2016

TUTORIAL ALTÍMETRO LAE-P

FABIO MAURICIO MATOS

LAE - UFPR

14/03/2016

TUTORIAL PARA UTILIZAÇÃO DO ALTÍMETRO LAE-P

O altímetro LAE-P é um altímetro baseado na placa de prototipagem Arduino. Ele foi desenvolvido no Laboratório de Atividades Espaciais (LAE) da UNIVERSIDADE FEDERAL DO PARANÁ (UFPR). A letra P é uma referência ao Primeiro altímetro do LAE, que ainda está na fase de protótipo.

Esse primeiro protótipo baseado no Arduino funciona como um kit de montagem. Cada sensor criado para o Arduino tem sua própria biblioteca, facilitando a implementação do código.

1. Componentes

1 Arduino pro mini 3,3 V

1 sensor bmp 180

1 módulo mini SD card

1 mini SD card

1 adaptador de gravação para o Arduino pro mini

1 pilha A23 e suporte

2. Ligação bmp-180

O esquema de ligação do sensor bmp 180 (Figura 1) é mostrado na Tabela 1 .

BMP 180	ARDUINO
SDA	A5
SCL	A4
GND	GND
VCC	3,3V

Tabela 1. Ligação entre o sensor bmp 180 e o Arduino

Fig. 1 Sensor BMP-180

3. LIGAÇÃO DO MÓDULO MINI SD CARD

Para o módulo mini SD card a ligação será a mesma para qualquer modelo de módulo SD card. Aqui vou indicar um modelo, mas você pode usar o que encontrar.

SD CARD	ARDUINO
CS	9
MOSI	11
MISO	12
SCK	13
GND	GND
VCC	3,3V

Tab.2 ligação entre o módulo sd card e o Arduino

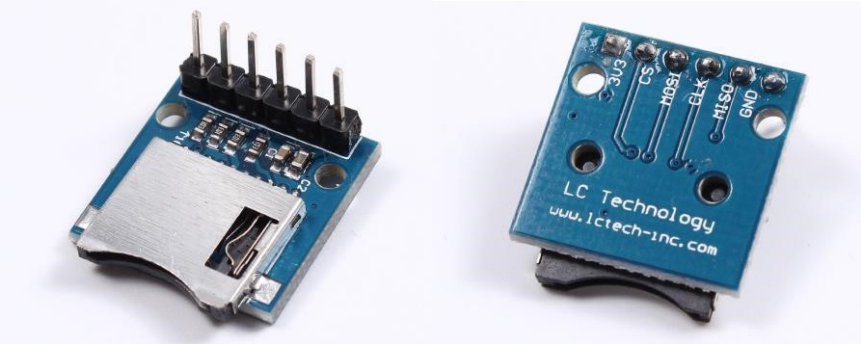

Fig. 2 módulo sd card indicado

http://produto.mercadolivre.com.br/MLB-731957017-leitor-gravador-micro-sd-p-arduinoraspberry-nodemcu-_JM

4. LIGAÇÃO DA BATERIA

Nesse projeto usamos uma bateria de 12V (A23). Esse modelo tem um tamanho e massa bem reduzidos.
A vantagem é fazer minifoguetes mais finos e leves. A desvantagem é que essa bateria fornece um pequena corrente; apenas 55 mAh.
O positivo da bateria deve ser ligado no pino **RAW** e o negativo no pino **GND** ao lado.

Fig.3 Pilha A23 e seu suporte

5. MONTAGEM

A montagem depende do usuário e da habilidade de solda. No protótipo LAE-P tentei deixar tudo o mais próximo possível de cada pino.
A configuração de montagem é livre. Devem ser mantidos apenas os pinos de ligação.
As figuras abaixo mostram uma opção de montagem.

