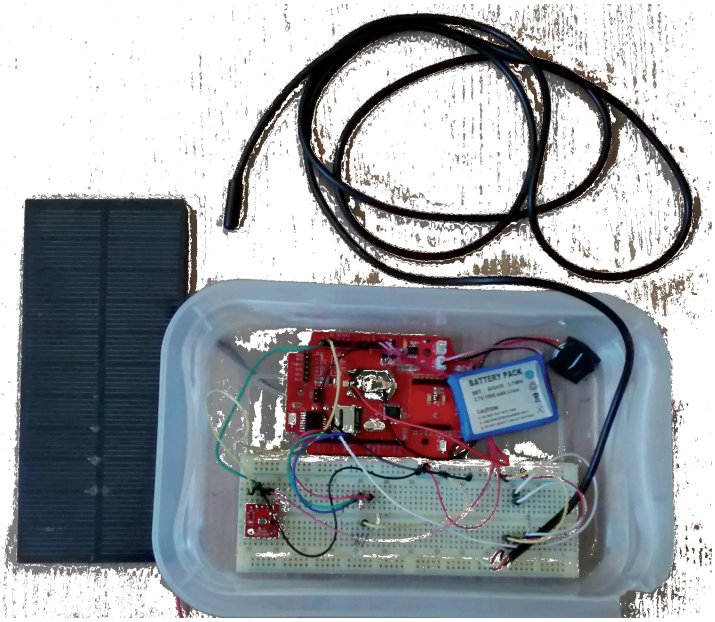
Documentos
206
ISSN 1518-4277
Dezembro, 2016

Sensores Móveis e Autônomos de Temperatura de Baixo Custo



Embrapa

ISSN 1518-4277
Dezembro, 2016

Empresa Brasileira de Pesquisa Agropecuária
Embrapa Milho e Sorgo
Ministério da Agricultura, Pecuária e Abastecimento

Documentos 206

Sensores Móveis e Autônomos de Temperatura de Baixo Custo

Ricardo Nunes Nery
Elena Charlotte Landau
Anderson Henrique dos Santos
Daniel Pereira Guimarães
Renan Vieira Mechetti Ferreira

Embrapa Milho e Sorgo
Sete Lagoas, MG
2016

Exemplares desta publicação podem ser adquiridos na:

Embrapa Milho e Sorgo
Rod. MG 424 Km 45
Caixa Postal 151
CEP 35701-970 Sete Lagoas, MG
Fone: (31) 3027-1100
Fax: (31) 3027-1188
www.embrapa.br/fale-conosco

Revisão de texto: Antonio Claudio da Silva Barros
Normalização bibliográfica: Rosângela Lacerda de Castro
Tratamento de ilustrações: Tânia Mara Assunção Barbosa
Editoração eletrônica: Tânia Mara Assunção Barbosa
Foto(s) da capa: Ricardo Nunes Nery

1ª edição
Versão Eletrônica (2016)

Dados Internacionais de Catalogação na Publicação (CIP)
Embrapa Milho e Sorgo

Sensores móveis e autônomos de temperatura de baixo custo / Ricardo Nunes Nery ... [et al.]. -- Sete Lagoas : Embrapa Milho e Sorgo, 2016.
25 p. : il. -- (Documentos / Embrapa Milho e Sorgo, ISSN 1518-4277; 206).

1. Clima. 2. Temperatura. 3. Sensor. 4. Armazenamento de dados. I. Nery, Ricardo Nunes. II. Série.

CDD 551.6 (21. ed.)

Autores

Ricardo Nunes Nery
Estudante do Curso de Agronomia da Univ. Fed. de São João del-Rei, Bacharel em Sistemas de Informação, Bolsista PIBIC do Convênio CNPq/Embrapa Milho e Sorgo, Rod MG 424 Km 45, Zona Rural, CEP: 35701-970, Sete Lagoas, MG

Elena Charlotte Landau
Bióloga, DSc. Pesquisadora em Zoneamento Ecológico-Econômico, Geotecnologias e Agroclimatologia na Embrapa Milho e Sorgo, MG 424 Km 45, Zona Rural, CEP: 35701-970, Sete Lagoas, MG, charlotte.landau@embrapa.br

Daniel Pereira Guimarães
Engenheiro Florestal, D.Sc. em Manejo Florestal, Pesquisador na Embrapa Milho e Sorgo,Rod MG 424 Km 45, Zona Rural, CEP: 35701-970, Sete Lagoas, MG, daniel.guimaraes@embrapa.br

Anderson Henrique dos Santos
Estudante do Curso de Sistemas de Informação da Faculdade Cenecista de Sete Lagoas, estagiário na Embrapa Milho e Sorgo, Embrapa Milho e Sorgo, Rod MG 424 Km 45, Zona Rural, CEP: 35701-970, Sete Lagoas, MG

Renan Vieira Mechetti Ferreira
Estudante do Curso de Agronomia da Univ. Fed. de São João del-Rei, Campus Sete Lagoas, estagiário na Embrapa Milho e Sorgo, Embrapa Milho e Sorgo - Rod MG 424 Km 45, Zona Rural, CEP: 35701-970, Sete Lagoas, MG

Apresentação

A verificação da temperatura foi iniciada no final do século XVI e até hoje continua evoluindo. Há várias aplicações para as medições de temperatura, mas seu registro automático nem sempre é possível. O registro de temperatura automático ainda pode representar um desafio, pois os equipamentos capazes de armazenar essas informações são caros, tornando-se inatingíveis para grande parte da população. Neste trabalho, são aplicadas tecnologias livres, para desenvolvimento de um amostrador de temperatura autônomo utilizando sensores de baixo custo. A conexão entre os sensores e o armazenamento de dados se dá em um microcontrolador, difundido por também apresentar baixo custo e ser de hardware livre. Neste trabalho, foi desenvolvido e testado um amostrador autônomo de temperatura com registro automático utilizando Arduino. O suprimento energético foi provido por um painel solar e uma bateria, fazendo assim com que o amostrador tenha grande mobilidade em campo.

