```
/*
ReferÃªncias:
https://www.fernandok.com/2017/12/modulo-sd-card-com-esp8266.html
https://www.filipeflop.com/blog/cartao-sd-com-arduino/
http://pedrominatel.com.br/pt/esp8266/data-e-hora-no-esp8266-com-ntp/
http://geek.adachsoft.com/img/DS1307/schematic.png
http://geek.adachsoft.com/home/article/id/2/n/ESP8266-wiht-DS1307-RTC,-NV-SRAM-
and-Square-Wave-output-signal/refid/acc

Equivalencia das saidas Digitais entre NodeMCU e ESP8266 (na IDE do Arduino)
NodeMCU - ESP8266
D0 = 16;
D1 = 5;
D2 = 4;
D3 = 0;
D4 = 2;
D5 = 14;
D6 = 12;
D7 = 13;
D8 = 15;
D9 = 3;
D10 = 1;
*/

#include <SdFat.h>
#include <DHT.h>
#include <ESP8266WiFi.h>
#include <NTPClient.h>
#include <WiFiUdp.h>
#include <time.h>
#include <math.h>
#include <PubSubClient.h> //https://www.youtube.com/watch?v=oX4ttJEULmA
#include "RTC_DS1307.h" //http://geek.adachsoft.com/home/article/id/2/n/ESP8266-
wiht-DS1307-RTC,-NV-SRAM-and-Square-Wave-output-signal/refid/acc

#define DHTPIN D3     // pino de dados do DHT serÃƒÂ¡ ligado no D6 do esp
#define DHTTYPE DHT22   // tipo do sensor

#define servidor_mqtt             "m15.cloudmqtt.com"  //URL do servidor MQTT
#define servidor_mqtt_porta       "19921"  //Porta do servidor (a mesma deve ser
informada na variÃ¡vel abaixo)
#define servidor_mqtt_usuario     "ptstacxh"  //UsuÃ¡rio
#define servidor_mqtt_senha       "kRP4DMHZULJd"  //Senha
//#define mqtt_topico_pub           "esp8266/pincmd"    //TÃ³pico para publicar
o comando de inverter o pino do outro ESP8266

SdFat sdCard;
SdFile meuArquivo;

// construtor do objeto para comunicar com o sensor
DHT dht(DHTPIN, DHTTYPE);
unsigned long id1 = 0;
//pino ligado ao CS do mÃ³dulo SD Card
#define CS_PIN  D8

WiFiUDP ntpUDP;

int16_t utc = -3; //UTC -3:00 Brazil
uint32_t currentMillis = 0;
uint32_t previousMillis = 0;

NTPClient timeClient(ntpUDP, "a.st1.ntp.br", utc*3600, 60000);

WiFiClient espClient;
```

```
PubSubClient client(espClient); //Passando a instÃ¢ncia do WiFiClient para a
instÃ¢ncia do PubSubClient

RTC_DS1307 rtc;

//WiFiServer server(80);

int ano;
int mes;
int dia;
int hora;
int minuto;
float segundo;
String datac;
String dias;
String meses;
String horas;
String minutos;
String segundos;
String arquivo;

int ldr = 0; //Setando a utilizaÃ§Ã¢o do LDR Ã  porta ADC A0 do Nodemcu
//int valorldr = 0;//variÃ¡vel para armazenar a leitura do ldr.
float vldr = 0;
bool precisaSalvar = false; //Flag para salvar os dados

//FunÃ§Ã£o que reconecta ao servidor MQTT
void reconectar() {
  //Repete atÃ© conectar
  while (!client.connected()) {
    Serial.println("Tentando conectar ao servidor MQTT...");

    //Tentativa de conectar. Se o MQTT precisa de autenticaÃ§Ã£o, serÃ¡ chamada
a funÃ§Ã£o com autenticaÃ§Ã£o, caso contrÃ¡rio, chama a sem autenticaÃ§Ã£o.
    bool conectado = strlen(servidor_mqtt_usuario) > 0 ?
