

UNIVERSIDADE FEDERAL DE OURO PRETO - UFOP  
ESCOLA DE MINAS

DEPARTAMENTO DE ENGENHARIA DE CONTROLE E AUTOMAÇÃO

PROJETO PRO-ATIVA

**Tutorial para criar um sistema de aquisição de dados remoto,  
alimentado por células fotovoltaicas**

Autor: Saulo Neves Matos

Orientadora: Adrielle de Carvalho Santana

Ouro Preto/MG

2015

## Resumo

Este é um tutorial cujo objetivo é ensinar a criar um sistema de aquisição de dados capaz de monitorar e registrar informações remotamente, ou seja, desenvolvido para o uso de um sistema telemétrico nas mais diversas aplicações e projetos acadêmicos tais como: monitoramento de aeronaves rádio controladas; estações meteorológicas, de sistemas hidráulicos e térmicos e de plantas industriais que, em geral, envolvam risco ao operador, fazendo necessário o uso de uma comunicação sem fio. Para seu desenvolvimento, utilizou-se a plataforma Arduino, devido ao seu baixo custo e fácil programação, e por possuir diversos sensores e *Shields*, o que pode ajudar discentes sem muita experiência com a área de eletrônica, além de possibilitar a adaptação do projeto às mais diversas aplicações. Foram usados dois tipos de sensores: temperatura (LM35) e vazão (HS05), e todos os dados medidos pelos sensores foram dispostos em um sistema supervisório. Além disso, o tutorial ensina a utilizar um módulo rádio frequência (RF) para que a comunicação entre o Arduino e o computador seja sem fio e com um vasto raio de cobertura. Por fim, este tutorial também ensina como utilizar células fotovoltaicas para a alimentação de todo o sistema de forma que este funcione de forma autônoma. Agradecemos a oportunidade ao projeto Pró-Ativa.

**Palavras-chave:** Arduino, temperatura, vazão, fotovoltaica, sensores, supervisório, wireless.

## **Abstract**

This is a tutorial aimed at teaching how to create a data acquisition system that can monitor and log information remotely, i.e., developed for the use of a telemetric system in several applications and academic projects such as monitoring controlled radio aircraft, weather stations, thermal and hydraulic systems of industrial plants and, in general, involve a risk for the operator, making necessary the use of wireless communication. For its development, we used the Arduino platform due to its low cost and easy programming, and have multiple sensors and Shields, which can help students with little experience in the area of electronics, in addition to enabling the adaptation of the project to various applications. They used two types of sensors: temperature (LM35) and flow (HS05), and all data measured by the sensors are arranged in a supervisory system. Moreover, the tutorial teaches using a radio frequency module (RF) so that communication between the Arduino and the computer is wireless and a wide area of coverage. Finally, this tutorial also explains how to use photovoltaic cells to power the whole system so that it functions autonomously. We appreciate the opportunity to Pró-Ativa.

**Keywords** Arduino, temperature, flow , photovoltaic, sensors, supervisory, wireless.

## Sumário

<b>1. Introdução .....</b>	<b>5</b>
1.1. Materiais utilizados .....	6
<b>2. Configurações do Arduino .....</b>	<b>7</b>
2.1. Hardware .....	7
2.2. Software .....	9
2.3. Conceitos básicos .....	13
2.3.1. Funções base .....	13
2.3.2. Sintaxe .....	13
2.3.3. Gravando e compilando .....	14
2.3.4. Comunicação serial .....	15
<b>3. Sensores .....</b>	<b>16</b>
3.1. Sensor de temperatura .....	16
3.2. Sensor de fluxo .....	19
<b>4. Comunicação remota .....</b>	<b>24</b>
<b>5. Sistema de supervisão .....</b>	<b>27</b>
5.1. Microsoft Office Excel .....	27
5.2. MakerPlot .....	39
<b>6. Alimentação .....</b>	<b>48</b>
<b>7. Conclusão .....</b>	<b>70</b>
<b>8. Referencias .....</b>	<b>71</b>

## 1. Introdução

Um sistema telemétrico é uma aplicação que permite o monitoramento, de diversas variáveis, podendo-se controlar, medir ou rastrear algum processo. A comunicação depende da aplicação, ou seja, existem situações em que a transmissão é cabeada e em que é necessário o uso de uma transmissão sem fio, a qual é normalmente mais utilizada. É usada em processos que existe risco ao operador ou o impossibilita de acompanhar o sistema de perto. Alguns exemplos são o monitoramento de aeronaves; de sistemas hidráulicos e térmicos; estações meteorológicas; veículos de corrida e de plantas industriais.

Para o desenvolvimento do tutorial capaz de ensinar a desenvolver tal sistema, foram usados a plataforma Arduino, devido ao seu baixo custo e fácil programação, e por possuir diversos sensores e *Shields*, o que pode ajudar discentes sem muita experiência com a área de eletrônica.

A plataforma Arduino foi criada em 2005 na Itália tendo como base o conceito de software e hardware livre, ou seja, o seu uso está aberto para uso e contribuição de toda a sociedade. A placa comunica com um computador hospedeiro de forma serial ou USB, e é basicamente composta por pinos de entrada/saída analógica ou digital e um microcontrolador, o ATMEL AVR. A programação da placa é feita por meio de uma sintaxe de fácil compreensão, similar as linguagens C e C++, além disso existe uma vasta quantidade de bibliotecas com funções que facilitam a interação do usuário com o dispositivo.

Sendo assim, Arduino pode ser entendido como uma unidade de processamento capaz de mensurar diversas variáveis tais como temperatura, vazão, velocidade, pressão entre outras. Tais variáveis são transformadas em sinais elétricos por meio de sensores ligados nos terminais de entrada do Arduino.

### 1.1.Materiais Utilizados

Foram utilizados os seguintes componentes para realização do protótipo:

- Arduino uno
- Cabo USB
- Sensor de temperatura LM35
- Sensor de vazão HS05
- Modulo RF transmissor sem fio (FT232 USB serial)
- Protoboard
- Jumpers
- Ferro de solda
- Estanho em fio
- Dois diodos 1N4007
- Duas placas fotovoltaicas

## 2. Configurações do Arduino

### 2.1. Hardware

O Arduino Uno tem 14 pinos de entrada/saída digital (dos quais 6 podem ser usados como saídas PWM), 6 entradas analógicas, um cristal oscilador de 16MHz, uma conexão USB, uma entrada de alimentação, uma conexão ICSP e um botão de reset. A tabela 2.1 lista todas as características da placa:

*Tabela 2.1 - Características Arduino*

Microcontrolador	ATmega328
Tensão Operacional	5V
Tensão de entrada recomendada	7-12V
Tensão de entrada limite	6-20V
Pinos E/S digitais	14
Pinos de entrada analógica	6
Corrente CC por pino E/S	40 mA
Corrente CC para o pino 3,3V	50 mA
Flash Memory	32 KB
SRAM	2 KB
EEPROM	1 KB
Velocidade de Clock	16 MHz

A alimentação da placa pode ser feita tanto pelo USB ou por qualquer fonte de bateria externa. Caso seja usada uma bateria adicional a tensão deve ser de 6 a 20v. Para realizar a conexão deve-se ligar a bateria no pino Vin (tensão de entrada) e GND (terra) ou ligá-la diretamente no plug P4 existente no Arduino. A figura 2.1 (a) ilustra estas entradas de alimentação.

Além do Vin e do GND existem outros pinos de alimentação usados para conexão de Shields e sensores. O IOREF, o qual fornece uma tensão de referência para que *shields* possam selecionar o tipo de interface apropriada, dessa forma *shields* que funcionam com a placas Arduino que são alimentadas com 3,3V. podem se adaptar para ser utilizados em 5V. e vice-versa. O 3,3 V que fornece tensão de 3,3V para alimentação de *shields* e

módulos externos, e o 5 V que fornece tensão de 5 V. Os conectores USB e de alimentação estão mostrados na figura 2.1.

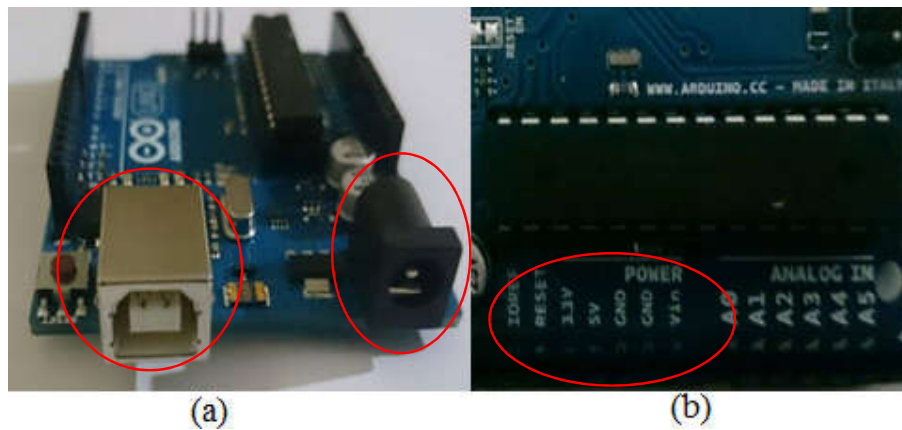


Figura 2.1 – (a) USB, Plug P4 e (b) conectores de alimentação

Conforme exibido na figura 2.2, a placa Arduino UNO possui 14 pinos que podem ser usados como entrada ou saída digitais. Alguns desses pinos possuem funções especiais: os pinos 3,5,6,9,10 e 11 podem ser usados como saídas PWM de 8 bits através da função `analogWrite()`; os 0 e 1 podem ser utilizados para comunicação serial, e os 2 e 3 podem ser configurados para gerar uma interrupção externa, através da função `attachInterrupt()`, função que será usada na programação do sensor de vazão.

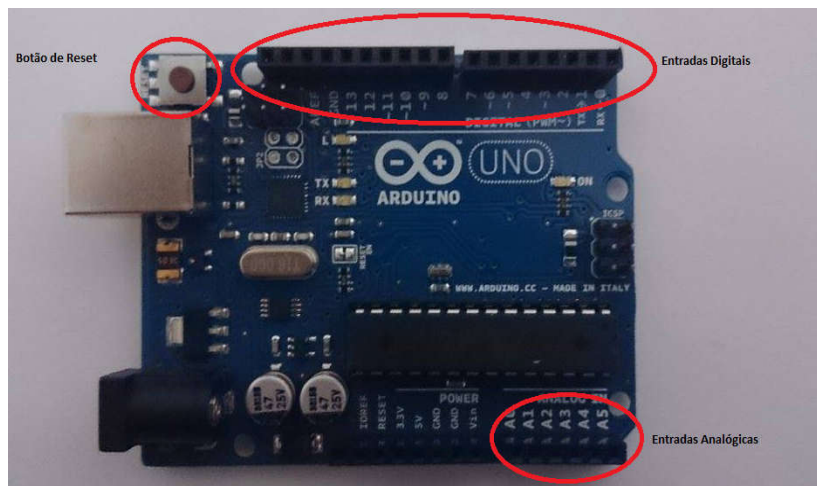


Figura 2.2 - Entradas digitais e analógicas



## 2.2. Software

Todo código de programação será escrito em uma IDE (*Integrated Development Environment*) do Arduino, em que é possível a criação de sketches. Assim é necessário que se faça o download do software disponível no site [www.arduino.cc](http://www.arduino.cc).

Para isso, basta clicar na aba “download”, escolher o sistema operacional utilizado pelo usuário, que se iniciará o download do arquivo, como mostra a figura 2.3.

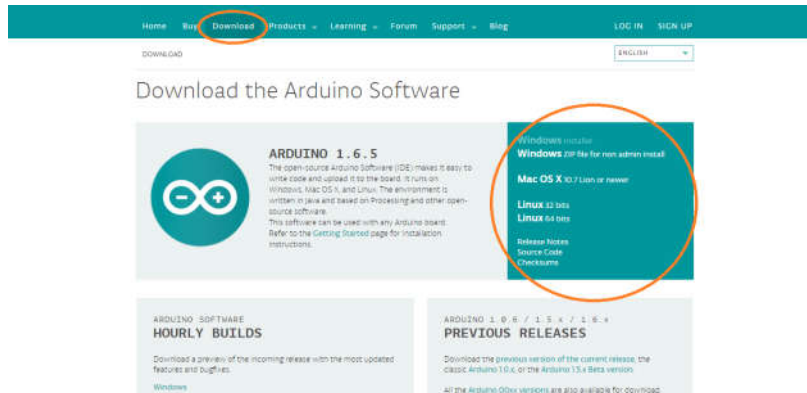


Figura 2.3 - Download da IDE

No caso da instalação no Windows, após o download deve-se clicar no arquivo para que se inicie a instalação. Irá abrir uma janela, mostrada na figura 2.4, com a licença do software a ser instalado, para prosseguir basta clicar em “I Agree”.

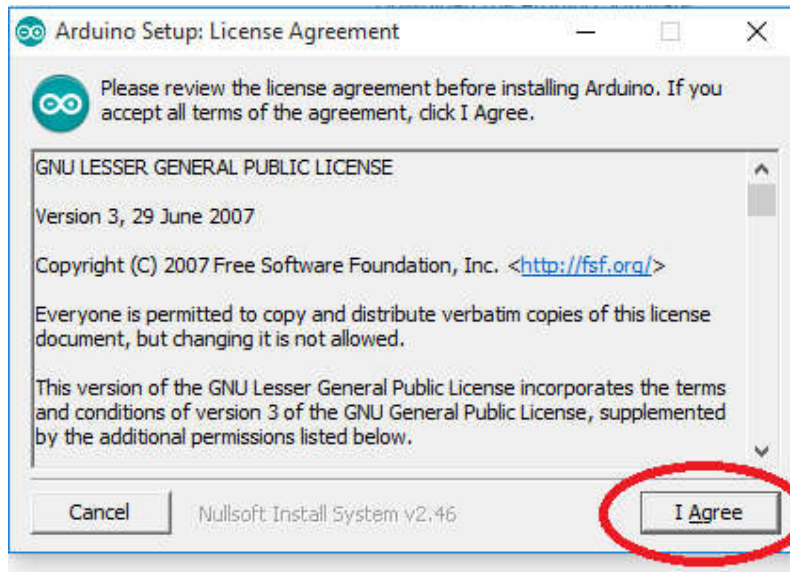


Figura 2.4 - Instalação da IDE

Após esse passo, abrirá uma janela contendo as opções de instalação, para que o usuário escolha quais arquivos serão instalados, recomenda-se deixar todas as opções marcadas e clicar em “Next”.