Antonio Alvaro Corsetti Purcino
Chefe-Geral
Embrapa Milho e Sorgo

Sumário

Sensores Móveis e Autônomos de Temperatura de Baixo Custo[1]

Ricardo Nunes Nery

Elena Charlotte Landau

Anderson Henrique dos Santos

Daniel Pereira Guimarães

Renan Vieira Mechetti Ferreira

Introdução

Sendo considerado por Ayoade (1996) como possível elemento mais discutido do tempo atmosférico, a temperatura é uma grandeza física que representa a energia cinética de um corpo. A temperatura pode ser utilizada para caracterizar o estado da atmosfera, podendo ser de forma instantânea ou através de estatística média de medições (PEREIRA et al., 2007).

A temperatura pode ser registrada em valores médios (diários, mensais e anuais) e valores de máxima e mínima (PEREIRA et al., 2007). Esta variável pode ser utilizada como referência para a compreensão de vários processos termorregulatórios em seres vivos (BARROS et al., 2014).

O conceito de graus-dia, onde se considera o somatório da temperatura do ar durante o ciclo de uma cultura resultante de uma constante, constante esta que pode ser considerada como

[1]Trabalho financiado pelo CNPq

a maturação, estádio fenológico, entre outras (CARAMORI, 2006), é utilizado para auxiliar escolhas no campo, como: material genético para plantio, planejamento da época de plantio, planejamento da época de colheita, estimativa do potencial de infestações e outros (CARAMORI, 2006). O conceito de horas de frio também é utilizado para auxiliar escolhas, da mesma forma como os graus-dia é somado e acumulado gerando uma constante (CARAMORI, 2006). Os dois conceitos apresentados são indicadores importantes para a agricultura, e utilizam medidas de temperatura para suas afirmações, reafirmando assim a importância de trabalhos como este.

O objetivo deste trabalho foi realizar a montagem de um amostrador de temperatura, sendo ele autônomo, de baixo custo de aquisição, com mobilidade, possibilitando uma análise comparativa da temperatura.

Desenvolvimento do Sistema de Leitura

O registro da temperatura neste projeto foi feito através dos sensores BMP180 e DS18B20 (Sensor Solo). As características dos sensores podem ser observadas na Tabela 1.

Para a ligação dos sensores foi necessária a utilização de um microcontrolador. No caso deste projeto o microcontrolador escolhido foi o *Arduino* UNO. Além do *Arduino*, o módulo de relógio, um cartão de memória e seu gravador, possibilitando o registro com data e horário das leituras de temperatura realizadas pelos sensores.

Tabela 1. Características dos sensores de temperatura

Modelo dos sensores de temperatura	 BMP180	 DS18B20
Precisão	+/- 1 °C entre 0 °C e +65 °C	+/- 0,5 °C entre -10 °C e +85 °C
Faixa de medição	0 °C e +65 °C	-55 °C a +125 °C

O *Arduino* necessita de energia para seu funcionamento, a opção utilizada foi uma célula fotossensível de 2 *watts* de potência e 5 *volts*, além de um controlador de energia e uma bateria de 1 *ampere* com 3,7 *volts*.

O código fonte pode ser observado no apêndice A que, para atender as necessidades do projeto, foi adaptado de outros códigos fontes, como: *Lab de Garagem* (http://labdegaragem.com/profiles/blogs/tutorial-reprodu-o-de-cores-com-o-led-rgb-e-o-ardui no), *Seeed Studio* (http://www.seeedstudio.com/wiki/Seeeduino_Stalker_v2.3) e *SparkFun* (https://www.sparkfun.com/products/11824). Na Figura 1 é mostrado o desenho das ligações dos componentes, microcontrolador e sensores.

A Tabela 2 apresenta os custos com os componentes do projeto (os impostos necessários para importação dos componentes não estão incluídos na tabela; utilizar uma taxa de 100% para se estimar o valor real gasto). Neste caso, o valor foi de US$ 89,14 para a confecção deste projeto.

Tabela 2. Componentes do projeto com valor em dólares.

Componente	Valor (US$)
Arduino UNO RV3	24.95
SD Card	3.95
BMP 180	9.95
DS18B20	9.95
RTC	14.95
Kit Solar	24.95
RGB	0.25
Resistores	0.19
Total	**89.14**

Objetivando realizar a comparação dos dados obtidos através do amostrador autônomo, foi realizada a sua instalação junto à estação meteorológica do Instituto Nacional de Meteorologia (INMET), em Sete Lagoas (19° 29′ 5″ S, 44° 10′ 25″ W).

O registro das informações da estação convencional INMET (Sete Lagoas-MG) ocorreu durante 46 dias. Mostra-se gráfico, na Figura 2, da amplitude térmica observada no período. A amplitude térmica média, por dia, foi de 16,8 °C com desvio padrão de 4,28 °C e coeficiente de variação de 25,5%.

