AULA 13 - IoTA

Programação física

Comunicação

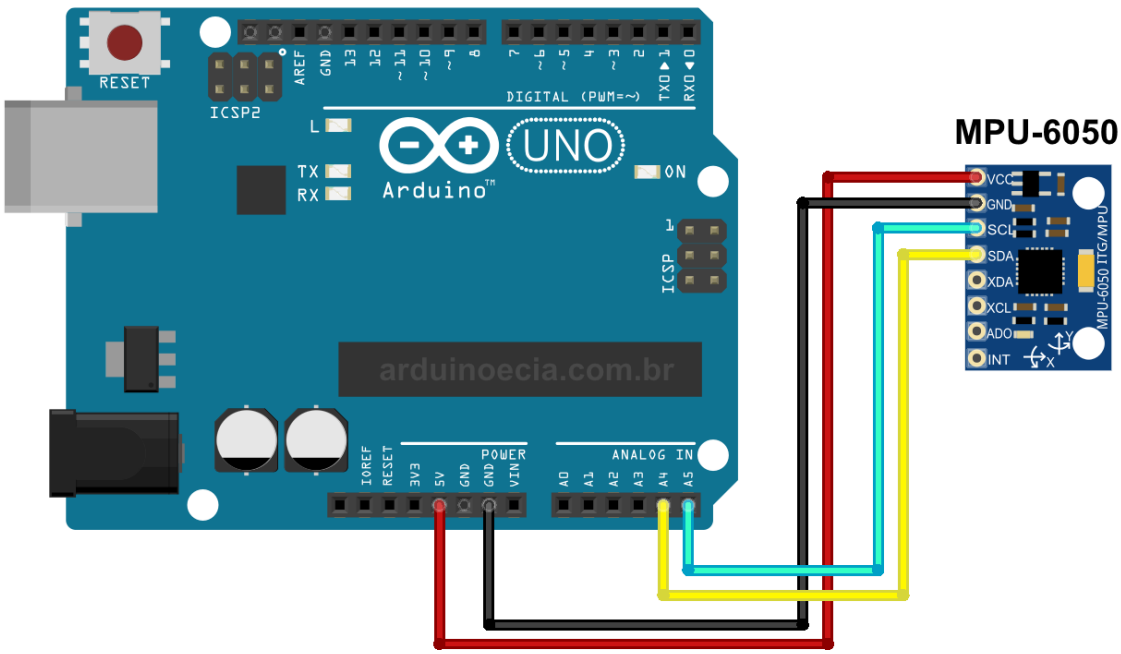
Nas semanas anteriores, você aprendeu a usar sensores e atuadores de diversos tipos com a sua placa Arduino. Nesta semana, vamos começar a falar sobre comunicação.
Vimos que há portas digitais e analógicas na Arduino, que são escolhidas de acordo com o resultado que queremos: se só precisamos de dois estados, utilizamos portas digitais e sensores digitais; se precisamos de maior precisão, utilizamos portas analógicas e sensores analógicos.
Mas, e se precisarmos nos comunicar com componentes mais complexos?
Neste caso, utilizamos protocolos de comunicação. Com eles, conseguimos sinais mais complexos a partir das portas disponíveis na Arduino. Esses sinais são utilizados por sensores específicos - como o módulo MPU6050, um acelerômetro e giroscópio, que está presente no nosso kit de eletrônica, e que utiliza o protocolo I2C.
No próximo vídeo mostraremos como realizar a comunicação da sua placa com o módulo acelerômetro e giroscópio.

Sensores e diferentes protocolos de comunicação

https://youtu.be/kzQqfaI8YuQ

Módulo MPU-6050: Circuito e código

No circuito do vídeo anterior utilizamos:
Uma placa Arduino
Um módulo MPU6050 (acelerômetro e giroscópio)
4 fios para conexão
Abaixo, apresentamos o circuito e o código utilizados.



Créditos da imagem: Arduino e Cia.
Fonte: http://www.arduinoecia.com.br/2015/04/acelerometro-giroscopio-mpu-6050.html

Conecte o módulo de acordo com a imagem e, em seguida, faça o upload do código para a sua placa.