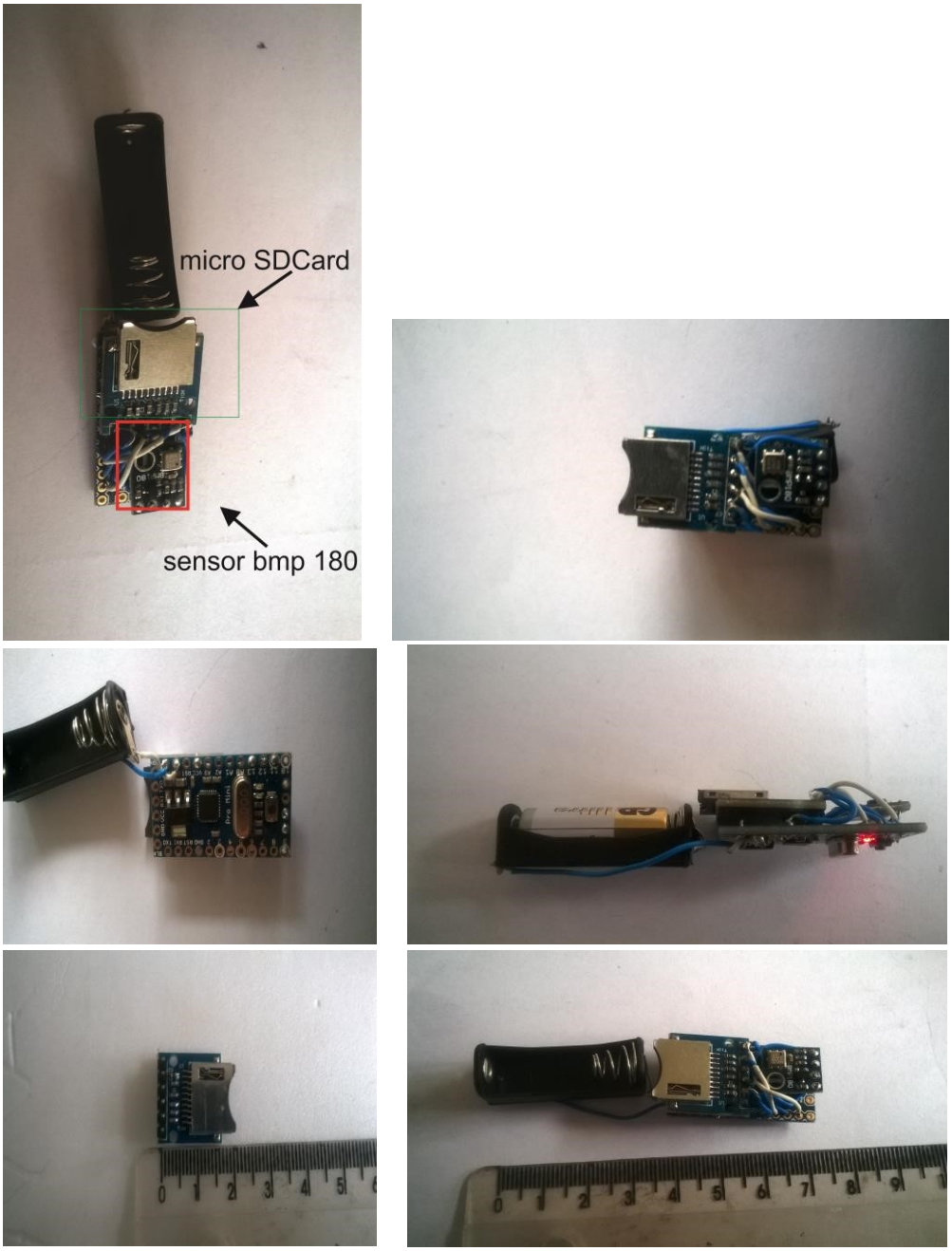

Figura 4. O altímetro LAE-P.

6. SOFTWARE PARA FUNCIONAMENTO DO ARDUINO

De posse da placa Arduino para prototipagem, necessitamos do software para escrever o código e fazer a compilação.

No link https://www.arduino.cc/en/Main/Software você pode fazer o download do software na versão atual.

Após precisamos instalar as bibliotecas para o funcionamento do sensor.

As bibliotecas são sub-rotinas preparadas especificamente para fazer a ligação do sensor com a placa Arduino. Assim, não precisamos nos preocupar com a configuração entre os componentes e podemos nos dedicar ao código e aos dados que queremos obter com o código.