                     client.connect("BANGALo", servidor_mqtt_usuario,
servidor_mqtt_senha) :
                     client.connect("BANGALo");

    if(conectado) {
      Serial.println("Conectado ao MQTT!");
    } else {
      Serial.println("Falhou ao tentar conectar. Codigo: ");
      Serial.println(String(client.state()).c_str());
      Serial.println(" tentando novamente em 5 segundos");
      //Aguarda 5 segundos para tentar novamente
      delay(5000);
    }
  }
}

void desconectar(){
  Serial.println("Fechando a conexao com o servidor MQTT...");
  client.disconnect();
}

//FunÃ§Ã£o que envia os dados de umidade e temperatura.
void publicaComando(char topico[60], char valor[60]) {
  if (!client.connected()) {
    Serial.println("MQTT desconectado! Tentando reconectar...");
    reconectar();
  }

  client.loop();
```

```
  //Publicando no MQTT
  Serial.println("Fazendo a publicacao...");
  client.publish(topico, valor, true);
}

void forceUpdate(void) {
    timeClient.forceUpdate();
  }

void checkOST(void) {
    currentMillis = millis();//Tempo atual em ms
    //LÃ³gica de verificaÃ§Ã£o do tempo
    if (currentMillis - previousMillis > 1000) {
      previousMillis = currentMillis;    // Salva o tempo atual
      //printf("Time Epoch: %d: ", timeClient.getEpochTime());
      //Serial.print(timeClient.getYear());
      //Serial.Print("-");
      Serial.println(timeClient.getFormattedTime());
      time_t dt1 = timeClient.getEpochTime();
      struct tm * ti;
      ti = localtime (&dt1);
      ano = ti->tm_year + 1900;
      mes = ti->tm_mon + 1;
      dia = ti->tm_mday;
      hora = ti->tm_hour;
      minuto = ti->tm_min;
      segundo = ti->tm_sec;

      if(mes<10){
        meses = "0" + String(mes);
      }else{
        meses = String(mes);
      }
       if(dia<10){
        dias = "0" + String(dia);
      }else{
        dias = String(dia);
      }
       if(hora<10){
        horas = "0" + String(hora);
      }else{
        horas = String(hora);
      }
       if(minuto<10){
        minutos = "0" + String(minuto);
      }else{
        minutos = String(minuto);
      }
       if(segundo<10){
        segundos = "0" + String(segundo, 0).substring(2);
      }else{
        segundos = String(segundo, 0);
      }

      datac= dias + "/" + meses + "/" + String(ano) + " " + horas + ":" +
minutos + ":" + segundos;
      Serial.print("data formatada: ");
      Serial.println(datac);
    }
}

String miliToHoras(unsigned long milli){
      int segs;
```

```
      int horas;
      int minutos;
      float segundos;
      segs = milli / 1000;
      horas = segs / 3600;
      minutos = (segs % 3600) / 60;
      segundos = (milli/1000) - (minutos * 60) - (horas * 3600);
      String res;
      if(horas<10){
        res = "0" + String(horas);
      }else{
        res = String(horas);
      }
      if(minutos<10){
        res = res + ":" + "0" + String(minutos);
      }else{
        res = res + ":" + String(minutos);
      }
      if(segundos<10){
        res = res + ":" + "0" + String(segundos,1).substring(2);
      }else{
        res = res + ":" + String(segundos,1);
      }
      return res + " hh:mm:ss.s";
  }

void setup(){
  Serial.begin(9600);

   Serial.print("Conectando");
  //Faz o ESP se conectar Ã  rede WiFi. No nosso exemplo o ssid da rede Ã©
TesteESP e a senha Ã© 87654321
  WiFi.begin("Sertao", "xxkkhlcg");

  int i1 = 0;
  //Enquanto o ESP nÃ£o se conectar Ã  rede