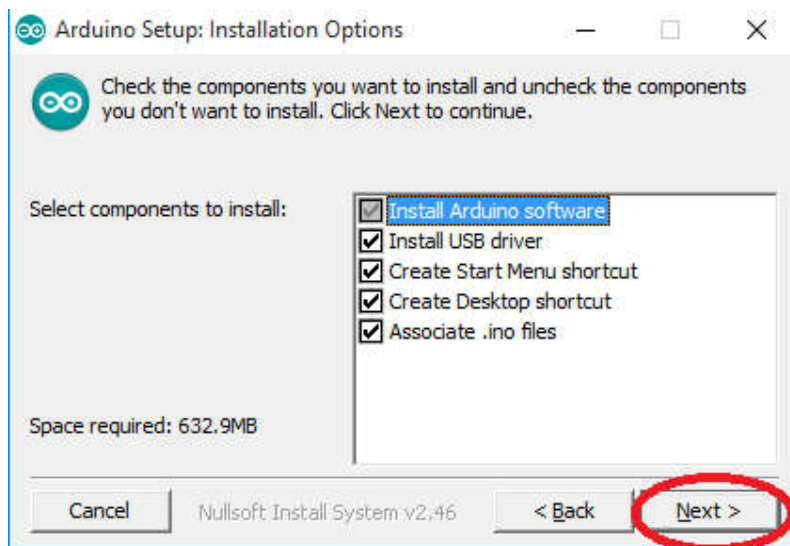


Figura 2.5 - Instalação da IDE

Assim, será aberta uma janela em que se pode alterar o local de onde o software será instalado, caso o usuário queira alterar o diretório, basta clicar em “Browse” e escolher a pasta de destino, após isso, deve-se instalar, clicando em “Install”.

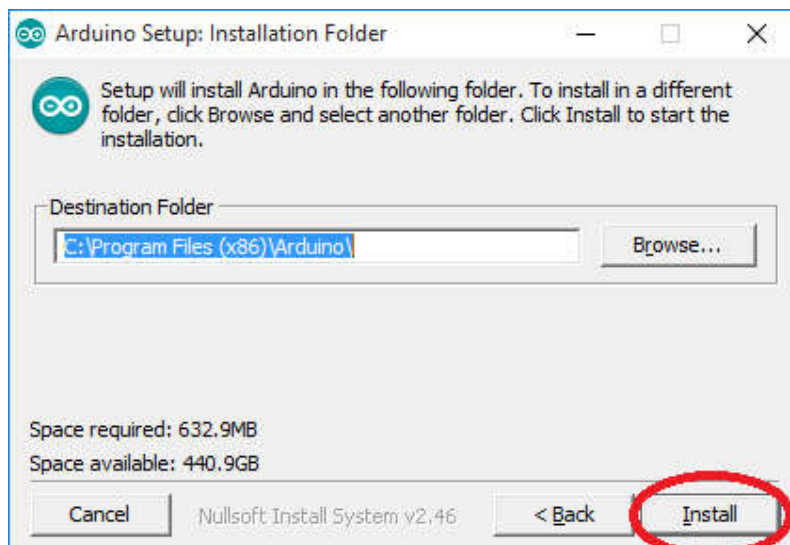


Figura 2.6 - Instalação da IDE

Após a realização desses passos, aparecerá uma mensagem informando que a IDE foi instalada com sucesso.

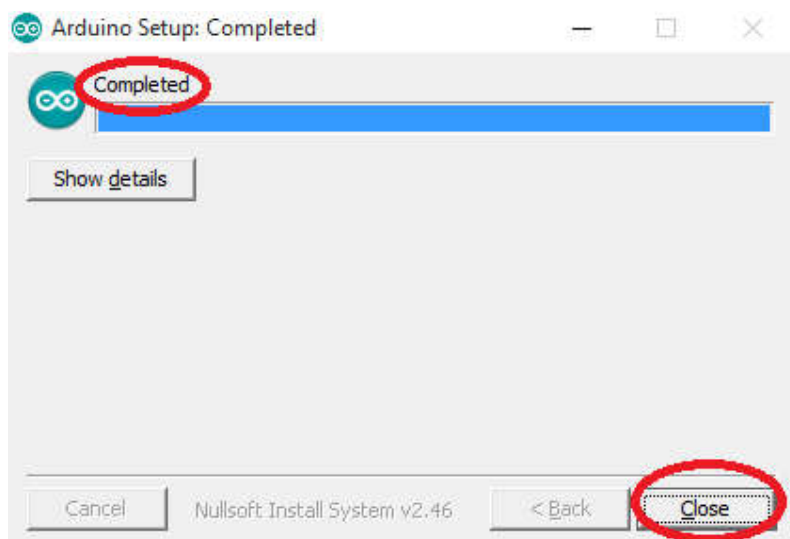


Figura 2.7 - Instalação da IDE

Feita a instalação, é necessário abrir a IDE e conectar o Arduino ao computador com auxílio do cabo USB, para que assim, possa configurar o tipo da placa e a porta serial utilizada.

Para isso, deve-se clicar na aba “Ferramentas”, e ir na opção “Placas”, para assim, escolher o tipo de Arduino, que nesse tutorial será o Uno. A configuração da placa e da porta serial é mostrada nas figuras 2.8 e 2.9.

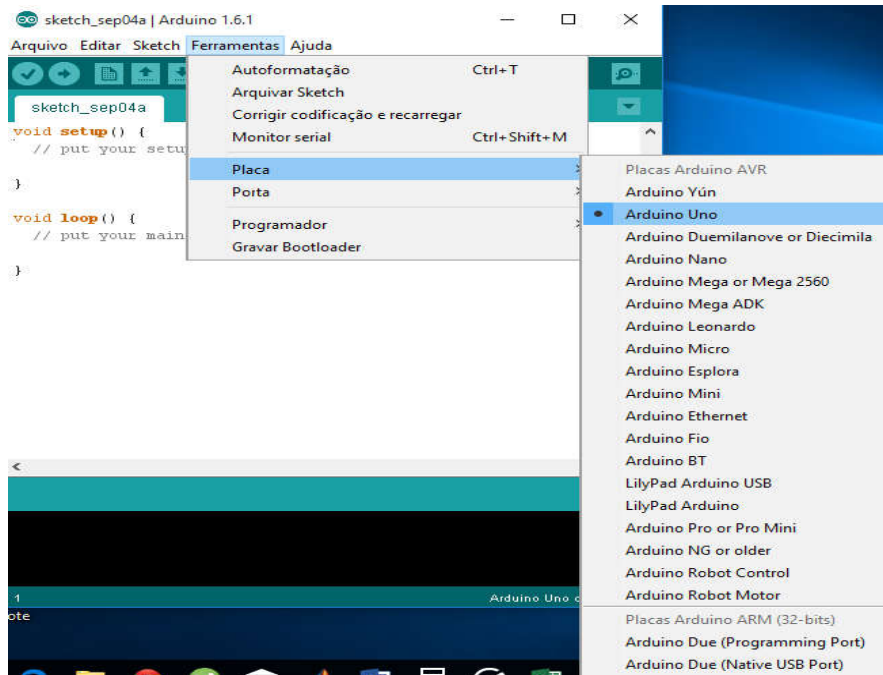


Figura 2.8 - Configuração da placa

Na mesma aba ferramenta deve-se clicar em “porta” para configurar a porta serial que está sendo usada.

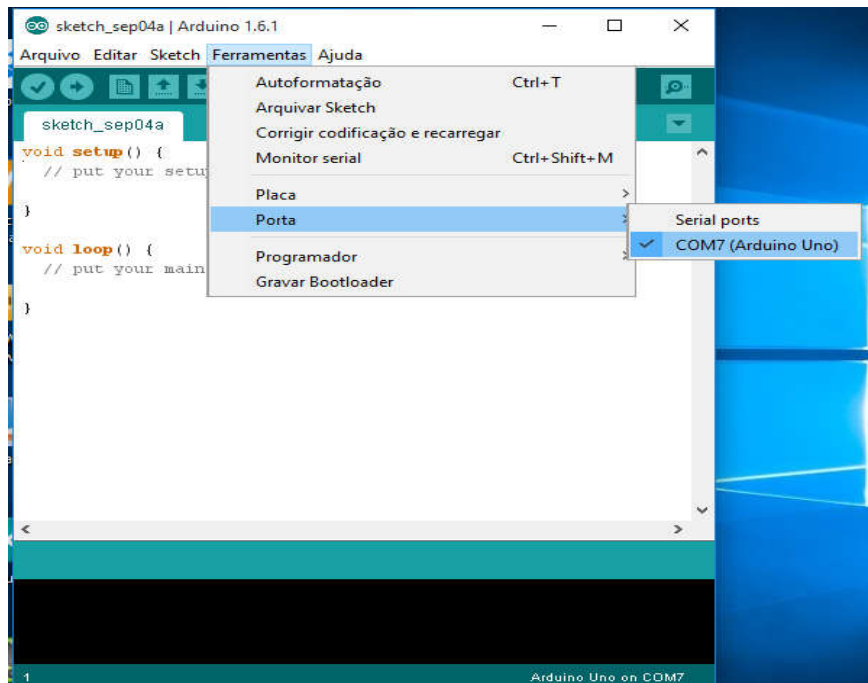


Figura 2.9 - Porta serial

### 2.3. Conceitos Básicos

Para escrita do código de programação existem funções e conectivos básicos, os quais estão listados abaixo:

#### 2.3.1. Funções Base

Tais funções já veem automaticamente escritas na IDE do Arduino na última versão (atualmente a versão é 1.6.1), sendo assim são funções de caráter obrigatório em todo código escrito para Arduino.

São as funções “void setup()” e “void loop()”, em que na primeira a função é executada uma vez, assim ela é usada para inicialização de variáveis, definição dos pinos, início da comunicação serial, etc. Já na segunda, existe um loop, ou seja, os comandos são repetidos, o que permite a leitura dos valores presentes nas portas, oriundos de sensores externos, por exemplo.

#### 2.3.2. Sintaxe

A linguagem usada nos códigos para Arduino é semelhante à usada em C e C++, exemplo:

- Ciclos

If...else, for, switch/case, while, do...while.

- Operadores

+, -, =, ==, !=, &&, ||, !, ++, --, +=.

- Variáveis

Boolean, char, int, long.

### 2.3.3. Gravando e compilando

Para gravar ou compilar, basta usar os botões que estão na IDE, os quais estão ilustrados na figura 2.10:

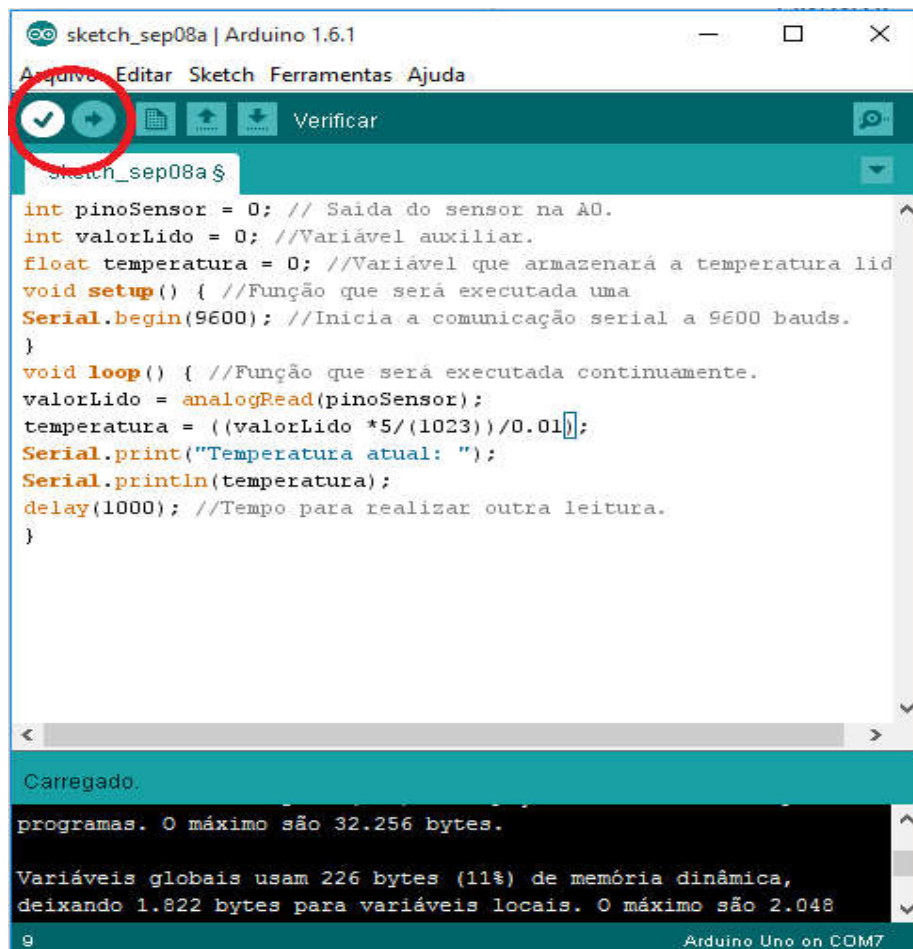


Figura 2.10 - Gravando e compilando

O primeiro botão compila o código, ou seja, verifica se existe algum erro de sintaxe no código, já o segundo grava o sketch no Arduino, faz o upload do código. Após se clicar em algum desses botões irá abrir uma janela para que salve o arquivo.

#### 2.3.4. Comunicação Serial

A comunicação serial é utilizada para comunicar o Arduino com o módulo sem fio a rádio e outros dispositivos, como Bluetooth e módulos xbee. A comunicação com o computador é possível através do conversor serial USB presente na placa.