Abra o monitor serial e faça movimentos com o módulo para observar o que ocorre com os valores das leituras.

```
/*
 Lendo o Módulo Acelerômetro e Giroscópio MPU-6050
*/
//Carrega a biblioteca Wire para comunicação com o módulo
#include<Wire.h>
//variável com o endereco I2C do MPU6050
const int MPU=0x68;
//Variaveis para armazenar valores dos sensores presentes no módulo
int AcX,AcY,AcZ,Tmp,GyX,GyY,GyZ;
// configuração inicial
void setup()
{
  // inicialização do monitor serial em 9600 baudrates
  Serial.begin(9600);
 // inicialização da comunicação com o módulo
  Wire.begin();
  Wire.beginTransmission(MPU);
  Wire.write(0x6B);
 // inicialização do módulo
  Wire.write(0);
  Wire.endTransmission(true);
}
// loop infinito, tudo que estiver dentro será repetido indefinidamente
void loop()
{
  // Configuração do módulo
  Wire.beginTransmission(MPU);
  Wire.write(0x3B);
  Wire.endTransmission(false);
 // Solicita os dados do sensor
  Wire.requestFrom(MPU,14,true);
 // Armazena o valor dos sensores nas variaveis correspondentes
  AcX=Wire.read()<<8|Wire.read(); //0x3B (ACCEL_XOUT_H) & 0x3C (ACCEL_XOUT_L)
  AcY=Wire.read()<<8|Wire.read(); //0x3D (ACCEL_YOUT_H) & 0x3E (ACCEL_YOUT_L)
  AcZ=Wire.read()<<8|Wire.read(); //0x3F (ACCEL_ZOUT_H) & 0x40 (ACCEL_ZOUT_L)
  Tmp=Wire.read()<<8|Wire.read(); //0x41 (TEMP_OUT_H) & 0x42 (TEMP_OUT_L)
  GyX=Wire.read()<<8|Wire.read(); //0x43 (GYRO_XOUT_H) & 0x44 (GYRO_XOUT_L)
  GyY=Wire.read()<<8|Wire.read(); //0x45 (GYRO_YOUT_H) & 0x46 (GYRO_YOUT_L)
  GyZ=Wire.read()<<8|Wire.read(); //0x47 (GYRO_ZOUT_H) & 0x48 (GYRO_ZOUT_L)
 // Mostra os valores no monitor serial
  Serial.print("Acel. X = "); Serial.print(AcX);
  Serial.print(" | Y = "); Serial.print(AcY);
  Serial.print(" | Z = "); Serial.print(AcZ);
  Serial.print(" | Gir. X = "); Serial.print(GyX);
  Serial.print(" | Y = "); Serial.print(GyY);
  Serial.print(" | Z = "); Serial.print(GyZ);
  Serial.print(" | Temp = "); Serial.println(Tmp/340.00+36.53);
 // Aguarda 300 milisegundos e reinicia o processo
  delay(300);
}
```

Sensores I2C: como funcionam

O módulo com acelerômetro e giroscópio que utilizamos é um tipo de sensor que utiliza um protocolo de comunicação I2C.

Este protocolo de comunicação utiliza dois fios: um para os dados e outro para o clock (ou relógio). Este protocolo permite que, utilizando apenas estes dois fios, múltiplos dispositivos sejam conectados a um único dispositivo central (neste caso, a placa Arduino), compartilhando as linhas de comunicação .

No exemplo, conectamos um acelerômetro e um giroscópio, lendo um total de 7 valores de entrada. Observe que, além da alimentação (GND e 5V), temos apenas dois fios enviando informações à placa: A5 e A4. Assim, no código, não fazemos a leitura diretamente dos sensores com os comandos analogRead() como costumávamos fazer. Neste caso, utilizamos a biblioteca Wire.h, que gerencia a comunicação com o sensor.
O módulo que apresentamos é um exemplo de sensor I2C. Entretanto, como ele, existem diversos outros módulos para Arduino que utilizam este tipo de comunicação. Todos eles apresentam uma característica em comum: o compartilhamento das portas para a comunicação com múltiplos sensores ou atuadores.