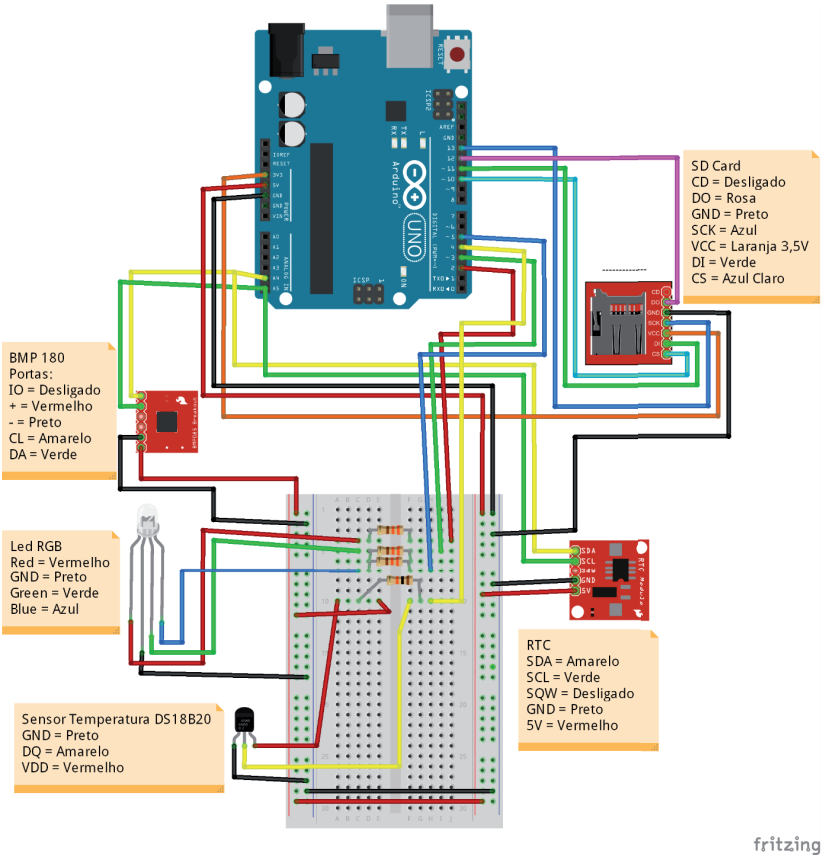


Figura 1. Desenho esquemático das ligações dos sensores, componentes e microcontrolador.

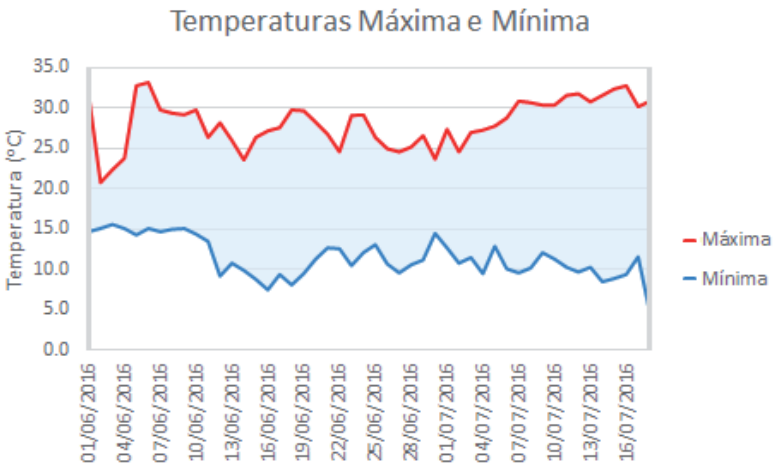


Figura 2. Temperatura máxima e mínima registrada na estação convencional INMET (Sete Lagoas-MG).

Resultados e Discussão

A Figura 3 demonstra o amostrador autônomo após sua montagem e programação embarcada e a Figura 4 mostra a estação meteorológica INMET (Sete Lagoas-MG) com o amostrador autônomo desenvolvido em seu interior.

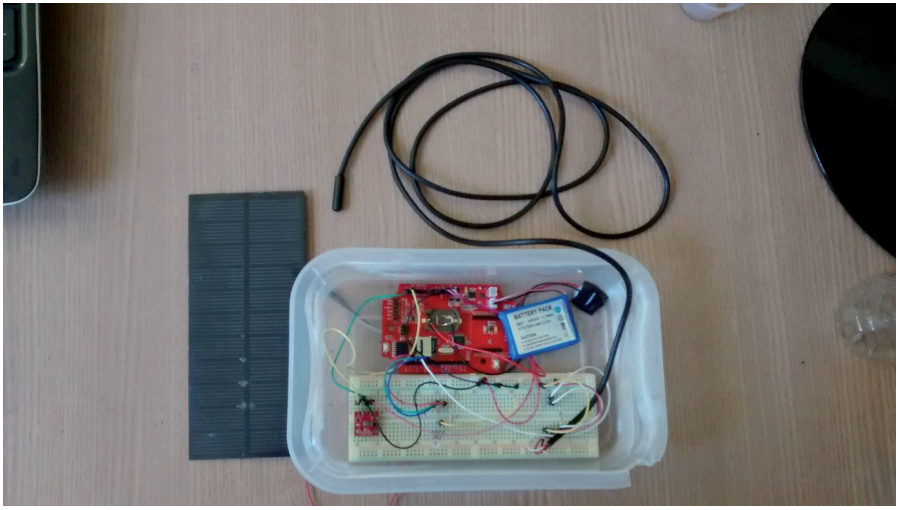


Figura 3. Interior da unidade amostradora autônoma de temperatura.