Para se iniciar a comunicação serial deve-se habilitá-la através da função “begin”, dentro da função “void setup()”: “Serial.begin(9600);”, com esse comando, inicializa-se a comunicação serial a uma velocidade de baud igual a 9600. Essa velocidade é a taxa de comunicação em bits por segundo. A seguir exemplifica-se o código de habilitação da comunicação serial:

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

void loop() {
}
```

A IDE do Arduino possui uma ferramenta gráfica que facilita a comunicação entre o Arduino e o PC, o “monitor serial” nele são dispostos os dados lidos na porta de forma serial. Para acessá-lo, deve-se clicar na aba “Ferramentas”, e clicar em “Monitor Serial”. Outra forma é utilizar o atalho “Ctrl+Shift+M”. A figura 2.11 demonstra a função de monitor serial.

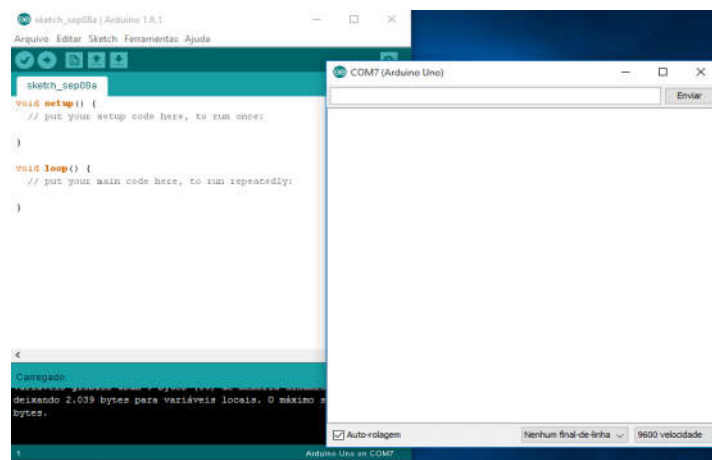


Figura 2.11 - Monitor serial

### 3. Sensores

Sensores são dispositivos capazes de transformar uma grandeza física em sinal elétrico, respondendo a um estímulo físico/químico de maneira mensurável analogicamente. Tais dispositivos, são amplamente utilizados em diversas áreas, bem como na medicina, indústria e robótica.

São classificados de acordo com a grandeza que detectam como luz, temperatura, pressão, fluxo, etc. Nesse tutorial serão apresentados os sensores LM35, que é um sensor de temperatura, e o HS0, o qual é um sensor de fluxo.

#### 3.1. Sensor de temperatura

O LM35 é um sensor de precisão fabricado pela National Semiconductor e por ser um circuito integrado, ele concentra todos os componentes necessários para a verificação da temperatura em uma única peça, fazendo com que o LM35 se torne um sensor extremamente simples de se trabalhar.

Ele funciona de forma que sua tensão de saída é proporcional a temperatura que se deseja medir. Como o sensor é alimentado com 5V, a variação de tensão vai de 0 a 5V.

Esse sensor não necessita de qualquer calibração externa para fornecer com exatidão os valores de temperatura com variações de  $\frac{1}{4}^{\circ}\text{C}$  ou até mesmo  $\frac{3}{4}^{\circ}\text{C}$  dentro da faixa de temperatura de  $-55^{\circ}\text{C}$  à  $150^{\circ}\text{C}$ . O sensor e sua ligação no Arduino é mostrado na figura 3.1.



Figura 3.1 - LM35



Colocando o terminal de saída do sensor na entrada analógica 0 do Arduino (A0), ele não lerá os valores em 0 a 5V, mas fará a leitura analógica da porta de valores inteiros de 0 a 1023. Sendo assim 1023 corresponde ao valor máximo de saída do sensor (5v) e 0 o menor valor (0V). O Arduino realiza essa conversão pelo fato de suas entradas analógicas possuírem resolução de 10 bits, ou seja,  $2^{10} = 1024$ .

Sendo assim, para saber o valor lido em A0 em volts, deve-se multiplicar o valor lido na porta por (5/1023), ou seja,  $Tensão\_em\_A0 = (Valor\ lido\ em\ A0) * (5/1023)$ .

Para obter a saída em graus Celsius deve-se levar em conta que 1°C corresponde a 0,01V, logo,  $Temperatura = [(Valor\ lido\ em\ A0) * (5/1023)] / 0,01$ .

A seguir tem-se o código comentado completo:

```
int pinoSensor = 0; // Saída do sensor na A0.
int valorLido = 0; //Variável auxiliar.
float temperatura = 0; //Variável que armazenará a temperatura lida
void setup() { //Função que será executada uma
  Serial.begin(9600); //Inicia a comunicação serial a 9600 bauds.
}
void loop() { //Função que será executada continuamente.
  valorLido = analogRead(pinoSensor); //Leitura analógica da porta A0
  temperatura = (valorLido * 0.00488);
  temperatura = temperatura * 100;
  Serial.print("Temperatura atual: ");
  Serial.println(temperatura); //Imprime na serial o valor da temperatura
  delay(1000); //Tempo para realizar outra leitura.
}
```

A montagem foi feita como ilustrado na figura 3.2 sendo que a primeira perna do LM35 (com o chanfro do sensor virado para a frente – cabo vermelho) é a alimentação, que é ligada na porta 5v do Arduino. A segunda é a saída do sensor (cabo azul) que é ligada na porta analógica 0 do Arduino (A0), e por fim, o último terminal é o terra, ligado na porta GND. A figura 3.2 foi elaborada usando o software Fritzing.

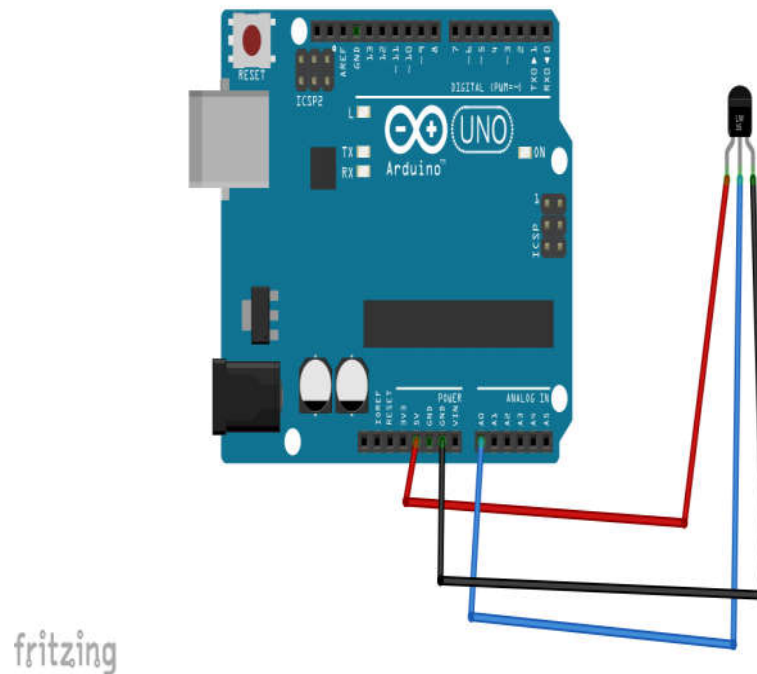


Figura 3.2 - Diagrama do circuito

Com esse código gravado na placa, pode-se ver os valores da temperatura através do monitor serial. Para verificar a variação de temperatura, pode-se aproximar a ponta do ferro de solda ao sensor, conforme é mostrado na figura 3.3:

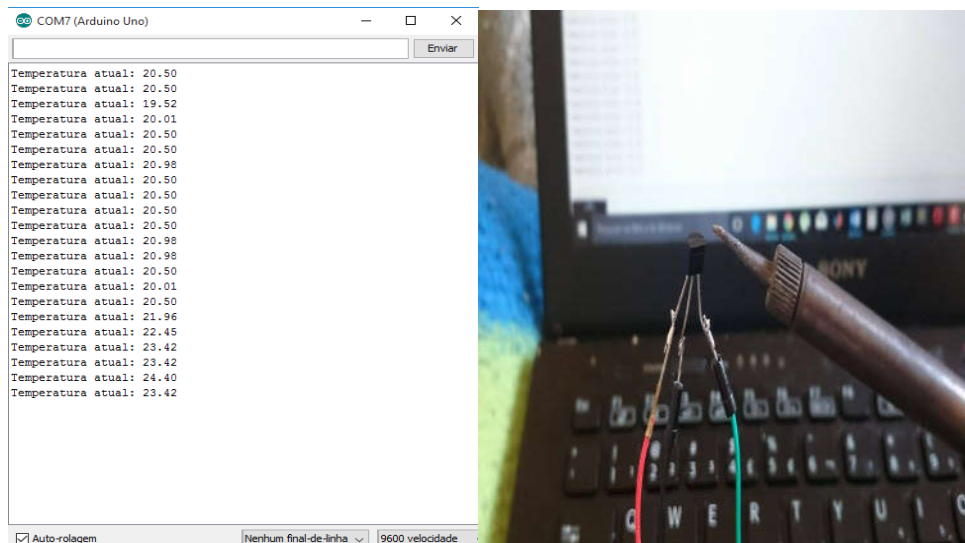


Figura 3.3 - LM35

### 3.2. Sensor de fluxo

O HS05, é um sensor de alta precisão capaz de medir a vazão de um fluido. Ele é composto por um encapsulamento de nylon, uma turbina contendo ímãs e um sensor de efeito Hall. Sendo assim, quando o fluido rotar a turbina, influenciará na frequência de pulsos que são colhidos na saída do sensor de efeito Hall. Através dessa contagem de pulsos é possível se medir a vazão.

Devido à ausência de datasheet do sensor, foi necessário realizar a calibração estática do módulo. Baseado na análise comparativa, utilizou-se como padrão um recipiente graduado, fazendo assim passar quantidades conhecidas de fluido pelo sensor e observou-se sua resposta.

Vale ressaltar que em todos os sensores desse tipo, existe em uma das suas extremidades uma “seta”, indicando o sentido em que o fluido deve passar. O sensor é mostrado na figura 3.4.



*Figura 3.4 - Sensor de fluxo HS05*

O sensor é alimentado com 5V, o terra deve ser ligado em GND e o cabo de dados deve ser ligado na entrada digital 2, visto que no código será usada a função interrupção. Para isso, foi realizado o código de forma que se contasse o número de pulsos realizado pelo sensor, usando a função interrupção do Arduino. A conexão do sensor com o Arduino é mostrada na figura 3.5

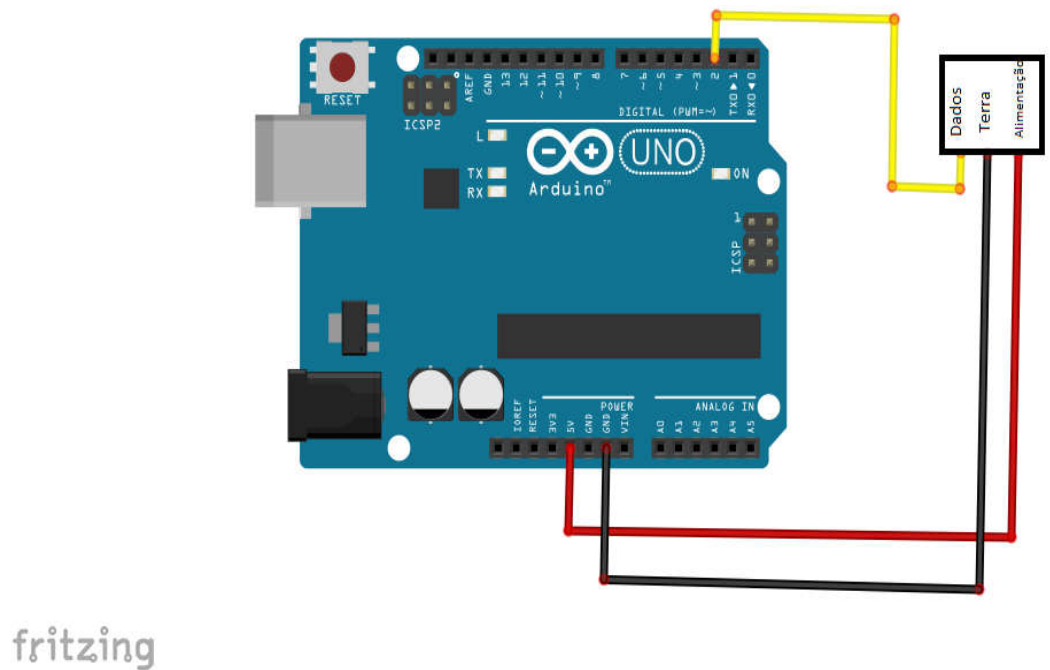


Figura 3.5 - Diagrama do circuito

A seguir, tem-se o código utilizado na calibração do HS05.