Comunicação Bluetooth

https://youtu.be/EmY2ibApJN8

Controlando um LED com o monitor serial

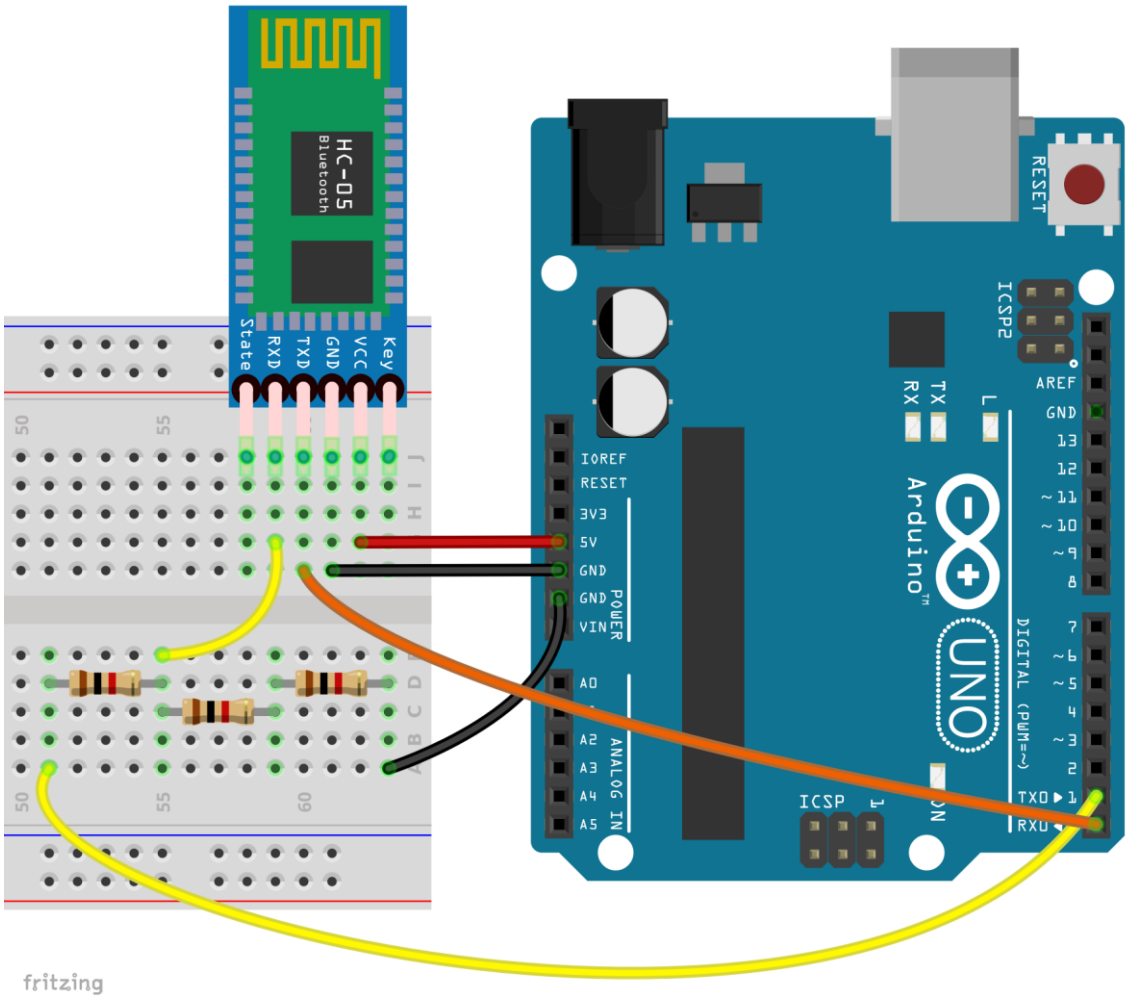
No vídeo, apresentamos uma forma de controlar o LED utilizando o monitor serial.
Nas semanas iniciais deste curso, utilizamos o monitor serial para ler valores de saída. No entanto, ele também pode ser usado para que façamos o oposto: ao invés de receber informações da placa, podemos enviar comandos a ela.
Neste exemplo não precisamos de nenhum circuito externo, já que podemos observar o comportamento do LED interno da placa (controlado pelo pino13).
Faça o upload do código que abaixo para a sua placa. Em seguida, abra o monitor serial, digite as letras l e d e observe o comportamento do LED.

```
/*
 Controle de LED por comunicação serial
*/
char valorSerial = '';
int pinoLED = 13;
void setup() {
   Serial.begin(9600);
   pinMode(pinoLED, OUTPUT);
}
void loop() {
   if (Serial.available() > 0) {
            valorSerial = Serial.read();
            if (valorSerial == 'l') {
                  digitalWrite(pinoLED, HIGH);
                  Serial.println("LED da Arduino ligado!");
            }
            else if (valorSerial == 'd') {
                  digitalWrite(pinoLED, LOW);
                  Serial.println("LED da Arduino desligado!");
            }
   }
   delay(1000);
}
```