Figura 4. Abrigo meteorológico da estação convencional INMET (Sete Lagoas - MG). Em 'A' está o amostrador autônomo.

As informações obtidas entre os dias 1° de junho de 2016 e 17 de julho de 2016 foram armazenadas em arquivos texto no cartão de memória (exemplo do arquivo texto do dia 5 de junho de 2016 pode ser observado no apêndice B, os primeiros 5 minutos com registro a cada 5 segundos). Após extrair o cartão de memória, foi possível realizar a relação entre as informações registradas pelos sensores e as informações disponibilizadas pela estação convencional INMET (Sete Lagoas-MG). A Figura 5 apresenta a relação entre as informações coletadas às 15 horas.

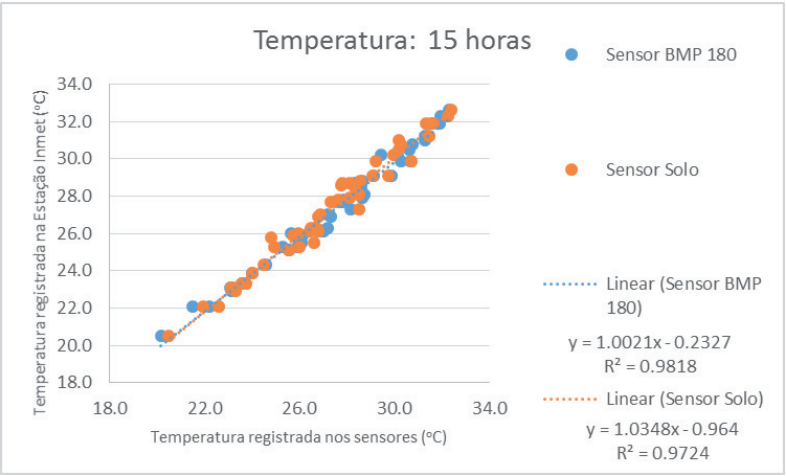


Figura 5. Relação entre as temperaturas registradas às 15 horas entre os sensores e a estação convencional INMET.

A Figura 6 apresenta a relação entre as temperaturas diárias mínimas observadas e a Figura 7 mostra a relação entre as temperaturas diárias máximas observadas.

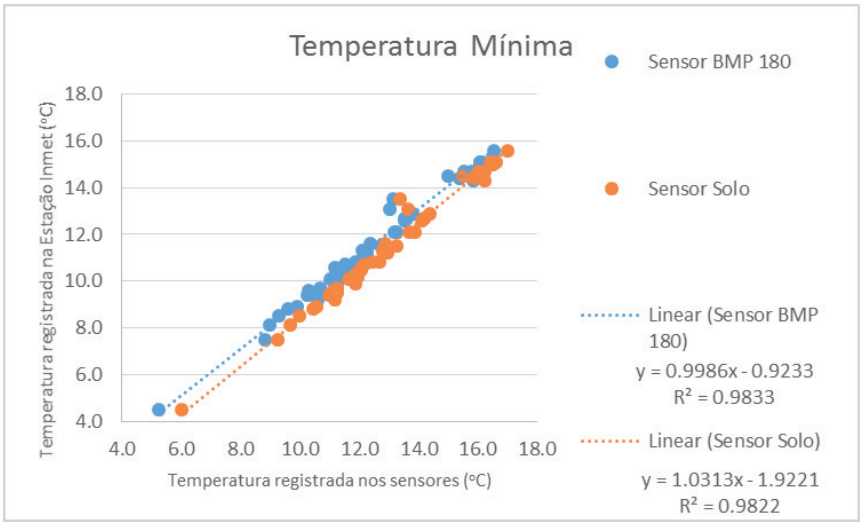


Figura 6. Relação entre as temperaturas mínimas diárias entre os sensores e a estação convencional INMET.

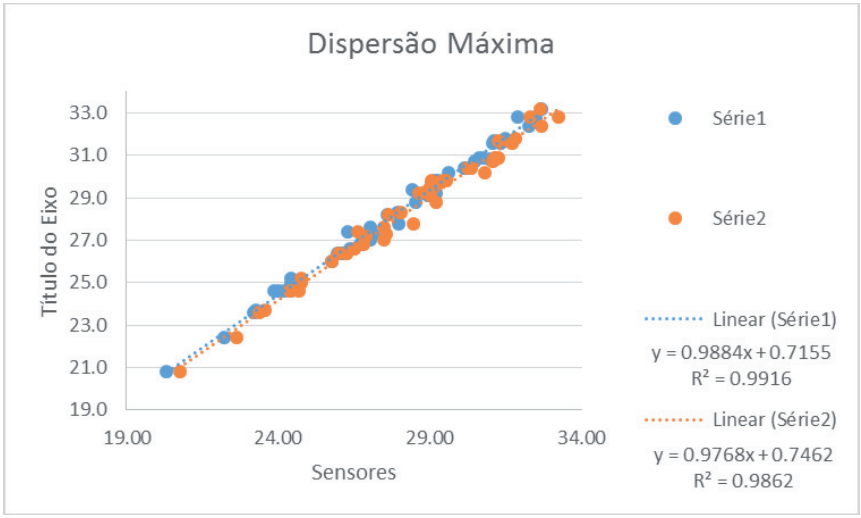


Figura 7. Relação entre as temperaturas máximas diárias os sensores e a estação convencional INMET.