```

int contaPulso; //Variável para a quantidade de pulsos
int i=0; //Variável para contagem
int soma=0; //Variável para armazenar o total de pulsos
void setup()
{
  Serial.begin(9600); //Inicia a serial com um baud rate de 9600
  pinMode(2, INPUT);
  attachInterrupt(0, incpulso, RISING); //Configura o pino 2(Interrupção 0) para
  trabalhar como interrupção
  Serial.println("\n\nInicio\n\n"); //Imprime Inicio na serial
}

void loop ()
{
  contaPulso = 0; //Zera a variável para contar os giros por segundos

```

```

sei(); //Habilita interrupção
delay (1000); //Aguarda 1 segundo
cli(); //Desabilita interrupção
i++; // Incrementa i
soma=soma+contaPulso; // Soma dos pulsos mensurados
Serial.println(soma); // Imprime na serial a soma de pulsos
Serial.print(contaPulso); //Imprime na serial o valor de pulsos
Serial.print(" pulsos "); //Imprime "pulsos"
Serial.print(i); //Imprime a contagem i (segundos)
Serial.println("s"); //Imprime s indicando que está em segundos
}
void incpulso ()
{
    contaPulso++; //Incrementa a variável de contagem dos pulsos
}

```

Dessa forma, após observar a resposta para diferentes quantidades de fluido, passando pelo sensor 50, 100, 150, 200, 250ml de água, pode-se plotar um gráfico e por regressão linear, obter a curva de calibração, o qual é mostrado na figura 3.6.

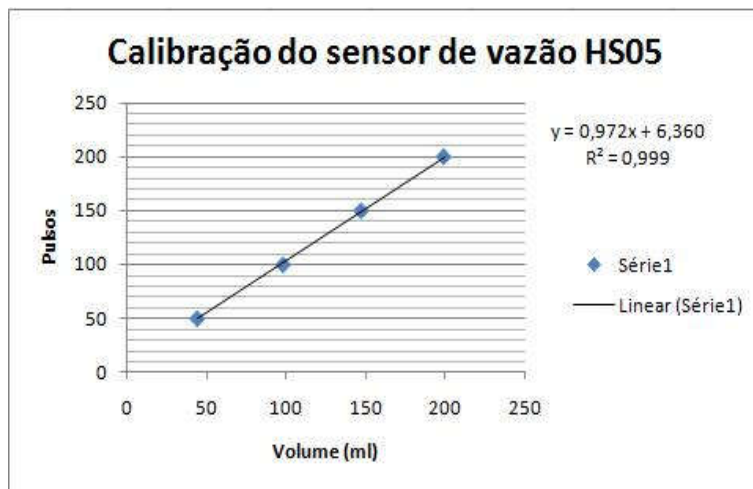


Figura 3.6 - Calibração do sensor

Feito isso, usando a equação da reta encontrada na regressão linear, é possível medir a vazão com o seguinte sketch:

```

short int Conta_Pulso;
short int Tanque_Vazio = 0;
void setup()

```

```

{
  Serial.begin(9600); //Inicia a serial com um baud rate de 9600
  pinMode(2, INPUT);
  pinMode(8, OUTPUT);
  attachInterrupt(0, incpulso, RISING); //Configura o pino 2 (Interrupção 0) para
  trabalhar como interrupção

  Serial.println("\n\nInicio da execucao do Software\n\n"); //Imprime Inicio na
  serial
}

void loop ()
{
  Serial.print(Tanque_Vazio); // Imprime na serial o volume
  Serial.print(" ml - ");
  Serial.print(Conta_Pulso); // Imprime o número de pulsos
  Serial.println(" pulsos");
  delay (1000);
}