Módulo HC-05: circuito

O HC-05 é um módulo que permite que a placa Arduino se comunique com outros dispositivos via Bluetooth.
A comunicação entre a Arduino e o HC-05 é feita por um protocolo de comunicação serial - o mesmo protocolo que usamos anteriormente para ler valores de sensores no Monitor Serial com o comando Serial.print e para acender o LED com o comando Serial.read.

Para configurar e usar o módulo HC-05 também usaremos o Monitor Serial. Mas, dessa vez, enviaremos as mensagens para o módulo, e não para o computador. As mensagens são enviadas pelos pinos RX e TX da Arduino (pinos digitais 0 e 1): TX é o pino que transmite informações e o RX é o pino que as recebe.
Para conectar o módulo bluetooth HC-05 à sua placa Arduino, utilize o seguinte circuito:



Os resistores são utilizados para diminuir a tensão do pino TX da Arduino para o pino RX do módulo, uma vez que a Arduino opera com 5V e o módulo com 3,3V.

Neste caso, vamos utilizar o mesmo programa que faz o LED piscar com os comandos do monitor serial, utilizado na unidade anterior. Assim, não é necessário fazer o upload de um novo código. No entanto, como há conflito entre a comunicação USB e a comunicação estabelecida entre o HC-05 e a Arduino, caso queira modificar o programa desconecte os pinos RX e TX da Arduino antes de fazer o upload do seu código. Após a finalização do upload, você pode conectar novamente os pinos RX e TX.

Configurando o HC-05 e enviando mensagens

Após montar o circuito, precisamos configurar o módulo. E, além do módulo e da Arduino, você também vai precisar de um celular. Aqui, vamos utilizar um celular com o sistema operacional Android.

1) Comece alimentando o circuito para verificar se ele funciona corretamente. Conecte o quarto pino ao GND da Arduino, e o quinto pino ao 5V*.

2) Uma vez energizado, o módulo já deve ser detectado pelo celular. No celular, vá nas Configurações de Bluetooth e habilite o Bluetooth. Após algum tempo, você deverá reconhecer o módulo HC-05.
3) Em seguida, você deverá realizar o pareamento. Selecione o dispositivo HC-05 na lista e insira o código 0000 quando solicitado.

* no vídeo é dito para conectar o VCC do HC-05 ao pino de 3,3V, mas essa informação não está correta. Apesar do módulo funcionar com 3,3V, a alimentação adequada deve estar entre 3,6V e 5V.

Agora que seu celular já está se comunicando com o módulo via Bluetooth, volte ao circuito e faça as demais conexões apresentadas na imagem da unidade anterior.

Como a Arduino já está executando o programa de comunicação serial que criamos anteriormente, se enviarmos mensagens do celular para o módulo HC-05, elas serão recebidas pela Arduino podendo ligar ou desligar o LED.