Os sensores apresentaram boa relação nos registros de 15 horas e nas temperaturas máximas e mínimas. Conforme Tabela 3, é possível perceber que as medições possuíam acurácia, pois a sua variação em relação aos dados da estação é inferior a 0,3 °C em ambos os sensores.

Tabela 3. Valores de variância por horário e modelo.

Horário	Variância (°C)	
	BMP180	DS18B20
15 Horas	0,16	0,26
Mínimo	0,10	0,11
Máximo	0,07	0,12

Conclusões

O amostrador autônomo de temperatura de baixo custo foi montado com sucesso, possibilitando o registro de temperatura de maneira automática e confiável. Através da metodologia aplicada neste trabalho, é possível, com outros sensores e componentes, realizar o registro de outras informações, tais como umidade relativa do ar e do solo, precipitação e outras variáveis desejadas. Além disso, existe a possibilidade de envio de informações através da internet diretamente do microcontrolador, eliminando a necessidade da retirada da informação no local onde o amostrador foi inserido, pois as informações, no caso deste projeto, ficam armazenadas em um cartão de memória.

Agradecimentos

Agradecemos ao Conselho Nacional de Desenvolvimento Científico e Tecnológico (CNPq), à Embrapa Milho e Sorgo, à Universidade Federal de São João del-Rei/Campus Sete Lagoas (UFSJ/CSL), ao funcionário da Embrapa Milho e Sorgo Múcio, e à Fundação de Amparo à Pesquisa do Estado de Minas Gerais (Fapemig) pelo apoio para a realização deste trabalho.

Referências

AYOADE, J. O. **Introdução a climatologia para os trópicos**. Rio de Janeiro: Bertrand Brasil, 1996. 322 p.

BARROS, J. P. A.; SOUZA, L. S. B.; MOURA, M. S. B. **Influência das variáveis meteorológicas na temperatura da superfície da caatinga em Petrolina - PE.** Delmiro Gouveia: UFAL, 2014. 4 p.

CARAMORI, P. H. Escopo da biotecnologia vegetal. In: CONGRESSO BRASILEIRO DE BIOMETEOROLOGIA, 4., 2006, Ribeirão Preto. **Mudanças climáticas**: impacto sobre homens, plantas e animais: anais. Ribeirão Preto: Instituto de Zootecnia, 2006.

GOMES, M. L. S. S. **Conceitos, referências e programações básicas com Arduino**. Porto Alegre: SMED, 2014. 83 p.

PEREIRA, A. R.; ANGELOCCI, L. R.; SENTELHAS, P. C. **Meteorologia agrícola**. Piracicaba: ESALQ, 2007. 202 p.