void incpulso ()
{
  Conta_Pulso++;
  Tanque_Vazio = Conta_Pulso * 0.972 + 6.36;
}

```

A figura 3.7 mostra os testes para a calibração do sensor:

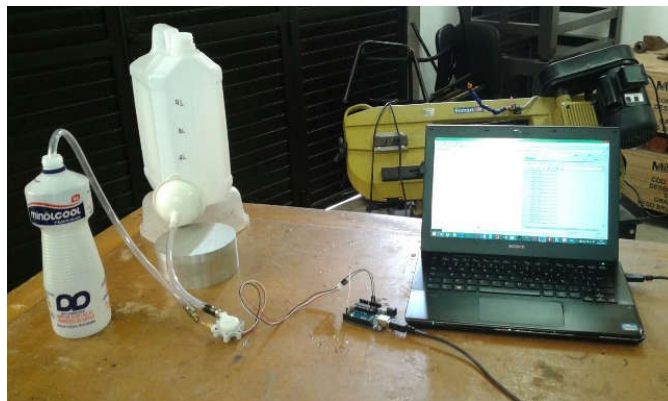


Figura 3.7 - Calibração do sensor de fluxo

Decidiu-se testar o sensor em uma aplicação, colocando-o para medir o volume de combustível consumido pelo motor a combustão OS 6.1. Porém o teste não obteve

sucesso, já que a quantidade de combustível exigida pelo motor era muito pouca, mesmo em sua aceleração máxima e não conseguia girar a turbina do sensor.

Como o sensor foi mal escolhido para essa aplicação, a solução encontrada foi o uso de um sensor de pressão, que ao ser colocado abaixo do tanque mediria a diferença de pressão, e assim é possível saber o nível do tanque.

#### 4. Comunicação remota

Os testes realizados nos capítulos anteriores foram feitos com o Arduino ligado diretamente na USB do computador, porém, em certas aplicações, é necessário utilizar uma comunicação sem fio, wireless, em que os sensores ficam distanciados do microcontrolador.

Existem diversas formas de se realizar essa comunicação: usando módulos xbee, os quais possuem uma área de alcance considerável e são de alto custo, dispositivos Bluetooth que apesar do baixo custo têm alcance muito reduzido e os módulos RF a rádio, em que o alcance varia com o preço.

Nesse tutorial foi utilizado um módulo a rádio, o 3DRobotics Radio Telemetry, cuja frequência de trabalho é de 915Mhz. Como seu alcance é de quase 2km, ele é ideal para telemetria de aeromodelos e drones.

Por ser de código aberto o módulo se torna uma boa opção para substituir os módulos xbee, os quais são de alto custo e possuem alcance menor, comparado a 3DR Radio Telemetry.

O módulo é composto por uma placa serial, que funciona como transmissor, e outra placa USB, cuja base fica ligada no PC hospedeiro e funciona como receptor. O módulo pode ser visto na figura 4.1.



*Figura 4.1 - Módulo sem fio*

Para usá-lo, deve-se primeiramente descobrir a velocidade de baud que o dispositivo trabalha. Para isso, foram realizados testes com usando as velocidades de comunicação com o computador: 300, 600, 1200, 2400, 4800, 9600, 14400, 19200, 28800, 38400, 57600 e 115200, desse modo descobriu que a velocidade que se deve usar

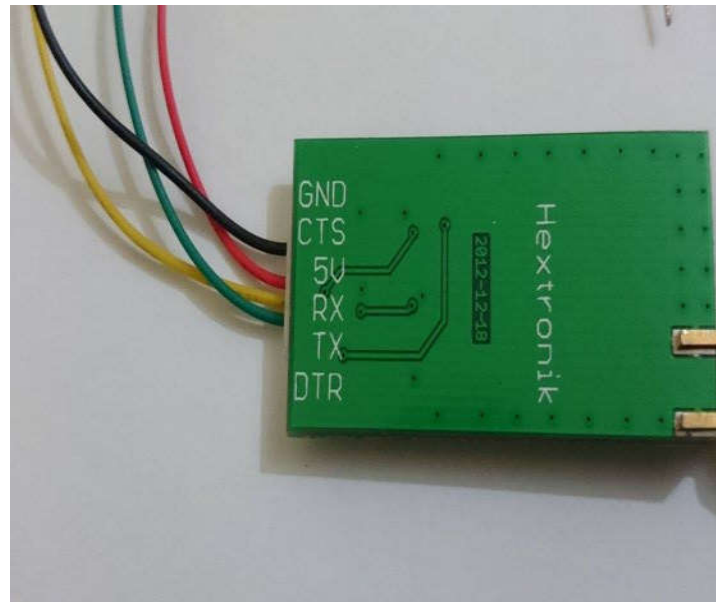


é a de 57600, pelo fato de que com as outras velocidades o transmissor fica mandando dados com interferência para a serial.

Nos capítulos anteriores foi usada uma velocidade de 9600. Para utilizar o transmissor à rádio deve-se habilitar a comunicação serial, usando a função “begin”, a uma velocidade de 57600.

A montagem deve ser feita de forma que o a alimentação do transmissor seja de 5v, o terra deve ser ligado no GND do Arduino e os cabos TX e RX, devem ser ligados de forma invertida, ou seja, o TX do transmissor deve ser conectado no RX do Arduino, e vice-versa.

Como mostra a figura 4.2, o cabo preto representa o terra, o vermelho a alimentação, a qual é de 5V, o amarelo o RX e verde o TX.



*Figura 4.2 - Módulo sem fio*

Para utilizar esse componente, o Arduino precisará de uma alimentação adicional, já que o receptor ficará ligado na USB e o receptor no Arduino sem nenhuma fonte de alimentação. Desse modo, pode-se usar uma simples bateria cuja tensão seja maior que 6v e menor que 20, ou uma célula fotovoltaica, a qual será melhor detalhada nos próximos capítulos. Para alimentar o Arduino, deve-se ligar a bateria sendo um de seus fios na entrada Vin do Arduino, e o outro em GND. A ligação é demonstrada na figura 4.3:

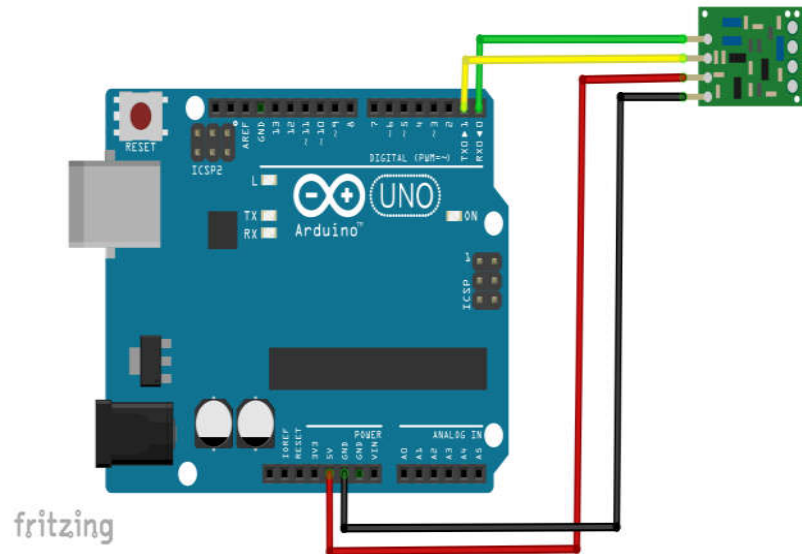


Figura 4.3 - Diagrama do circuito

É importante ressaltar que todo código deve ser gravado com o Arduino conectado ao computador com o cabo USB, pois diferentemente de alguns módulos Xbee, com o 3DR telemetry não é possível fazer o upload de um sketch de forma sem fio.

Por exemplo, usando a IDE do Arduino, deve-se gravar o código, usando o cabo USB, mudar a velocidade na programação, retirar o cabo e conectar as duas placas da 3DR telemetry, fazendo isso o dispositivo estará comunicando sem fio.

Desse modo, pode-se conectar os dois sensores e fazer com que apareça os dados colhidos por ambos.

## 5. Sistema de supervisão

Existem diversos meios de se projetar um sistema de supervisão como por meio de softwares como Matlab e Labview, porém devido o transmissor sem fio proposto nesse tutorial, fica impossível usar esses programas, uma vez que o transmissor usa uma velocidade de baud diferente do cabo USB e não é possível fazer o upload do código usando diretamente o transmissor.

