.add(pina3);
617       ina3.add(String(eina3, 2));
618
619       serializeJson(doc3, buffer);
620
621       if (cod_monitor == 1)
622       {
623         Serial.println("");
624         Serial.println(buffer);
625       }
626
627       MQTT.publish(inversor, buffer);                          // Publica MQTT inversor
628
629       doc4["data"] = tempo;
630       doc4["energia_total_in"] = String(etotalin, 2);
631       serializeJson(doc4, buffer);
632
633       if (cod_monitor == 1)
634       {
635         Serial.println("");
636         Serial.println(buffer);
637       }
638
639       MQTT.publish(etotin, buffer);                            // Publica MQTT energia acumulada
640
641       doc5["data"] = tempo;
642       doc5["energia_total_out"] = String(etotalout, 2);
643       serializeJson(doc5, buffer);
644
645       if (cod_monitor == 1)
646       {
647         Serial.println("");
648         Serial.println(buffer);
```

```
649       }
650
651       MQTT.publish(etotout, buffer);                          // Publica MQTT energia
          consumida
652
653       //_____________________________________________________________________________________________________
654
655       // Define os valores para os campos do canal ThingSpeak
656       ThingSpeak.setField(1, vina1);
657       ThingSpeak.setField(2, cina1);
658       ThingSpeak.setField(3, vina2);
659       ThingSpeak.setField(4, cina2);
660       ThingSpeak.setField(5, vina3);
661       ThingSpeak.setField(6, cina3);
662       ThingSpeak.setField(7, etotalin);
663       ThingSpeak.setField(8, etotalout);
664
665       // Define o status para o canal do ThingSpeak
666       if(cina1 > cina2)
667       {
668         myStatus = String("corrente de carga tracer1 > tracer2");
669       }
670       else if(cina1 < cina2)
671       {
672         myStatus = String("corrente de carga tracer2 > tracer1");
673       }
674       else if(etotalout > etotalin)
675       {
676         myStatus = String("energia consumida maior que energia acumulada");
677       }
678       else {}
679
680       // Grava o status
681       ThingSpeak.setStatus(myStatus);
682
683       // Envia dados para o canal do ThingSpeak
```

```
684       int x = ThingSpeak.writeFields(myChannelNumber, myWriteAPIKey);
685       if(x == 200)
686       {
687         if (cod_monitor == 1)
688         {
689           Serial.println("");
690           Serial.println("Canal atualizado com sucesso");
691           Serial.println("");
692         }
693       }
694       else
695       {
696         if (cod_monitor == 1)
697         {
698           Serial.println("");
699           Serial.println("Problema fazendo o update do canal. Codigo de erro HTTP " + String(x));
700         }
701       }
702
703       //_____________________________________________________________________________________________________________
704
705       // Grava eeprom
706       eeprom.etracer1 = eina1;
707       eeprom.etracer2 = eina2;
708       eeprom.einversor = eina3;
709       eeprom.eetotalin = etotalin;
710       eeprom.eetotalout = etotalout;
711       eeWrite(0,eeprom);
712       delay (30);
713
714       if (cod_monitor == 1)
715       {
716         Serial.print("energia tracer1 ");
717         Serial.print(eeprom.etracer1, 2);
718         Serial.println(" Wh");
719         Serial.print("energia tracer2 ");
```