Apêndice A – Script adaptado para coleta de temperatura e pressão

```
// Codigo adpatdo de:
//  Lab de Garagem: http://labdegaragem.com/profiles/blogs/tutorial-reprodu-o-de-cores-com-o-led-rgb-e-o-arduino
//  Seeedstudio: http://www.seeedstudio.com/wiki/Seeeduino_Stalker_v2.3
//  SFE_BMP180 altitude example sketch: https://www.sparkfun.com/products/11824


#include <SdFat.h> //Biblioteca para SDfat
#include <Wire.h> //Biblioteca para relogio
#include <SFE_BMP180.h> //Biblioteca para temperatura e pressao BMP180
#include <OneWire.h> //Biblioteca para temperatura solo

int SensorPin = 4;
OneWire ds(SensorPin);


#define DS3231_I2C_ADDRESS 0x68 //Endereco do RTC - Fixo


const uint8_t SD_CS_PIN = 10; // Endereco do SD

SFE_BMP180 pressure;
#define ALTITUDE 784.0 // Altitude de trabalho
boolean SensorBarometrico = false;

SdFat SD;
File myFile;
SdFile file;


#define LedRED 2 // Define o pino 2 com PWM como RED
#define LedBLUE 5 // Define o pino 5 com PWM como BLUE
#define LedGREEN 3 // Define o pino 6 com PWM como GREEN

long DataSensor = 0;
String NomeArquivoAtual;

// Convert normal decimal numbers to binary coded decimal
byte decToBcd(byte val)
{
 return( (val/10*16) + (val%10) );
}
// Convert binary coded decimal to normal decimal numbers
byte bcdToDec(byte val)
{
 return( (val/16*10) + (val%16) );
}
// You will need to create an SFE_BMP180 object, here called "pressure":

void setDS3231time(byte second, byte minute, byte hour, byte dayOfWeek, byte
dayOfMonth, byte month, byte year)
{
 // sets time and date data to DS3231
 Wire.beginTransmission(DS3231_I2C_ADDRESS);
 Wire.write(0); // set next input to start at the seconds register
 Wire.write(decToBcd(second)); // set seconds
 Wire.write(decToBcd(minute)); // set minutes
 Wire.write(decToBcd(hour)); // set hours
 Wire.write(decToBcd(dayOfWeek)); // set day of week (1=Sunday, 7=Saturday)
 Wire.write(decToBcd(dayOfMonth)); // set date (1 to 31)
 Wire.write(decToBcd(month)); // set month
 Wire.write(decToBcd(year)); // set year (0 to 99)
 Wire.endTransmission();
}
void readDS3231time(byte *second,
byte *minute,
byte *hour,
byte *dayOfWeek,
byte *dayOfMonth,
byte *month,
byte *year){
 Wire.beginTransmission(DS3231_I2C_ADDRESS);
 Wire.write(0); // set DS3231 register pointer to 00h
 Wire.endTransmission();
 Wire.requestFrom(DS3231_I2C_ADDRESS, 7);
 // request seven bytes of data from DS3231 starting from register 00h
 *second = bcdToDec(Wire.read() & 0x7f);
```

```
  *minute = bcdToDec(Wire.read());
  *hour = bcdToDec(Wire.read() & 0x3f);
  *dayOfWeek = bcdToDec(Wire.read());
  *dayOfMonth = bcdToDec(Wire.read());
  *month = bcdToDec(Wire.read());
  *year = bcdToDec(Wire.read());
}

void DataTexto(long *DataDia,String *DataFormatoTexto)
{
 byte second, minute, hour, dayOfWeek, dayOfMonth, month, year;
 // retrieve data from DS3231
 readDS3231time(&second, &minute, &hour, &dayOfWeek, &dayOfMonth, &month,
 &year);

 String TextoData;

 TextoData = String(year,DEC);
 long Data = TextoData.toInt();
 Data = Data*10000;

 TextoData = String(month,DEC);
 Data +=((TextoData.toInt())*100);

 TextoData = String(dayOfMonth,DEC);
 Data +=TextoData.toInt();



 *DataDia = Data;

 // send it to the serial monitor

 TextoData = String(year);
 if(month<10){
  TextoData += "0";
 }
 TextoData+=String(month);
 if(dayOfMonth<10){
  TextoData += "0";
 }
 TextoData+=String(dayOfMonth);

 if(hour<10){
  TextoData += "0";
 }
 TextoData += String(hour, DEC);
 // convert the byte variable to a decimal number when displayed

 if (minute<10)
 {
  TextoData += "0";
 }
 TextoData += String(minute, DEC);


 if (second<10)
 {
  TextoData +="0";
 }
 TextoData += String(second, DEC);

 *DataFormatoTexto = TextoData;


}

void setup() {
 //  Serial.begin(9600);
 Wire.begin();
 pinMode (LedRED, OUTPUT); // Pino 3 declarado como saída
 pinMode (LedBLUE, OUTPUT); // Pino 5 declarado como saída
 pinMode (LedGREEN, OUTPUT); // Pino 6 declarado como saída

 //  while (!Serial) {}  // wait for Leonardo
```

```
  if (!SD.begin(SD_CS_PIN)) {
    //Serial.println("begin failed");
    return;
  }

  if (pressure.begin()){
    //Serial.println("BMP180 init success");
    SensorBarometrico = true;
  }
  else
  {
    // Oops, something went wrong, this is usually a connection problem,
    // see the comments at the top of this sketch for the proper connections.

    SensorBarometrico = false;

    //Serial.println("BMP180 init fail\n\n");
    //    while(1); // Pause forever.
  }

  getTemp(); //Necessario para carregar o sensor
  delay (2000);

  // DS3231 seconds, minutes, hours, day, date, month, year
  //  setDS3231time(10,24,15,6,27,02,15);

}
//--------------------------------------------------------------------------------

void loop() {
  long DiaAno;
  long TamanhoDoArquivo;
  long TamanhoDoArquivoD;
  String RecebeDataFormatadaTexto;
  DataTexto(&DiaAno,&RecebeDataFormatadaTexto);
  int Erro = 0;
  TamanhoDoArquivo=0;
  TamanhoDoArquivoD=0;

  String LinhaDados = "";

  if(DataSensor!=DiaAno){
    String NomeArquivo = "Embrapa-MS-Sensor_"+RecebeDataFormatadaTexto+".TXT ";
    NomeArquivoAtual = NomeArquivo;
    char fileNameCharArray[NomeArquivo.length()];
    NomeArquivo.toCharArray(fileNameCharArray, NomeArquivo.length());
    //file.size()
    file.open(fileNameCharArray, O_RDWR | O_CREAT | O_AT_END);


file.println("Entrada:LeituraAltitude(m);temperatura(grausC);pressaoAbsoluta(mb);Pressao(nivelMar)Relativa(mb);AltitudeComputada(m);
TemperaturaSolo(grausC);Horario;");

    analogWrite (LedRED, 255);
    delay(200);
    analogWrite (LedRED, 0);
    delay(200);
    analogWrite (LedRED, 255);
    delay(200);
    analogWrite (LedRED, 0);
    delay(200);
    analogWrite (LedRED, 255);
    delay(200);
    analogWrite (LedRED, 0);

    //Serial.println("iniciou o arquivo");

    DataSensor=DiaAno;

  }
  else{
    String NomeArquivo = NomeArquivoAtual;
    char fileNameCharArray[NomeArquivo.length()];
    NomeArquivo.toCharArray(fileNameCharArray, NomeArquivo.length());
    //Serial.println("Tamanho arquivo");
```

```
  myFile = SD.open(fileNameCharArray, FILE_WRITE);
  TamanhoDoArquivo=myFile.size();
  myFile.close();

  file.open(fileNameCharArray, O_RDWR | O_CREAT | O_AT_END);

}


if(SensorBarometrico){

  char status;
  double T,P,p0,a;