No caso do Matlab e Labview usando o toolkit LIFA (Labview interface for Arduino) é necessário o upload de um sketch para que o Arduino comunique com o software. Outro toolkit usado no Labview é o LINX, que não é necessário gravar manualmente um código, porém, após seu firmware instalado não se pode alterar a velocidade de baud, além disso, quando se troca o cabo USB pelo transmissor sem fio a porta serial é mudada o que causa uma incompatibilidade.

Desse modo, será abordado no tutorial alternativas em que se pode coletar os dados de forma sem fio, utilizando softwares que possibilitam alterar a porta serial e a velocidade de baud. Sugeriu-se aqui como opções de software, o Meguno Link pro, o Makerplot e o Microsoft Office Excel.

### 5.1. Microsoft Office Excel

Utilizando o Excel é possível fazer a aquisição dos dados presentes na porta serial. Para isso é necessário utilizar a versão do office 2003 ou 2007 e instalar o programa PLX-DAQ.

Deve-se efetuar o download no site da empresa Parallax, no site <https://www.parallax.com/downloads/plx-daq>. Para isso, basta clicar em “PLX-DAQ.zip” para efetuar o download conforme ilustrado na figura 5.1.

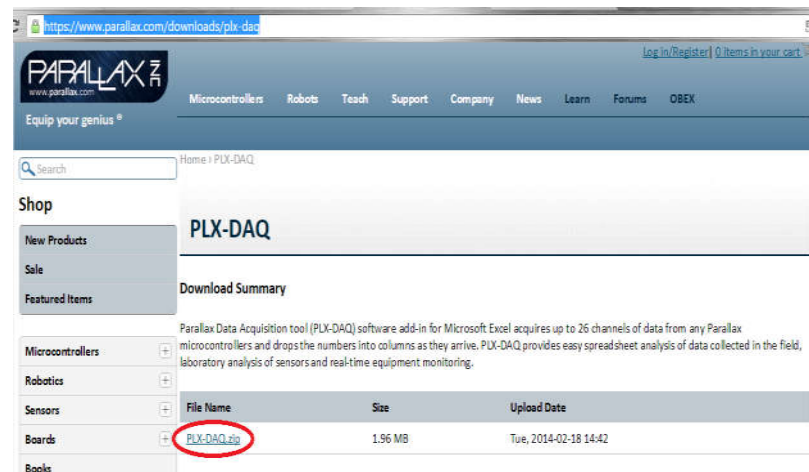


Figura 5.1 – Download do PLX-DAQ

Feito isso, deve-se abrir o arquivo .zip e clicar no arquivo “plx\_daq\_install.exe” para se iniciar a instalação.

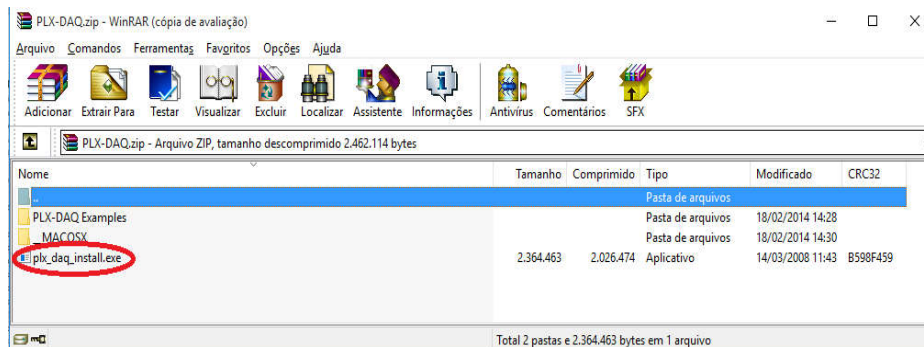


Figura 5.2 – Instalação PLX-DAQ

Assim, o arquivo será descompactado e abrirá a janela ilustrada na figura 5.3:

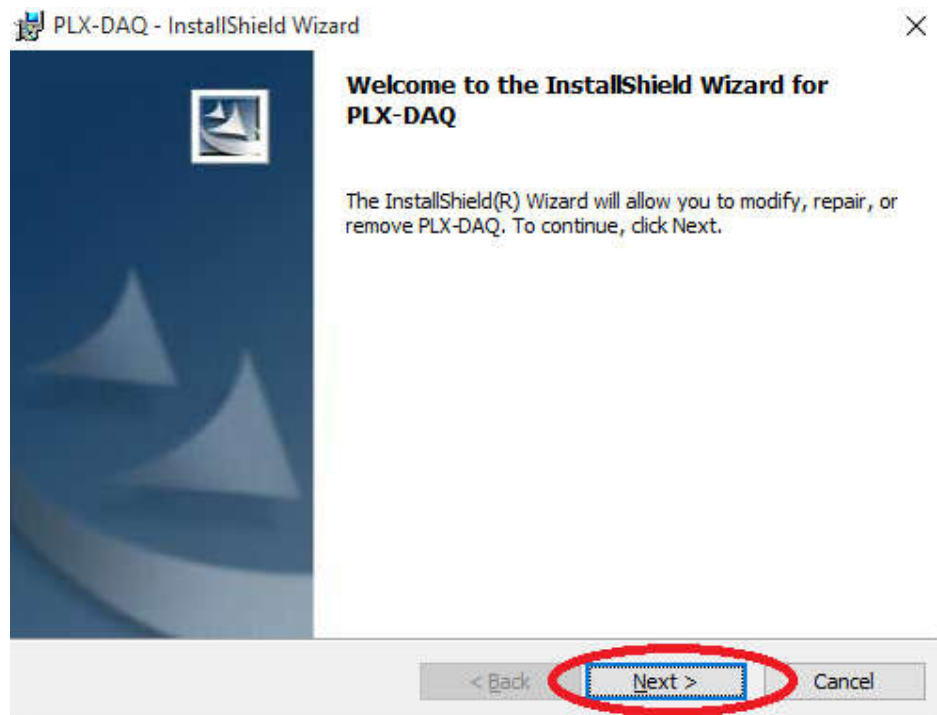
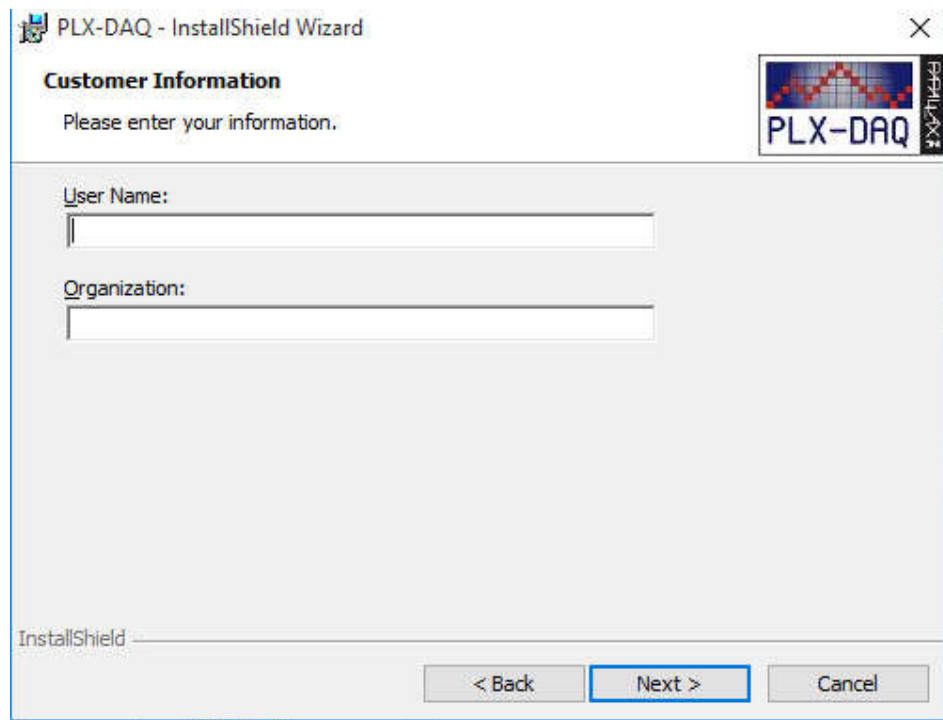


Figura 5.3 – Instalação PLX-DAQ

Deve-se clicar em “next” e assim, irá se abrir uma janela para se colocar o nome do usuário e a organização conforme figura 5.4.



The image shows a Windows-style dialog box titled "PLX-DAQ - InstallShield Wizard". The window has a standard title bar with a close button (X) in the top right corner. Below the title bar, the text "Customer Information" is displayed in bold, followed by the instruction "Please enter your information." In the top right corner of the window, there is a logo for "PLX-DAQ" which features a red and blue waveform graphic. The main area of the window contains two text input fields. The first field is labeled "User Name:" and the second is labeled "Organization:". Both fields are currently empty. At the bottom of the window, there is a status bar that says "InstallShield". To the right of the status bar, there are three buttons: "< Back", "Next >", and "Cancel". The "Next >" button is highlighted with a blue border, indicating it is the active or recommended action.

Figura 5.4 – Instalação PLX-DAQ

Assim, prosseguir com a instalação recomendada clicando em “next” e escolher o tipo de instalação “Typical”. Figura 5.5.

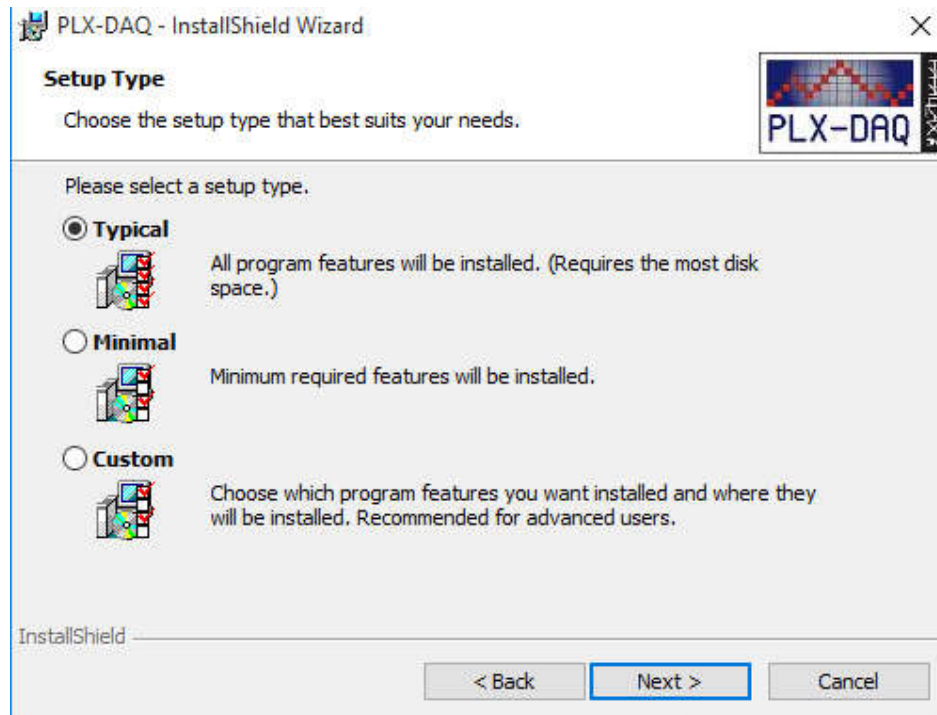
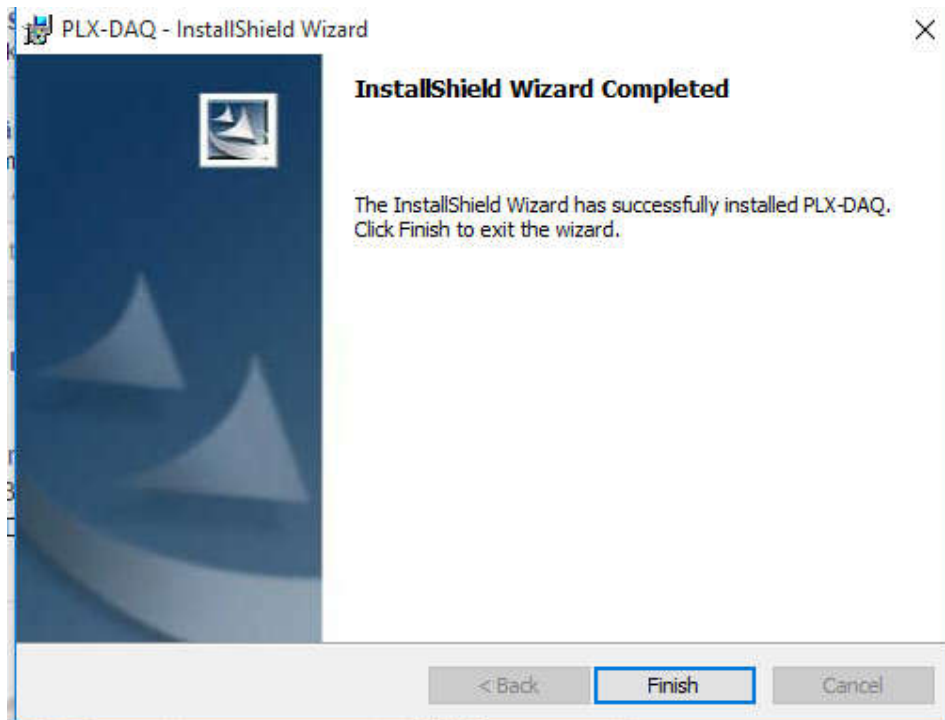


Figura 5.5 – Instalação PLX-DAQ

A seguir, finalizando a instalação do programa, clicar em “finish” de acordo com a figura 5.6.



*Figura 5.6 – Finalização da Instalação*

Feita a instalação deve-se ir até o diretório que contém o arquivo, geralmente fica no arquivo de programas, C:\Arquivos de Programas (x86)\ Parallax Inc, e assim clicar na planilha chamada PLX-DAQ. O arquivo é mostrado na figura 5.7:



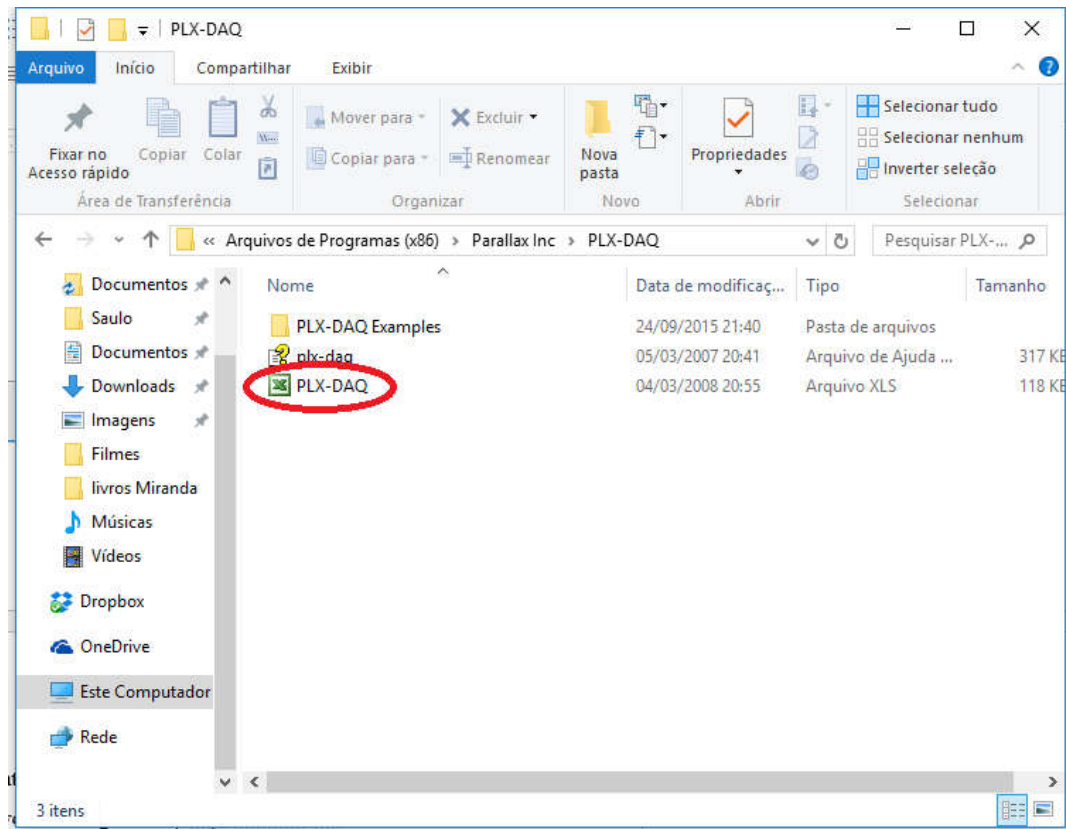


Figura 5.7 – Planilha base

Feito isso, irá aparecer uma mensagem para configurar a ferramenta macro, caso contrário deve-se configurar manualmente, assim como mostra a figura 5.8. A macro serve para que o programa trabalhe em tempo real.

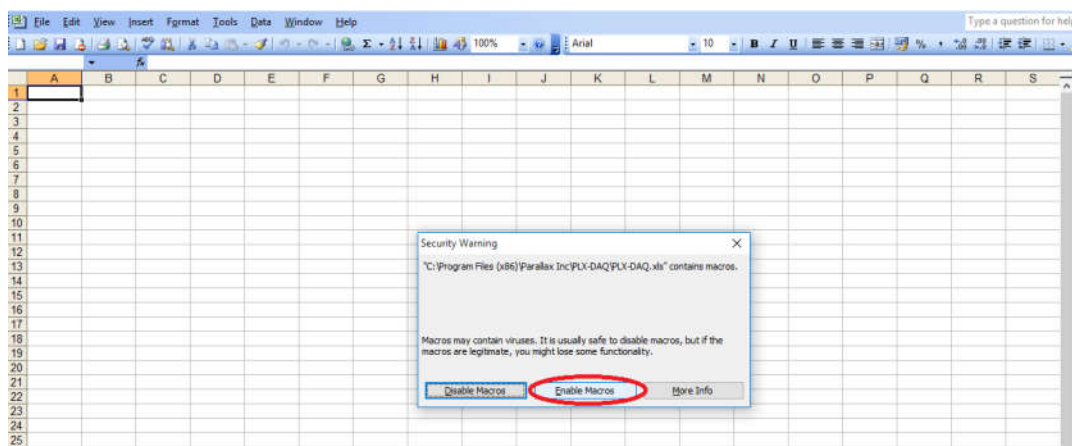


Figura 5.8 Configuração da Macro

Isso irá abrir a seguinte mensagem da figura 5.9, para prosseguir basta clicar em “ok”.

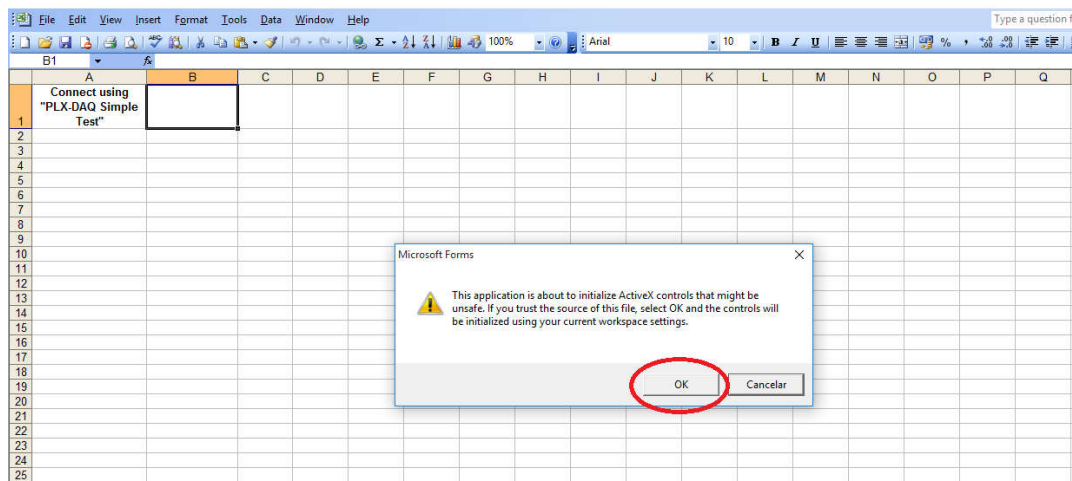


Figura 5.9 – Configuração da Macro

Assim, irá se abrir a interface do PLX-DAQ, em que se pode configurar a porta serial, o *baud rate* e o status da comunicação.

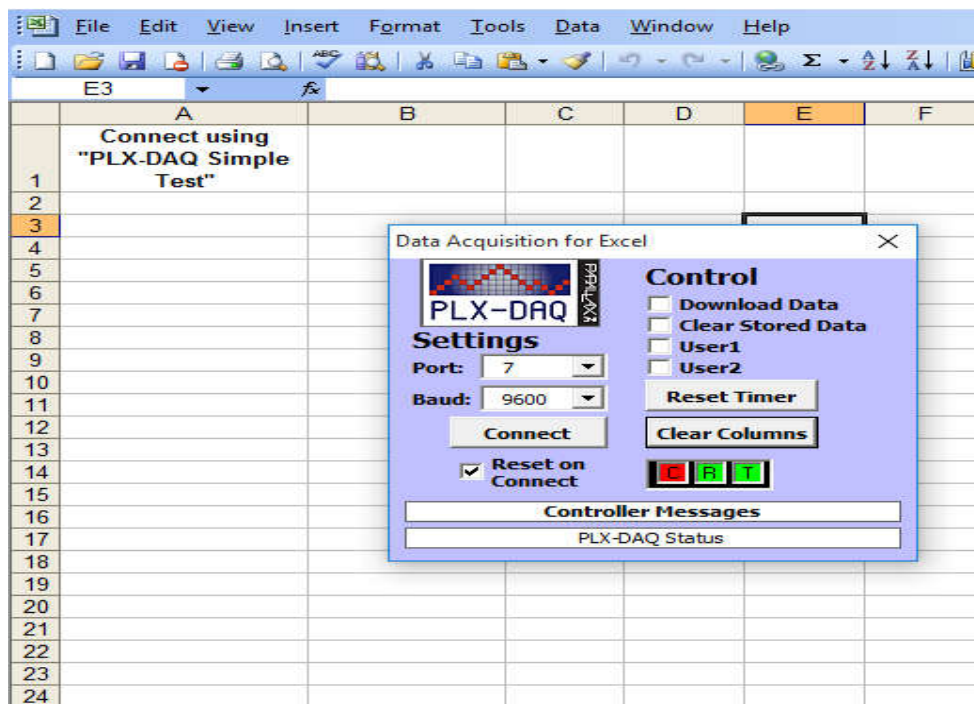


Figura 5.10 – Interface PLX-DAQ

Caso a macro não seja habilitada automaticamente, deve-se clicar na aba “Ferramentas/Tools”, depois em “Macro”, e em “Segurança/Security”.

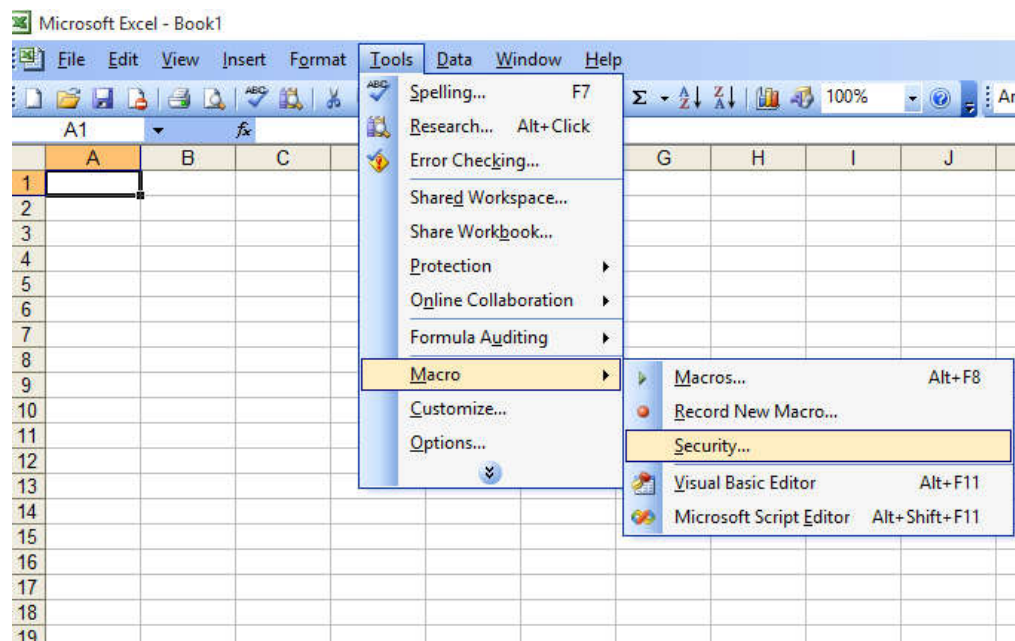


Figura 5.11 – Configuração da Macro

Após isso, marcar o nível de segurança como médio. Esse passo é demonstrado na Figura 5.12.

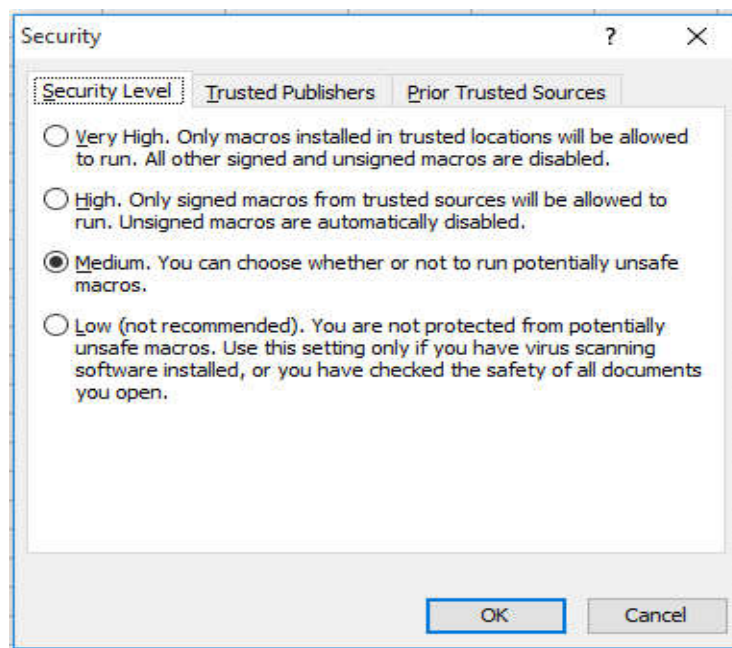


Figura 5.12 – Configuração da Macro

Assim, para que as variáveis e os dados emitidos pelos sensores sejam mostrados na planilha, deve-se realizar algumas alterações no código, utilizando a IDE do Arduino. Alterações as quais possibilitam a comunicação entre o Arduino e o Excel. A seguir é apresentado o código comentado referente à montagem da figura 5.13.

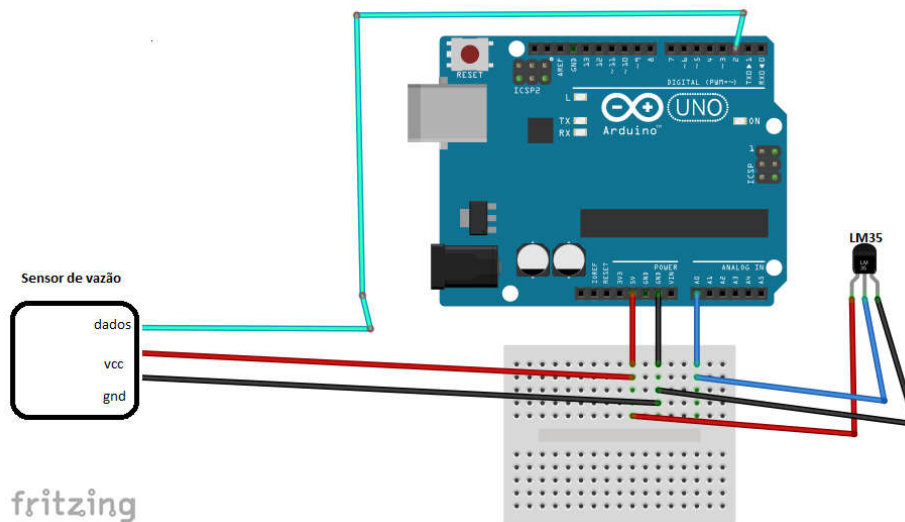


Figura 5.13 – Montagem do circuito