Para enviar essas mensagens, vamos baixar e usar o aplicativo Bluetooth Terminal - mas você pode usar outros aplicativos, se preferir.

1) Abra o aplicativo, selecione o dispositivo HC-05 e então clique em "Connect”
2) Agora que o aplicativo já está se comunicando com o módulo, podemos enviar mensagens com as letras "l" ou "d" para ligar ou desligar o LED.

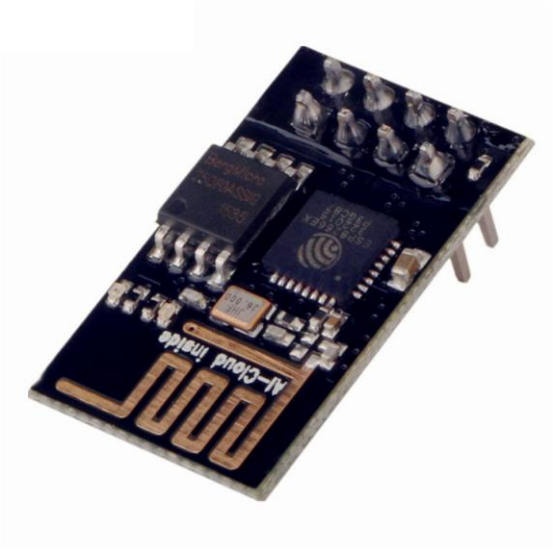
Observe que o programa é exatamente o mesmo da unidade anterior. No entanto, ao invés de digitar as informações no Monitor Serial, as digitamos no nosso celular.

Se quiser, você pode agora desconectar a Arduino do computador e alimentá-la com um powerbank. Observe que tudo funciona da mesma forma, ainda que a Arduino esteja conectada apenas a uma bateria.

O módulo ESP-01S

Nas aulas anteriores, aprendemos a utilizar o módulo Bluetooth HC-05 para realizar comunicação sem fio entre a placa Arduino e outros dispositivos, como o computador ou celular. Agora, apresentaremos um novo tipo de comunicação sem fio possível: a comunicação Wi-Fi. Para isso, utilizaremos o módulo Wi-Fi ESP-01S.

O ESP-01S é uma pequena placa (imagem abaixo) que permite que a Arduino acesse uma rede Wi-Fi, e que também pode ser programada de maneira autônoma para ter acesso à rede (sem necessariamente estar conectada a um microcontrolador).



Nesta semana, utilizaremos a ESP conectada à placa Arduino para controlá-la remotamente. Para isso, vamos seguir alguns passos para começar:

1) Conectar o módulo à placa Arduino.
2) Realizar configurações iniciais via monitor serial. Para isso, limparemos antes a programação da Arduino utilizando o programa Bare Minimum (em Arquivo > Exemplos) .
3) Testar a comunicação.
4) Configurar o nome da rede Wi-Fi e a senha, bem como o IP.
5) Enviar comandos via navegador para ligar e desligar um LED, ou para ler no navegador os valores de leitura de um sensor.

Nas próximas unidades você terá orientações detalhadas sobre cada um destes passos.