  // Loop here getting pressure readings every 10 seconds.

  // If you want sea-level-compensated pressure, as used in weather reports,
  // you will need to know the altitude at which your measurements are taken.
  // We're using a constant called ALTITUDE in this sketch:


  LinhaDados += (ALTITUDE);
  LinhaDados += ";";


  // If you want to measure altitude, and not pressure, you will instead need
  // to provide a known baseline pressure. This is shown at the end of the sketch.

  // You must first get a temperature measurement to perform a pressure reading.

  // Start a temperature measurement:
  // If request is successful, the number of ms to wait is returned.
  // If request is unsuccessful, 0 is returned.

  status = pressure.startTemperature();
  if (status != 0)
  {
    // Wait for the measurement to complete:
    delay(status);

    // Retrieve the completed temperature measurement:
    // Note that the measurement is stored in the variable T.
    // Function returns 1 if successful, 0 if failure.

    status = pressure.getTemperature(T);
    if (status != 0)
    {
      // Print out the measurement:

      LinhaDados += (T);
      LinhaDados += ";";
      //        Serial.print(T,2);
      //        Serial.print(";");

      // Start a pressure measurement:
      // The parameter is the oversampling setting, from 0 to 3 (highest res, longest wait).
      // If request is successful, the number of ms to wait is returned.
      // If request is unsuccessful, 0 is returned.

      status = pressure.startPressure(3);
      if (status != 0)
      {
        // Wait for the measurement to complete:
        delay(status);

        // Retrieve the completed pressure measurement:
        // Note that the measurement is stored in the variable P.
        // Note also that the function requires the previous temperature measurement (T).
        // (If temperature is stable, you can do one temperature measurement for a number of pressure measurements.)
        // Function returns 1 if successful, 0 if failure.

        status = pressure.getPressure(P,T);
        if (status != 0)
        {
          // Print out the measurement:
```

```
        LinhaDados += (P);
        LinhaDados += ";";


        // The pressure sensor returns abolute pressure, which varies with altitude.
        // To remove the effects of altitude, use the sealevel function and your current altitude.
        // This number is commonly used in weather reports.
        // Parameters: P = absolute pressure in mb, ALTITUDE = current altitude in m.
        // Result: p0 = sea-level compensated pressure in mb

        p0 = pressure.sealevel(P,ALTITUDE);

        LinhaDados += (p0);
        LinhaDados += ";";


        // On the other hand, if you want to determine your altitude from the pressure reading,
        // use the altitude function along with a baseline pressure (sea-level or other).
        // Parameters: P = absolute pressure in mb, p0 = baseline pressure in mb.
        // Result: a = altitude in m.
        a = pressure.altitude(P,p0);

        LinhaDados += (a);
        LinhaDados += ";";


       }
       else{
        LinhaDados += "Sensor Barometrico sem funcionamento Pressao Inicio;";
        Erro=1;
       }
      }
      else{
       LinhaDados += "Sensor Barometrico sem funcionamento Pressao Inicio;";
       Erro=1;
      }
     }
     else{
      LinhaDados += "Sensor Barometrico sem funcionamento Temperatura Recebendo Valores;";
      Erro=1;
     }
    }
    else{
     LinhaDados += "Sensor Barometrico sem funcionamento Temperatura Inicio;";
     Erro=1;
    }
   }
   else{
    LinhaDados += "Sensor Barometrico sem funcionamento;";
    Erro=1;
   }

   float tempe = getTemp();
   LinhaDados += (tempe);
   LinhaDados += ";";

   if(tempe<-55){
    Erro=1;
   }

   LinhaDados += RecebeDataFormatadaTexto+";";


   file.println(LinhaDados);
   file.close();



   int numFiles = 0;

   if (!SD.begin(SD_CS_PIN)) {
    analogWrite (LedRED, 255);
    analogWrite (LedGREEN, 255);
    analogWrite (LedBLUE, 255);
    delay(2000);
```

```
    analogWrite (LedRED, 0);
    analogWrite (LedGREEN, 0);
    analogWrite (LedBLUE, 0);
   }

   while (file.openNext(SD.vwd(), O_READ)) {
    uint32_t sizefiletest;
    sizefiletest = file.fileSize();
    if ((int)sizefiletest>0){
     numFiles++;
    }
    file.close();
   }



   int CorLed = LedRED;
   if(numFiles%2==1){
    CorLed = LedGREEN;
   }
   else{
    CorLed = LedBLUE;
   }

   if(Erro==1){
    analogWrite (LedRED, 255);
    analogWrite (LedGREEN, 255);
    analogWrite (LedBLUE, 255);
    delay(2000);
    analogWrite (LedRED, 0);
    analogWrite (LedGREEN, 0);
    analogWrite (LedBLUE, 0);
   }
   else{
    analogWrite (CorLed, 255);
    delay(200);
    analogWrite (CorLed, 0);
    delay(200);
    analogWrite (CorLed, 255);
    delay(200);
    analogWrite (CorLed, 0);
   }

   String NomeArquivo = NomeArquivoAtual;
   char fileNameCharArray[NomeArquivo.length()];
   NomeArquivo.toCharArray(fileNameCharArray, NomeArquivo.length());
   //Serial.println("Tamanho arquivo");

   myFile = SD.open(fileNameCharArray, FILE_WRITE);
   TamanhoDoArquivoD=myFile.size();
   myFile.close();

   if(TamanhoDoArquivo==TamanhoDoArquivoD){
    analogWrite (LedRED, 255);
    delay(2000);
    analogWrite (LedRED, 0);
   }else{
    analogWrite (LedGREEN, 255);
    analogWrite (LedBLUE, 255);
    delay(2000);
    analogWrite (LedGREEN, 0);
    analogWrite (LedBLUE, 0);
   }



   delay(5000);
}

float getTemp(){
  byte data[12];
  byte addr[8];

  if ( !ds.search(addr)) {
    //no more sensors on chain, reset search
```

```
    ds.reset_search();
    return -1000;
  }

  if ( OneWire::crc8( addr, 7) != addr[7]) {
    //Serial.println("CRC is not valid!");
    return -1000;
  }

  if ( addr[0] != 0x10 && addr[0] != 0x28) {
    //Serial.print("Device is not recognized");
    return -1000;
  }

  ds.reset();
  ds.select(addr);
  ds.write(0x44,1);

  byte present = ds.reset();
  ds.select(addr);
  ds.write(0xBE);


  for (int i = 0; i < 9; i++) {
    data[i] = ds.read();
  }

  ds.reset_search();

  byte MSB = data[1];
  byte LSB = data[0];

  float TRead = ((MSB << 8) | LSB );
  //              float Temperature = TRead / 16;

  return TRead / 16;

}
```