```
720         Serial.print(eeprom.etracer2, 2);
721         Serial.println(" Wh");
722         Serial.print("energia inversor ");
723         Serial.print(eeprom.einversor, 2);
724         Serial.println(" Wh");
725         Serial.print("energia total acumulada ");
726         Serial.print(eeprom.eetotalin, 2);
727         Serial.println(" Wh");
728         Serial.print("energia total consumida ");
729         Serial.print(eeprom.eetotalout, 2);
730         Serial.println(" Wh");
731         Serial.println("");
732       }
733
734       timeElapsed2 = 0;                                   // Reset do contador
735     }
736   }
737
738   //_____________________________________________________________________________________________________________________
739
740   void telainicial()
741   {
742     display.clear();                                      //Apaga o display
743     display.setTextAlignment(TEXT_ALIGN_CENTER);
744     display.setFont(ArialMT_Plain_16);                    //Seleciona a fonte
745     display.drawString(63, 5, "CoMoSisFog V1.0");
746     display.drawString(63, 25, "ESP32");
747     display.drawString(63, 45, "Filippo Pardini");
748     display.display();
749   }
750
751   //_____________________________________________________________________________________________________________________
752
753   void conecta_MQTT(void)
754   {
755     if (!MQTT.connected())
```

```
756     {
757       long now = millis();
758       if (now - lastReconnectAttempt > 5000)
759       {
760         lastReconnectAttempt = now;
761         if (reconnect())                                    // Tentativa de reconexao
762         {
763           if (cod_monitor == 1)
764           {
765             Serial.println("");
766             Serial.println("Broker MQTT conectado");
767             Serial.println("");
768           }
769           lastReconnectAttempt = 0;
770         }
771       }
772     }
773   }
774
775   //_____________________________________________________________________________________________________________
776
777   boolean reconnect()
778   {
779     if (MQTT.connect("quiosque",ID_MQTT,PSW_MQTT))
780     {
781       MQTT.publish(aviso1,"conectado");                     // Quando reconectado, publica aviso
782     }
783     return MQTT.connected();
784   }
785
786   //_____________________________________________________________________________________________________________
787
788   void configura_INAs(unsigned char ch)
789   {
790     // Inicia INA1 endereco 0x40
791     ina1.begin(0x40);
```

```
792     // Configura
793     ina1.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
794     INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
795     // Calibra INA226 - shunt = 0.1 ohm, corrente maxima esperada = 0.8A
796     ina1.calibrate(0.1, 0.8);
797
798     // Inicia INA2 endereco 0x40
799     ina2.begin(0x40);
800     // Configura
801     ina2.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
802     INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
803     // Calibra INA226 - shunt = 0.1 ohm, corrente maxima esperada = 0.8A
804     ina2.calibrate(0.1, 0.8);
805
806     // Inicia INA3 endereco 0x40
807     ina3.begin(0x40);
808     // Configura
809     ina3.configure(INA226_AVERAGES_64, INA226_BUS_CONV_TIME_1100US,
810     INA226_SHUNT_CONV_TIME_1100US, INA226_MODE_SHUNT_BUS_CONT);
811     // Calibra INA226 - shunt = 0.1 ohm, corrente maxima esperada = 0.8A
812     ina3.calibrate(0.1, 0.8);
813
814   /*
815     if (ch == 1)
816     {
817       // Display das configuracoes
818       checkConfig(1);
819       checkConfig(2);
820       checkConfig(3);
821     }
822   */
823   }
824
825   //_____________________________________________________________________________________________________________
826
827   void conecta_wifi(unsigned char pr)
```

```
828   {
829     WiFi.begin(ssid, password);                                   // Inicia WiFi
830     while (WiFi.status() != WL_CONNECTED)
831     {
832       delay(500);
833       if (cod_monitor == 1)
834       {
835         Serial.println("Conectando WiFi..");
836       }
837     }
838     if (cod_monitor == 1)
839     {
840       Serial.print("Connectado na rede ");
841       Serial.println(ssid);
842     }
843     if (pr == 1)
844     {
845       IPAddress ip = WiFi.localIP();                              // Imprime o IP
846       if (cod_monitor == 1)
847       {
848         Serial.print("IP: ");
849         Serial.println(ip);
850       }
851
852       long rssi = WiFi.RSSI();                                    // Imprime o nível do sinal
853       if (cod_monitor == 1)
854       {
855         Serial.print("Nivel do sinal (RSSI): ");
856         Serial.println(rssi);
857       }
858     }
859   }
860
861   //_____________________________________________________________________________________________________________
862
```