ESP-01S: configuração

https://youtu.be/Uq09fMLIjM0

ESP-01S: circuito

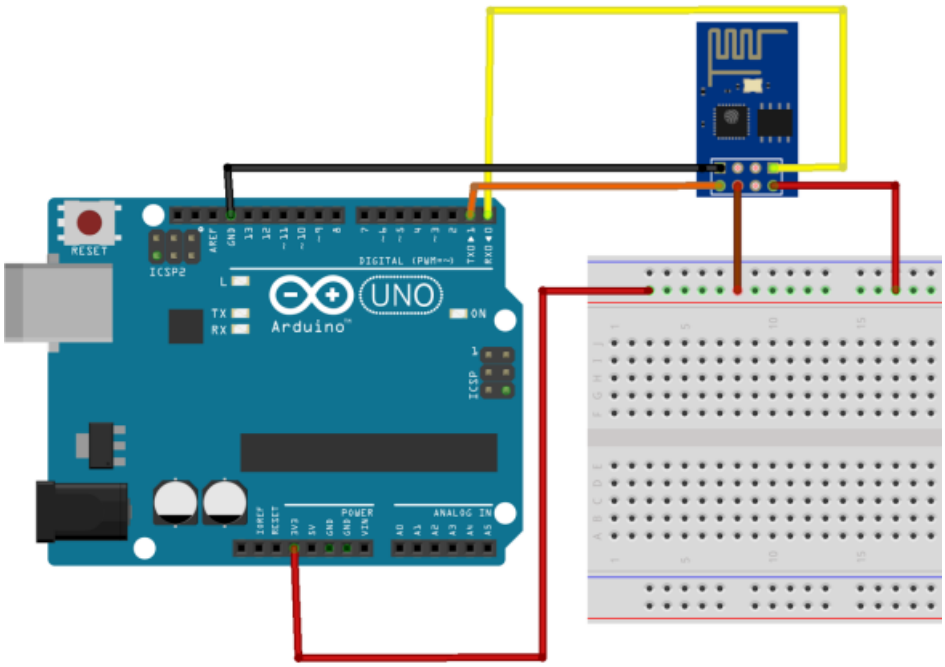
Abaixo apresentamos o circuito para configuração inicial do módulo ESP-01S.

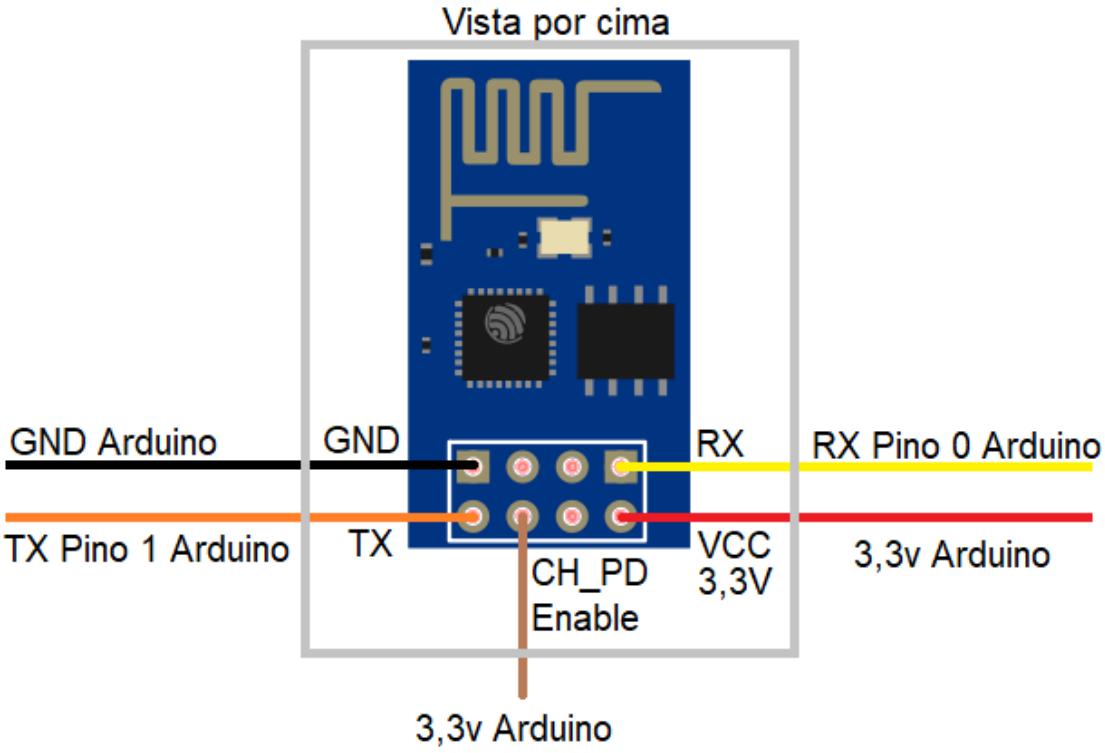
Materiais necessários:

- Arduino Uno
- ESP-01S
- protoboard
- 5 jumpers macho-fêmea
- 1 jumper macho-macho

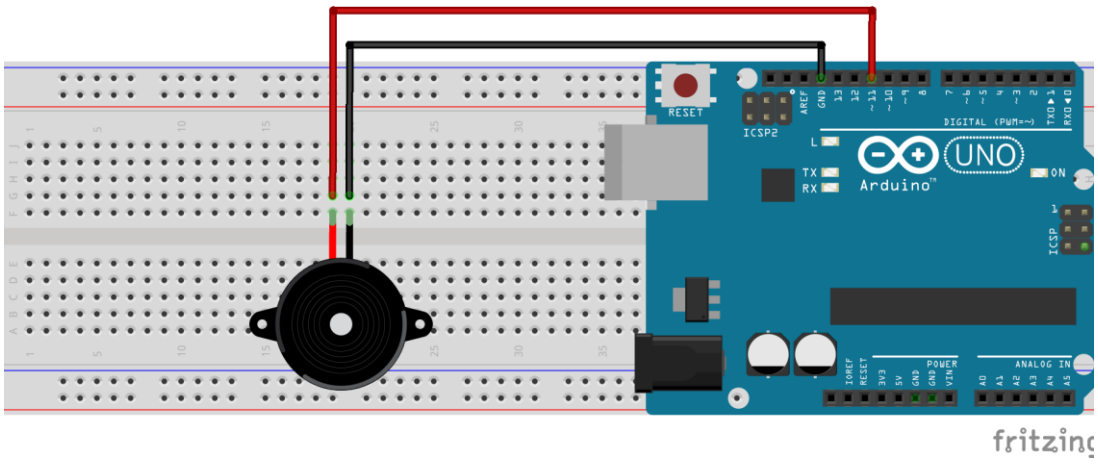
Começando

Começe conectando a Arduino ao seu computador. Abra a IDE e carregue para a sua placa o exemplo que está em Arquivo > Exemplos > 01.Basics > BareMinimum.
Transferimos este código para a Arduino antes de conectá-la à ESP-01S, para garantir que ela não usará a comunicação serial e que a ESP será capaz de se comunicar corretamente.
Em seguida, após desconectar o cabo USB, monte o seguinte circuito:





Observe que o pino TX da Arduino está conectado ao pino TX da ESP, bem como o pino RX da Arduino conecta-se ao pino RX da ESP.