6.1 INSTALAÇÃO DAS BIBLIOTECAS NO ARDUINO

Para o funcionamento do módulo são necessárias algumas bibliotecas.

Para esse Tutorial vamos usar a biblioteca bmp 085, bmp 180 e a biblioteca I2Cdev. Estas bibliotecas são facilmente encontradas na internet pelo próprio nome de chamada.

Após descompactar os arquivos de biblioteca, devemos copiar a pasta descompactada e colar na pasta libraries. em:

c:arquivos de programs(x86)/Arduino/libraries

Apenas após esse passo deve-se abrir o compilador Arduino. Se o compilador estava aberto nesse passo feche e abra novamente.

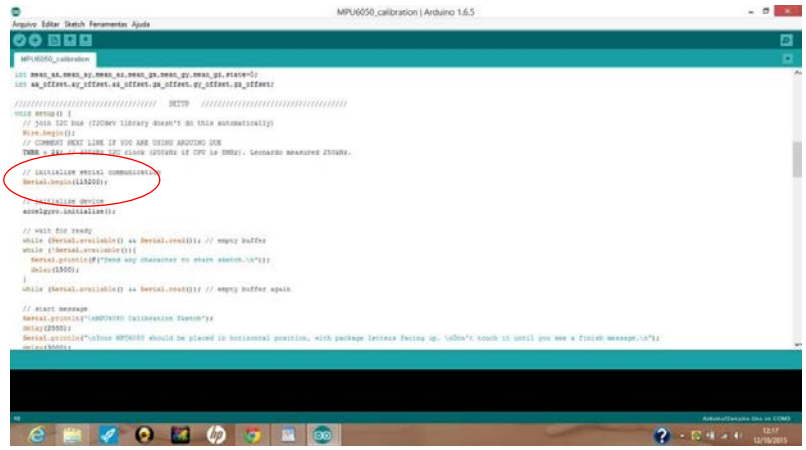

Figura 5. Linha de comando e velocidade de leitura da porta serial.

Confira na linha que contém o comando serial.begin(9600). Essa é a velocidade leitura da porta serial.

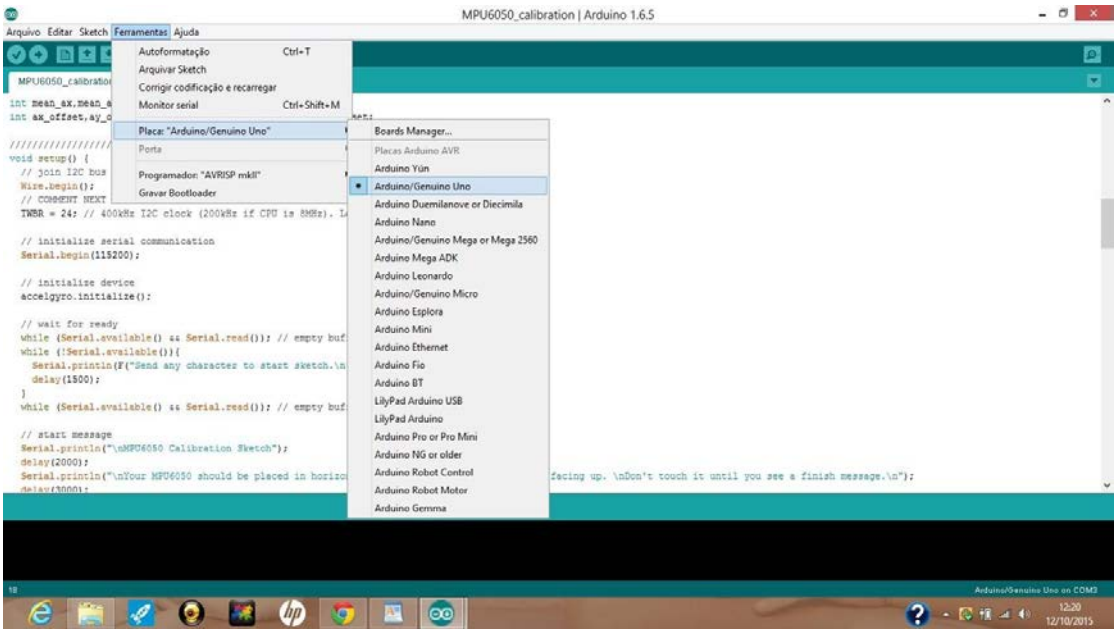

Figura 6. Seleção da placa Arduino disponível para o projeto.