  while (WiFi.status() != WL_CONNECTED)
  {
    //Esperamos 100 milisegundos
    delay(100);
    Serial.print("Tentativa: ");
    Serial.println(++i1);
  }

  //Se chegou aqui Ã© porque conectou Ã  rede, entÃ£o mostramos
  //no monitor serial para termos um feedback
  Serial.println("");
  Serial.println("Conectou");

  Serial.print("Inicializando o cartÃ£o SD...");

  //inicializa o objeto para comunicarmos com o sensor DHT
  dht.begin();

/*  // verifica se o cartÃƒÂ£o SD estÃ¡Â¡ presente e se pode ser inicializado
  if (!SD.begin(CS_PIN)) {
    Serial.println("Falha, verifique se o cartÃ£o estÃ¡Â¡ presente.");
    //programa encerrrado
    return;
  }*/

   // Inicializa o modulo SD
  if(!sdCard.begin(CS_PIN,SPI_HALF_SPEED)){
    sdCard.initErrorHalt();
```

```
    Serial.println("Falha, verifique se o cartÃ£o estÃ¡Â¡ presente. (SdFat");
  }else{
    //se chegou aqui Ã© porque o cartÃ£oo foi inicializado corretamente
    Serial.println("CartÃ£o inicializado.");
  }

  timeClient.begin();
  timeClient.update();
  checkOST();
  arquivo =  String(ano) + "_" + meses + "_" + dias + "_" + horas + "_" +
minutos + "_" + segundos + ".CSV";
  Serial.println(arquivo);

  pinMode(D0, OUTPUT);

  //Informando ao client do PubSub a url do servidor e a porta.
  int portaInt = atoi(servidor_mqtt_porta);
  client.setServer(servidor_mqtt, portaInt);

  Wire.begin();
  //byte second, byte minute, byte hour, byte dayOfWeek, byte dayOfMonth, byte
month, byte year
  //Setting the time on DS1307
  rtc.SetTime( segundo, minuto, hora, 2, dia, mes, ano - 2000 );
  //rtc.SQW( f32768hz );

}

void loop(){

  //Chama a verificacao de tempo
  checkOST();
  //Serial.println("teste..");
  Serial.print("ID: ");
  Serial.println(id1);
  Serial.print("TEMPO: ");
  Serial.print(millis());
  Serial.print(" - ");
  Serial.println(miliToHoras(millis()));
  //faz a leitura da umidade
  float umidade = dht.readHumidity();
  Serial.print("Umidade: ");
  Serial.println(umidade);
  //faz a leitura da temperatura
  float temperatura = dht.readTemperature();
  Serial.print("Temperatura: ");
  Serial.println(temperatura);
  vldr = analogRead(A0);
  Serial.print("LUMINOSIDADE: ");
  Serial.println(vldr);

  // Abre o arquivo
  char temp[255];
  arquivo.toCharArray(temp, 255);
  if (!meuArquivo.open(temp, O_RDWR | O_CREAT | O_AT_END)){
    sdCard.errorHalt("Erro na abertura do arquivo ");
    Serial.println("Falha ao abrir o arquivo " + arquivo);
  }
  if(!isnan(temperatura) && !isnan(umidade) && !isnan(vldr)){
    meuArquivo.print(id1);
    meuArquivo.print("; ");
    meuArquivo.print(datac);
    meuArquivo.print("; ");
    meuArquivo.print(umidade);
```

```
    meuArquivo.print("; ");
    meuArquivo.print(temperatura);
    meuArquivo.print("; ");
    meuArquivo.println(vldr);
    id1++;
    meuArquivo.close();
    Serial.println("Escrito no Arquivo");
  }

  char temp2[60];
  datac.toCharArray(temp2, datac.length());
  publicaComando("Bangalo/DATA", temp2);

  dtostrf(umidade, 6, 2, temp2); // Leave room for too large numbers!;
  publicaComando("Bangalo/UMIDADE", temp2);
  dtostrf(temperatura, 6, 2, temp2); // Leave room for too large numbers!;
  publicaComando("Bangalo/TEMPERATURA", temp2);
  dtostrf(vldr, 6, 2, temp2); // Leave room for too large numbers!;
  publicaComando("Bangalo/LUMINOSIDADE", temp2);
  desconectar();

  //intervalo de espera para uma nova leitura dos dados.
  delay(30000);
}
```