```
int pinoSensor = 0; //pino que está ligado o terminal central do LM35 (porta analógica 0)
int valorLido = 0; //valor lido na entrada analógica
float temperatura = 0; //valorLido convertido para temperatura
int linha = 0; //variavel que se refere as linhas do excel
short int Conta_Pulso;
short int Tanque_Vazio = 0; //variável referente ao volume
void setup() {
  Serial.begin(9600); //Inicializa comunicação Serial
  Serial.println("CLEARDATA"); // Reset da comunicação serial
  Serial.println("LABEL,Hora,Temperatura, Volume, linha"); // Nomeia as
colunas
  pinMode(2, INPUT);
  pinMode(8, OUTPUT);
  attachInterrupt(0, inculso, RISING); //Configura o pino 2(Interrupção 0) para
trabalhar como interrupção
```

```

}

void loop() {
  valorLido = analogRead(pinoSensor);
  temperatura = (valorLido * 0.00488); // 5V / 1023 = 0.00488 (precisão do A/D)
  temperatura = temperatura * 100; //Converte milivolts para graus celSius,
lembrando que a cada 10mV equivalem a 1 grau celSius

  linha++; // incrementa a linha do excel para que a leitura pule de linha em linha
  Serial.print("DATA,TIME,"); //inicia a impressão de dados, sempre iniciando
  Serial.print(temperatura);
  Serial.print(",");
  Serial.print(Tanque_Vazio);
  Serial.print(",");
  Serial.println(linha);

  if (linha > 100) //laço para limitar a quantidade de dados
  {
    linha = 0;
    Serial.println("ROW,SET,2"); // alimentação das linhas com os dados sempre
iniciando
  }