A Figura 6 mostra como selecionar o modelo do Arduino que você está usando. Para o caso de um Arduino pro mini devemos escolher o mesmo nessa aba. Para esse tutorial usaremos o Arduino uno.

6.2 COMPILANDO

Após escrever ou carregar o código devemos verificar o código clicando no botão verificar no canto superior esquerdo.

Concluída a verificação do código e não havendo erros de compilação, deve-se verificar a porta de comunicação.

Para o Arduino UNO usamos a porta com3. Verifique se há outra porta livre. Se não houver use a que estiver disponível.

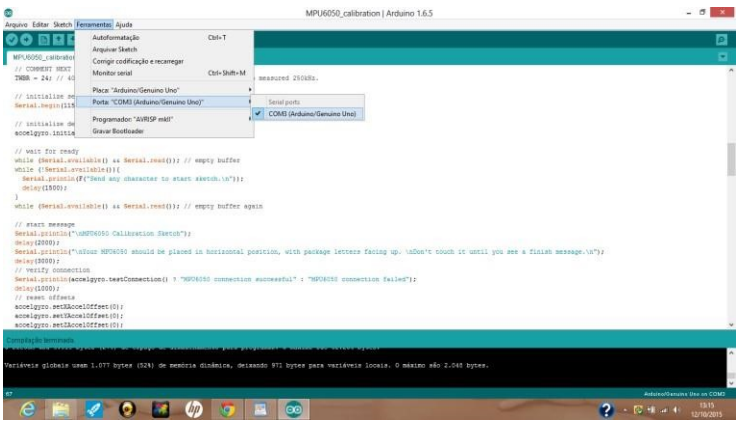

Figura 7. Seleção da porta.

Agora devemos carregar o código para o Arduino clicando no botão carregar no canto superior esquerdo ao lado do botão verificar.

Após a mensagem de que o código está carregado, devemos abrir a porta serial para visualização dos dados, clicando no ícone da lupa no canto superior direito da tela.

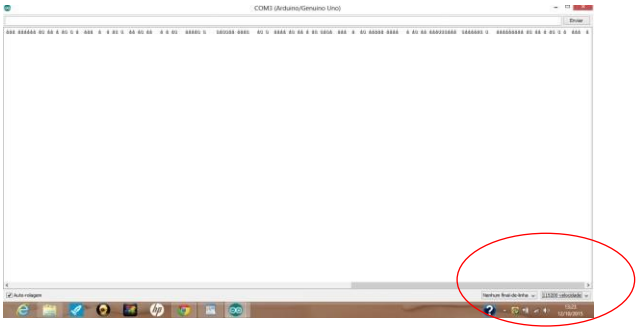

Fig8. Janela serial com velocidade incompatível com a linha de comando serial.begin

7. Leitura dos dados

Se as ligações estiverem corretas e o código carregado, podemos ligar nosso altímetro LAE-P, apenas colocando a pilha.

O altímetro só funciona com o sd card inserido. Após ligarmos veremos um led piscar no Arduino na mesma frequência da leitura dos dados. Enquanto o led estiver piscando o altímetro está colhendo dados.

Os dados são gravados em um arquivo chamado laedata.txt. Recomendo que a cada lançamento descarregue os dados ou renomeie o arquivo.

O código não muda o nome do arquivo a cada lançamento; ele salva no mesmo arquivo. Então se você fizer dois lançamentos no mesmo cartão sem renomear os dados dos dois lançamentos, eles estarão no mesmo arquivo, um após o outro. Para identificá-los você terá que localizar o ponto em que o tempo reinicia.