Apêndice B – Arquivo texto com coleta de informação dos primeiros 5 minutos a cada 5 segundos do dia 5 de junho de 2016

Entrada:LeituraAltitude(m);temperatura(grausC);pressaoAbsoluta(mb);Pressao(nivelMar)Relativa(mb);AltitudeComputada(m);Temperatura Solo(grausC);Horario;
784.00;17.34;930.57;1022.05;784.00;17.69;160605000006;
784.00;17.31;930.54;1022.02;784.00;17.69;160605000015;
784.00;17.34;930.54;1022.02;784.00;17.69;160605000022;
784.00;17.33;930.55;1022.03;784.00;17.69;160605000030;
784.00;17.33;930.57;1022.05;784.00;17.69;160605000038;
784.00;17.33;930.52;1022.00;784.00;17.69;160605000046;
784.00;17.34;930.60;1022.09;784.00;17.69;160605000054;
784.00;17.32;930.56;1022.04;784.00;17.69;160605000102;
784.00;17.32;930.57;1022.05;784.00;17.69;160605000109;
784.00;17.31;930.57;1022.05;784.00;17.69;160605000117;
784.00;17.34;930.61;1022.09;784.00;17.69;160605000125;
784.00;17.33;930.59;1022.07;784.00;17.69;160605000133;
784.00;17.32;930.57;1022.05;784.00;17.69;160605000141;
784.00;17.32;930.57;1022.05;784.00;17.69;160605000148;
784.00;17.32;930.50;1021.97;784.00;17.69;160605000156;
784.00;17.32;930.60;1022.08;784.00;17.69;160605000204;
784.00;17.32;930.60;1022.09;784.00;17.69;160605000212;
784.00;17.31;930.57;1022.05;784.00;17.69;160605000220;
784.00;17.31;930.57;1022.05;784.00;17.69;160605000227;
784.00;17.31;930.57;1022.05;784.00;17.69;160605000235;
784.00;17.31;930.58;1022.07;784.00;17.69;160605000243;
784.00;17.31;930.56;1022.04;784.00;17.69;160605000251;
784.00;17.30;930.61;1022.10;784.00;17.69;160605000259;
784.00;17.31;930.64;1022.13;784.00;17.69;160605000306;
784.00;17.32;930.61;1022.10;784.00;17.69;160605000314;
784.00;17.31;930.57;1022.05;784.00;17.69;160605000322;
784.00;17.32;930.66;1022.15;784.00;17.69;160605000330;
784.00;17.31;930.61;1022.10;784.00;17.69;160605000338;
784.00;17.32;930.63;1022.12;784.00;17.75;160605000346;
784.00;17.30;930.60;1022.08;784.00;17.75;160605000353;
784.00;17.32;930.65;1022.14;784.00;17.69;160605000401;
784.00;17.32;930.62;1022.11;784.00;17.75;160605000409;
784.00;17.31;930.60;1022.08;784.00;17.75;160605000417;
784.00;17.31;930.63;1022.12;784.00;17.75;160605000425;
784.00;17.33;930.66;1022.16;784.00;17.75;160605000432;
784.00;17.30;930.56;1022.04;784.00;17.75;160605000440;
784.00;17.32;930.56;1022.04;784.00;17.75;160605000448;
784.00;17.31;930.61;1022.10;784.00;17.75;160605000456;
784.00;17.31;930.61;1022.09;784.00;17.75;160605000504;
784.00;17.31;930.64;1022.12;784.00;17.75;160605000511;
784.00;17.32;930.63;1022.12;784.00;17.75;160605000519;
784.00;17.31;930.64;1022.13;784.00;17.75;160605000527;
784.00;17.32;930.65;1022.14;784.00;17.75;160605000535;
784.00;17.31;930.61;1022.09;784.00;17.75;160605000543;
784.00;17.32;930.67;1022.16;784.00;17.75;160605000550;
784.00;17.31;930.61;1022.10;784.00;17.75;160605000558;

CGPE - 13655