  delay(5000); //espera 5 segundos para fazer nova leitura
}

void incpulso ()
{
  Conta_Pulso++;
  Tanque_Vazio = Conta_Pulso * 0.972+6.36;
}

```

Como visto no código, pode-se limitar o número de medições, no código exemplo, foi usada 100 medições. Com isso, para gerar o gráfico em tempo real, basta seleccionar

as variáveis que serão usadas para compor o gráfico e fazer o gráfico de linha por dispersão x y.

Desse modo, os dados são dispostos na planilha, como mostrado na figura 5.14.

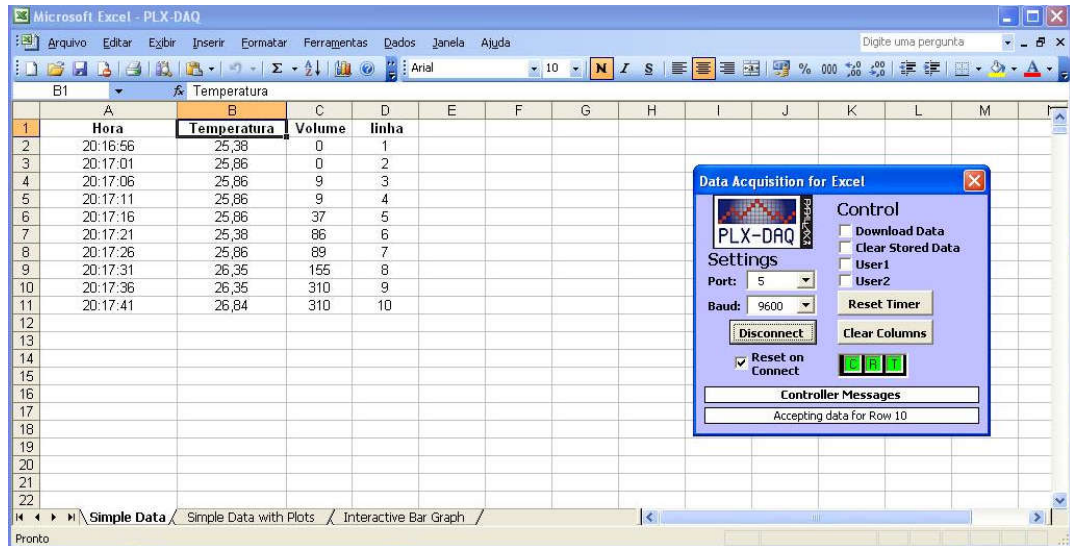


Figura 5.14 – Aquisição de dados com PLX-DAQ

Assim, pode-se inserir um gráfico de linha por regressão xy, selecionando as colunas em que se deseja plotar o gráfico, conforme mostra a figura 5.15.

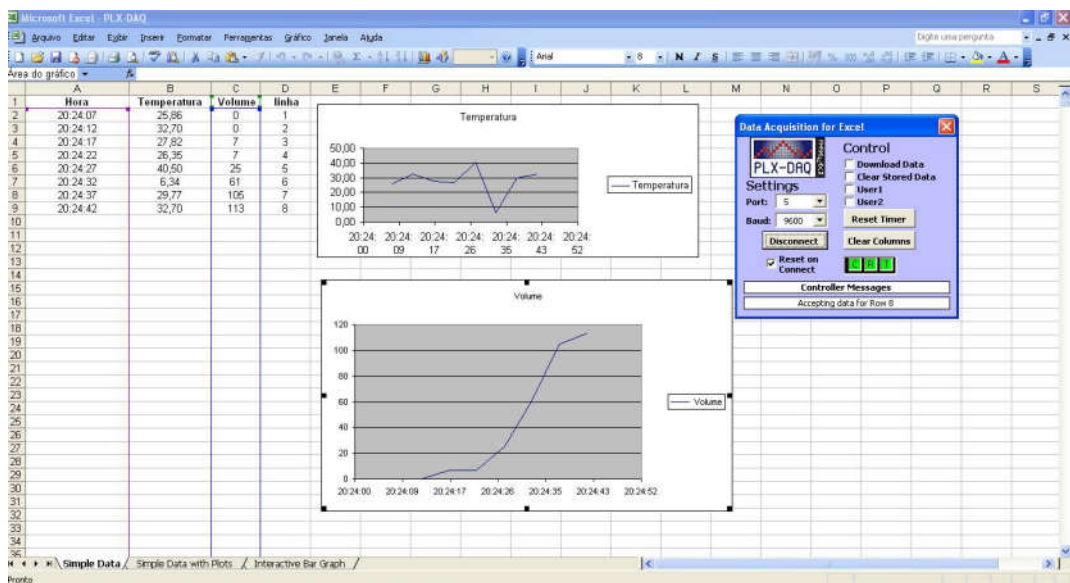


Figura 5.15 – Gráfico com PLX-DAQ



Os exemplos acima foram feitos com o Arduino conectado diretamente ao PC com o auxílio do cabo USB, sendo que para utilizar a comunicação sem fio, deve-se conectar o transmissor e receptor sem fio, e mudar a velocidade de baud e a porta serial usada pelo transmissor, visto que o código já está gravado na placa. A configuração é mostrada na figura 5.16.

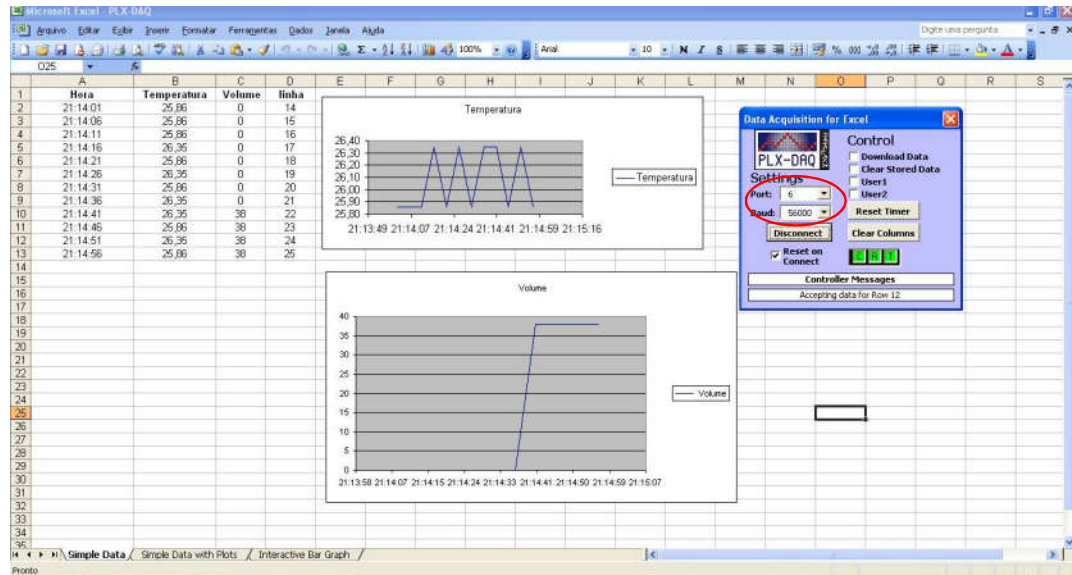


Figura 5.16 – Comunicação sem fio

## 5.2 Makerplot

Outra opção é o software Makerplot, em que se é capaz fazer a leitura da porta serial e realizar uma interface de fácil compreensão para o operador. Pode-se baixar uma versão demo no site da Makerplot: <http://www.makerplot.com/download.htm>.

Figura 5.17.



Figura 5.17 – Download MakerPlot

Feito o download da versão demo, clicando no arquivo, inicia-se a instalação do software. Abrirá uma janela com um aviso de segurança, para prosseguir, clica-se em “Executar”. Esse passo é mostrado na figura 5.18.

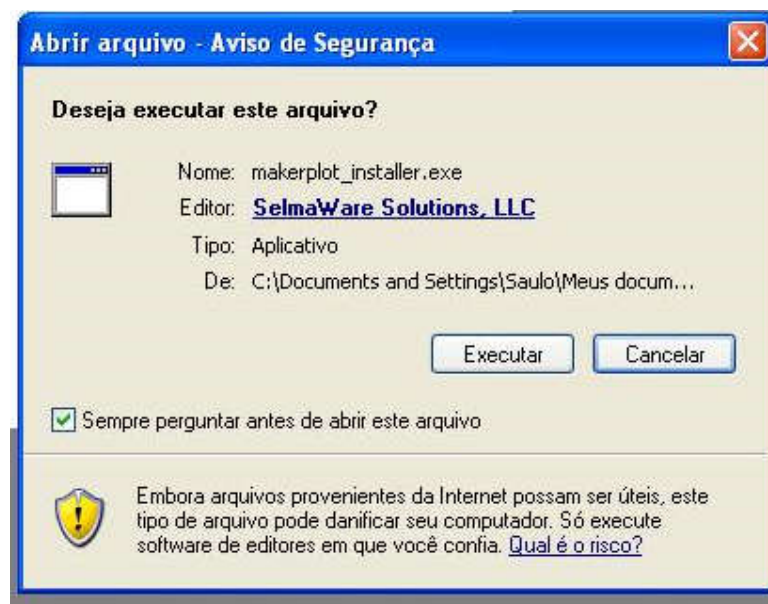


Figura 5.18 – Instalação MakerPlot

Após esse passo, irá se abrir a outra janela e para prosseguir deve-se clicar no botão “Next”.





Figura 5.19 – Instalação MakerPlot

Para continuar a instalação, basta clicar em “Next”, e aceitar os termos da licença. Figura 5.20.



Figura 5.20 - Instalação MakerPlot

Nesse passo, deve-se registrar o nome do usuário e a organização, e prosseguir.

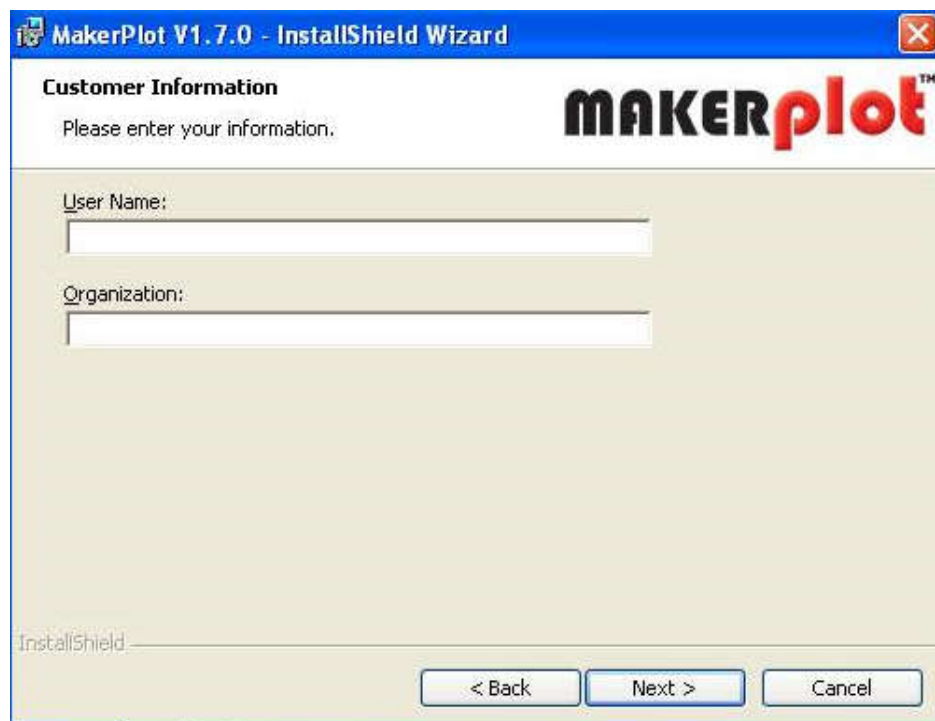


Figura 5.21 - Instalação MakerPlot

A próxima janela, Figura 5.22, é para informar o local em que o software será instalado.



Figura 5.22 - Instalação MakerPlot

Por fim, aparecerá uma revisão das configurações do software a ser instalado; clica-se em “Install” para começar o processo. Vide Figura 5.23.



Figura 5.23 - Instalação MakerPlot

Feito todos os passos, irá aparecer a janela informando que a instalação foi concluída com êxito conforme Figura 5.24.

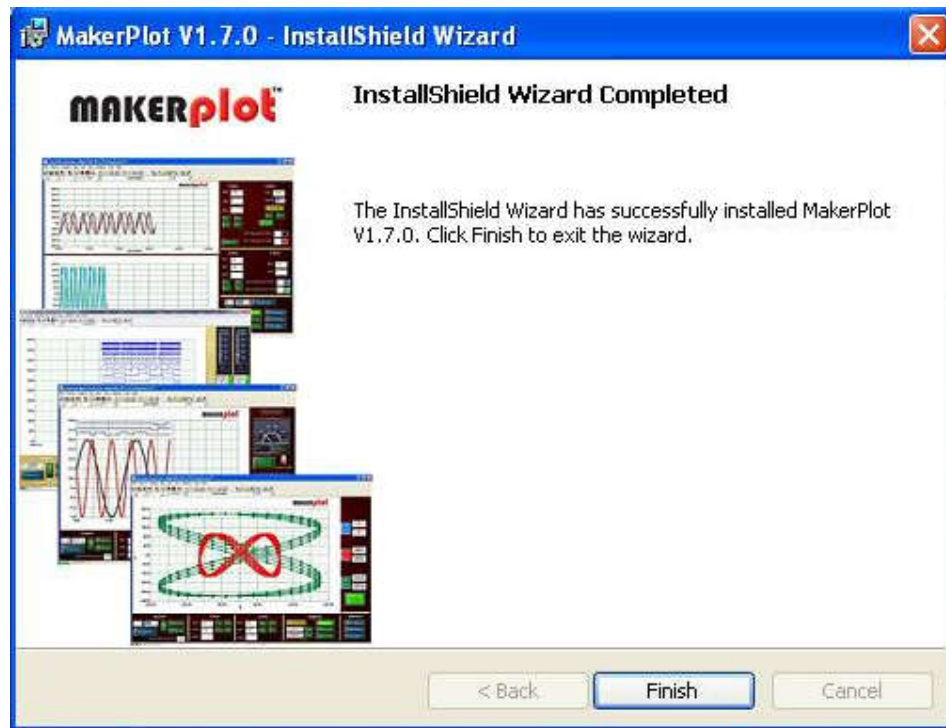


Figura 5.24 - Instalação MakerPlot

Feita a instalação, pode-se abrir o programa e perceber que existem exemplos de aplicações na tela inicial, podendo-se escolher o número de canais analógicos ou digitais que serão utilizados, o número de gráficos, valores máximos e mínimos, interfaces que serão mostradas ao operador, entre outros.

Utilizou-se a configuração com quatro canais e quatro indicadores de medição nesse tutorial. Vide Figura 5.25.

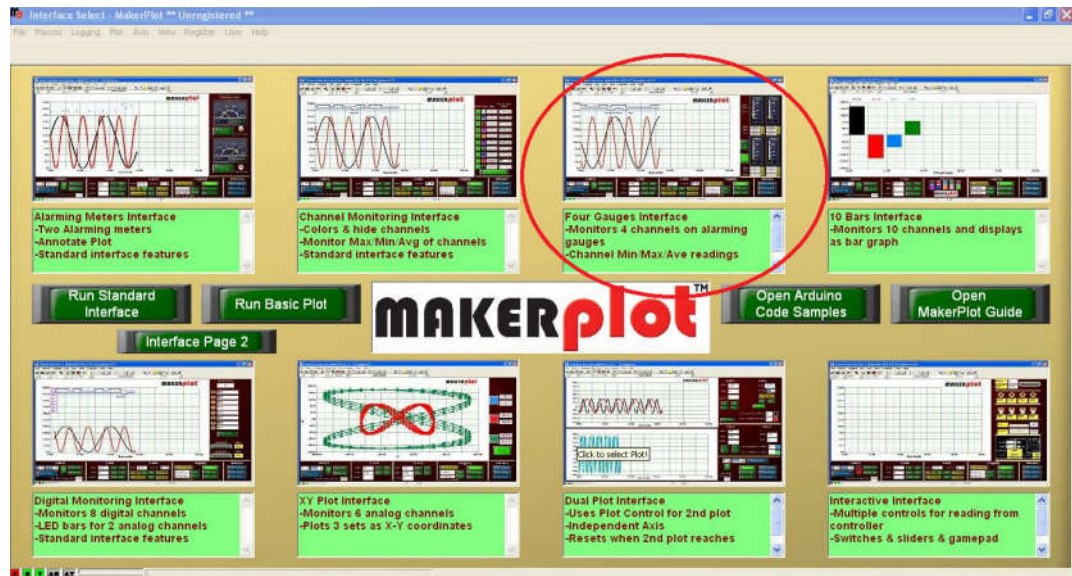


Figura 5.25 – Configuração MakerPlot

Desse modo, para se começar a usar o MakerPlot, basta alterar o código na IDE do Arduino, separando as variáveis que se deseja mostrar no gráfico por uma vírgula, ou seja, colocar no código “Serial.print(",");”.

Assim, basta configurar no MakerPlot a porta serial e a velocidade de baud e clicar na chave para conectar de acordo com a Figura 5.26.

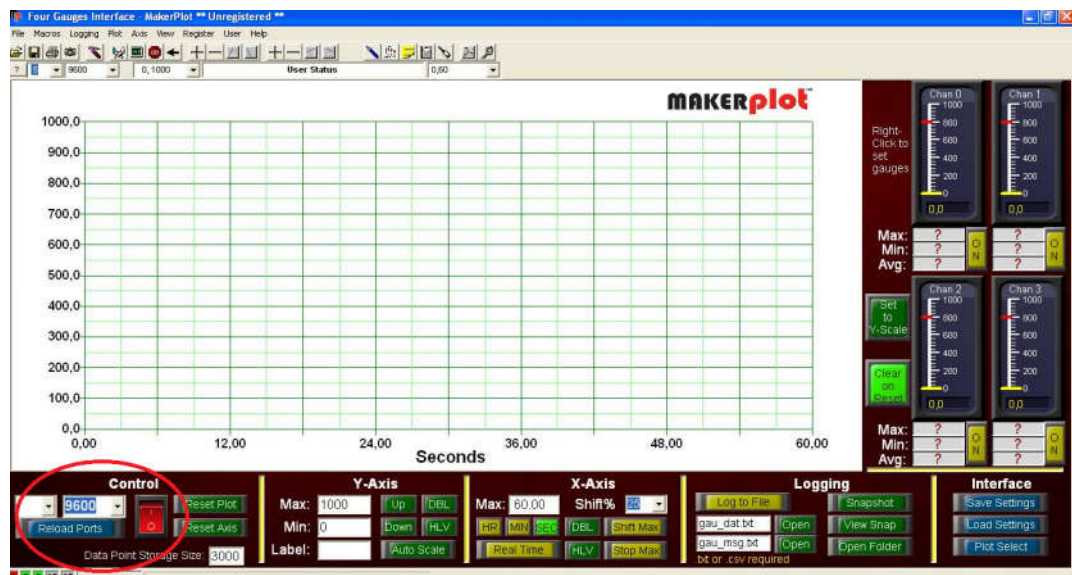


Figura 5.26 – Configuração MakerPlot



Feito isso, a comunicação irá se iniciar, com os dois canais, o volume e a temperatura. Outra função interessante é data logging, que quando acionada, o MakerPlot começa a gravar os dados lidos em um arquivo .txt.

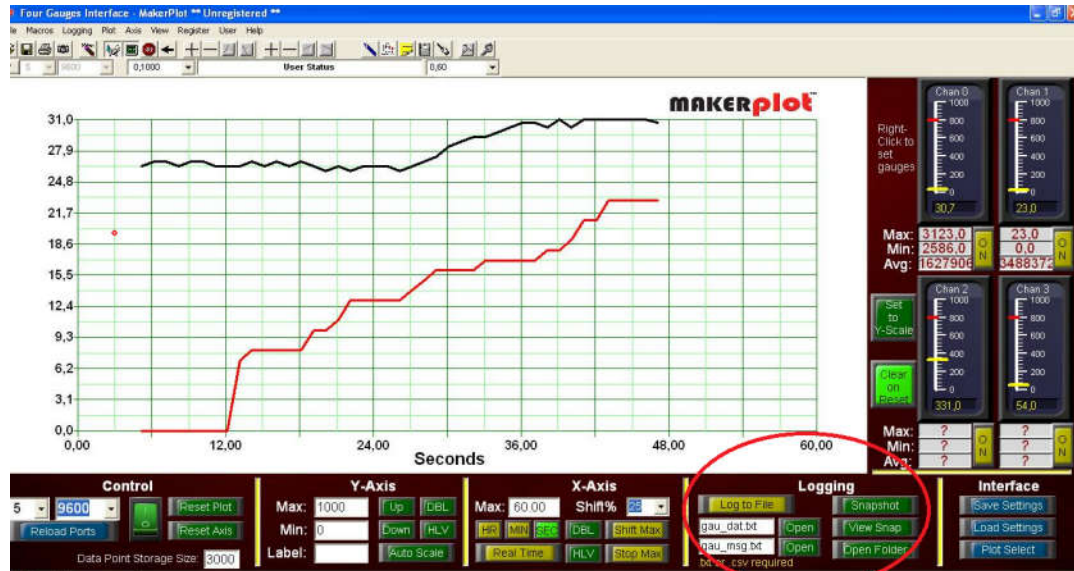


Figura 5.27 – Gráfico e data logging no MakerPlot

## 6. Alimentação

Como mencionado nos capítulos anteriores para se alimentar um Arduino, pode-se usar qualquer bateria ou transformador, sendo que a mesma possua tensão superior a 6V (valor limite). Desse modo, decidiu-se alimentar o sistema abordado nesse tutorial por placas fotovoltaicas.

Uma placa fotovoltaica é um dispositivo capaz de transformar luz em energia elétrica, por intermédio do efeito fotoelétrico. Tal fenômeno ocorre pelo fato da placa ser formada de um material semicondutor, ou seja, um material com características que não são nem condutoras e nem isolantes.

Um exemplo de semicondutor é o silício, material que é dopado com o elemento fósforo, obtendo-se um material com elétrons livres (silício tipo N). Dopando-se o silício com Boro, obtém-se um material com características inversas, ou seja, com falta de elétrons (silício tipo P).

Assim, uma célula fotovoltaica é composta por uma camada fina de material tipo N, acoplada com uma camada espessa de material tipo P. Essa união forma um campo elétrico devido os elétrons livres da região N irem para os lugares vagos na região P. Ao se incidir luz sob a célula, os fótons chocam-se com os elétrons da estrutura, transformando-os em condutores, ou seja, criando um fluxo de elétrons, a corrente elétrica.

Vale ressaltar que a placa não armazena energia elétrica, ela mantém o fluxo de elétrons quando se incide luz solar sob a placa, o que inviabiliza a placa em algumas aplicações pois depende das condições climáticas.

Como anteriormente informado, será usado neste tutorial dois painéis de 4V, como deseja-se aumentar a tensão dos painéis, eles devem ser associados em série. Foram associados assim como mostra a figura 6.1 e 6.3 os diodos foram colocados pelo fato dos diodos não deixarem que a corrente passe por outro sentido, ou seja, funcionam como dispositivo de proteção, já que caso a corrente mude de sentido isso pode acarretar na queima total ou parcial da placa.



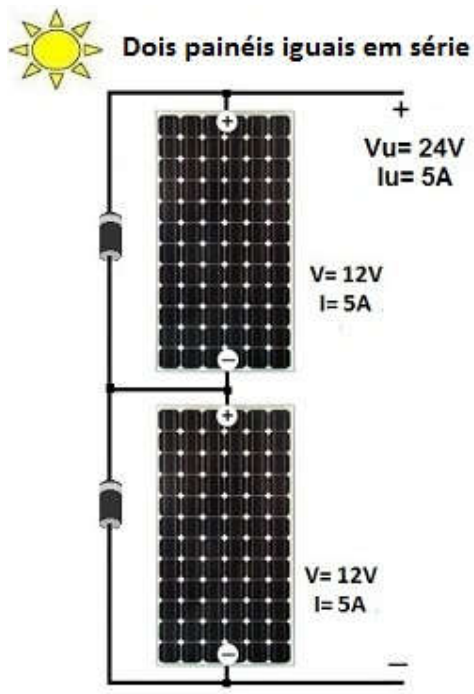


Figura 6.1 – Associação de placas em série

A ligação do painel com o Arduino é feita como se fosse qualquer bateria, o terminal positivo em  $V_{in}$  e o negativa no GND. As figuras 6.2 e 6.3 mostram a associação e a ligação no Arduino.

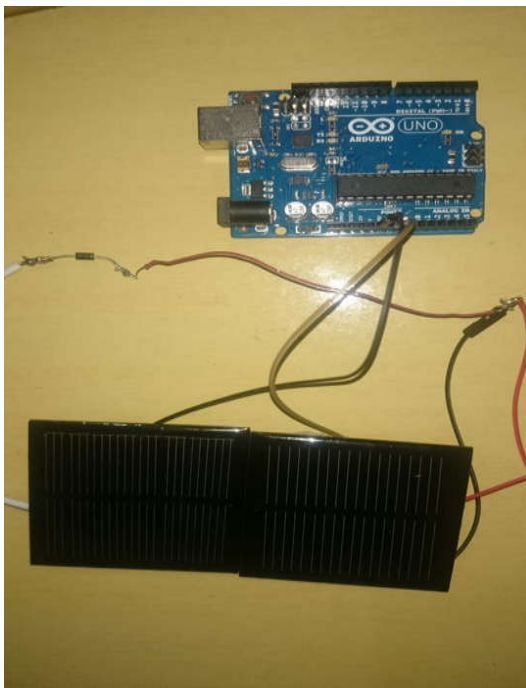


Figura 6.2 – Ligação no Arduino

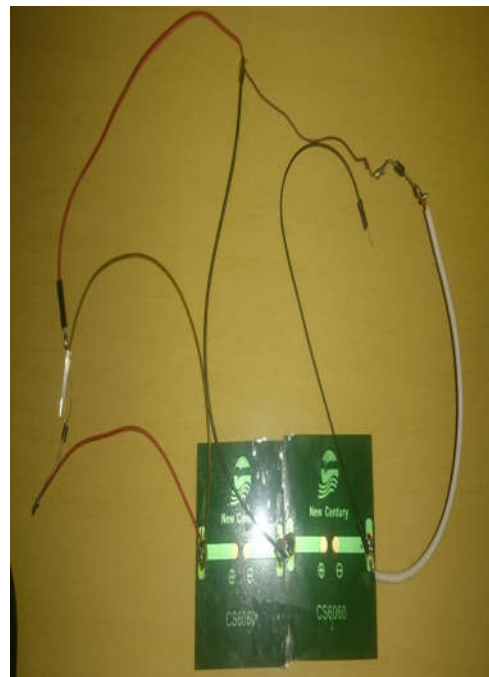


Figura 6.3 – Associação em série

## 7. Conclusão

Levando em consideração os resultados do projeto, pode-se dizer que foram alcançados com sucesso, porém deve-se ressaltar a escolha do sensor, o qual deve ser selecionado de forma que trabalhe de acordo com o tipo de aplicação que ele será usado, isso é de extrema importância.

Na medição do volume de combustível consumido pelo motor, utilizado no teste do protótipo aqui desenvolvido, a escolha do sensor foi equivocada, mas para outras aplicações o sensor de fluxo funcionaria perfeitamente, como por exemplo, medir o consumo de água de uma caixa d'água. Além disso, pode-se perceber a facilidade que o Arduino proporciona de se trabalhar com outros sensores e módulos, podendo-se adaptar o projeto para as mais diversas aplicações.

O uso de placas fotovoltaicas é de extrema importância, já que se trata de uma fonte de energia renovável, a qual não agride o meio ambiente, porém nesse tutorial as placas foram ligadas diretamente no Arduino, assim para acontecer o fluxo de corrente as placas devem estar constantemente sendo irradiadas pelo sol, visto que elas não armazenam energia. Uma solução para projetos que não necessariamente ficarão expostos ao sol é a criação de um circuito capaz de carregar uma bateria adicional que alimenta o Arduino.

## 8. Referências

MCROBERTS, Michael. **Arduino básico**, 1.ed. São Paulo: Novatec, 2011. 453 p.

SANTOS, Nuno Pessanha. **Arduino - introdução e recursos avançados**, Lisboa: Escola Naval Ramo de armas e eletrônica 2009. 69 p.

NASCIMENTO, Cassio Araújo. **Princípio de funcionamento da célula fotovoltaica**, 2004. 21 p. Monografia pós-graduação – Universidade Federal de Lavras.

Mpptsolar, **Ligação em série de mais painéis solares**. Disponível em:< <http://www.mpptsolar.com/pt/paineis-solares-em-serie.html>>. Acesso em 3 de setembro de 2015.