8. Código

```
#include <SD.h>
#include<SPI.h>

#include "Wire.h"
#include "Adafruit_BMP085.h"
#include "I2Cdev.h"
#include "MPU6050.h"
#include <SFE_BMP180.h>
```

```
Adafruit_BMP085 mySensor; //cria um sensor chamado mysensor
Adafruit_BMP085 bmp;
#define P0 91971
#define LED_PIN 9

SFE_BMP180 pressure;  //bmp 180
float tempC; //variável de temperatura em C
float pressure2; // variável de leitura de pressão
double h, d;
double a_ant;

unsigned long time;
double getPressure(); //bmp180
double baseline; //bmp180

int ledPin = 8; // ejetor
int ledPin2 = 3;

int chipSelect = 4;

bool blinkState = false;

File mySensorData;

//---------------------------------------------------------------------

void setup(){
  a_ant = 0;
  pinMode(ledPin, OUTPUT);
   pinMode(ledPin2, OUTPUT);

  Wire.begin();

Serial.begin(9600);
mySensor.begin();
Serial.println("Initializing I2C devices...");

  if (pressure.begin())   //bmp180
    Serial.println("BMP180 init success");
  else
  {

    Serial.println("BMP180 init fail (disconnected?)\n\n");
    while(1); // Pause
  }                       // bmp180
```

```
  baseline = getPressure(); // bmp180

  //
  pinMode(LED_PIN, OUTPUT);
pinMode(10, OUTPUT); // reserve
SD.begin(4); //Initializa the SD card reader
}

void loop() {

double a,P;
   double b;

P = getPressure();

tempC = mySensor.readTemperature(); //  0

time = millis();

digitalWrite(ledPin2, HIGH);
  delay(20);
  digitalWrite(ledPin2, LOW);
  delay(20);

  blinkState = !blinkState;

 digitalWrite(LED_PIN, HIGH);
  delay(5);
  digitalWrite(LED_PIN, LOW);
  delay(5);

  a = pressure.altitude(P,baseline);

  b= a-a_ant;
mySensorData = SD.open("laeData.txt", FILE_WRITE);
if (mySensorData) {
Serial.print(time); Serial.print("\t");

  Serial.print(tempC); Serial.print("\t");
  Serial.print(P); Serial.print("\t");
     Serial.print(a); Serial.print("\t");
      a_ant = a;

  if ( (b < 0) && (a >30) ) digitalWrite(ledPin, HIGH),
  Serial.println("apogeu"),
```

```
 Serial.print(a_ant);
 Serial.println("");
 delay(5);

mySensorData.print(time); mySensorData.print("\t");
mySensorData.print(tempC);
mySensorData.print("\t");
mySensorData.print(P);
mySensorData.print("\t");
mySensorData.println(a);

mySensorData.close();
}
}
// fim; linhas abaixo copiadas da biblioteca (não apagar senão falha o getpressure----------
--------------------------------------------------------------------------------------
double getPressure()
{
 char status;
 double T,P,p0,a;

 // 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.
  // Use '&T' to provide the address of T to the function.
  // Function returns 1 if successful, 0 if failure.

  status = pressure.getTemperature(T);
  if (status != 0)
  {
   // 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.
     // Use '&P' to provide the address of 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)
     {
       return(P);
     }
     else Serial.println("error retrieving pressure measurement\n");
    }
    else Serial.println("error starting pressure measurement\n");
  }
  else Serial.println("error retrieving temperature measurement\n");
 }
 else Serial.println("error starting temperature measurement\n");
}
```

9. Contatos e condições de uso

Não há custo nem restrição ao uso ou melhoria do altímetro LAE-P mas peço que em caso do leitor fazer alguma melhoria entre em contato comigo para dividi-la.

Mande-me seu e-mail caso tenha interesse em receber as atualizações do código.

Para ajudar no desenvolvimento desse altímetro, por favor, mande-me dados dos voos realizados para estudos e melhorias.

E-mail: fabiogtz2@hotmail.com

O primeiro uso bem-sucedido do altímetro LAE-P foi divulgado em

http://fogueteufpr.blogspot.com.br/2016/03/lae-p-nosso-primeiro-altimetro-para.html

Revisão: Marchi, 20 Mar 2016