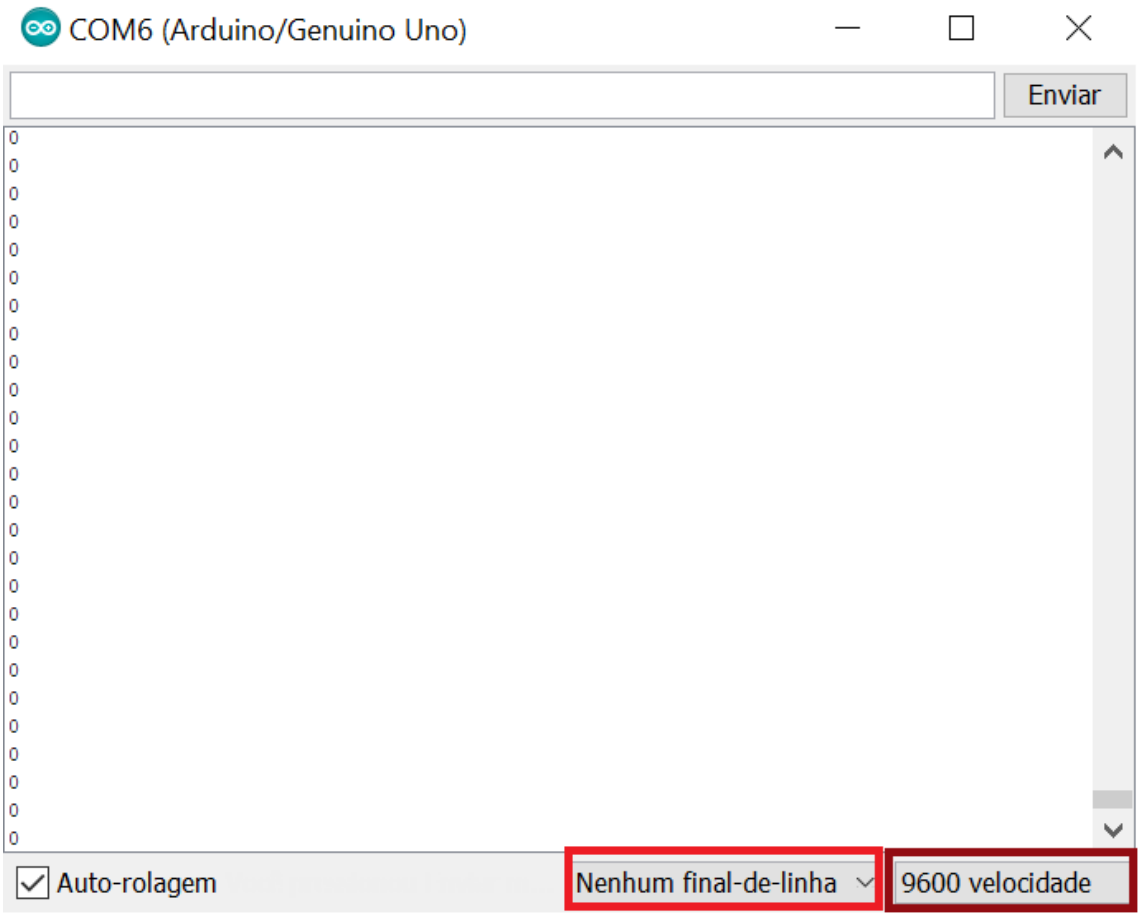


ESP-01S: comandos de configuração

Configuração:

Após montar o circuito, conecte novamente sua Arduino ao computador via USB.
Abra o monitor serial. Nas caixas de seleção inferiores:

- Mude a opção de “Nenhum final-de-linha” para “Ambos, NL e CR”.
- Mude a velocidade para “115200 Velocidade”.



Agora já podemos testar a comunicação: digite AT no espaço para enviar comandos e veja se recebe a resposta OK.
Vamos agora garantir que a ESP-01S está com a configuração de fábrica, restaurando sua configuração. Para isso, usamos o comando AT+RESTORE.
Em seguida:
Digite o comando, para mudar a velocidade para 9600: AT+CIOBAUD=9600
Na caixa de seleção no inferior da janela, mude a velocidade para “9600 Velocidade”
Teste a comunicação, no lado superior da janela digite AT e veja a resposta OK.
Agora vamos configurar o módulo em Station Mode para atuar como cliente da rede Wi-Fi, com o comando AT+CWMODE=1.

Observação:

Acima, utilizamos o número 1 para que o módulo atue como cliente da rede. Veja as opções possíveis de modos para a ESP:
STA = 1 - o módulo conecta-se a rede e age como cliente dela
AP = 2 - o módulo fornece uma rede, atuando como Access Point (AP)
Both = 3 - o módulo pode atua tanto como Access Point quanto como Cliente
Agora podemos conectar o módulo à rede Wi-Fi.
Para isso, utilize os comandos abaixo, substituindo o "nome_da_rede" pelo nome da sua rede Wi-Fi e "senha_da_rede" pela sua senha. Mantenha as aspas.
AT+CWJAP="nome_da_rede","senha_da_rede"
Se você visualizar a resposta abaixo, sua conexão foi estabelecida corretamente:
WIFI CONNECTED
WIFI GOT IP

Descobrindo o IP

Para o próximo exercício, precisaremos saber o endereço IP do nosso módulo. Para descobrir o IP, execute o comando AT+CIFSR e anote o valor de saída.

Controlando a Arduino via Wi-Fi

https://youtu.be/vmwYWOIaByA

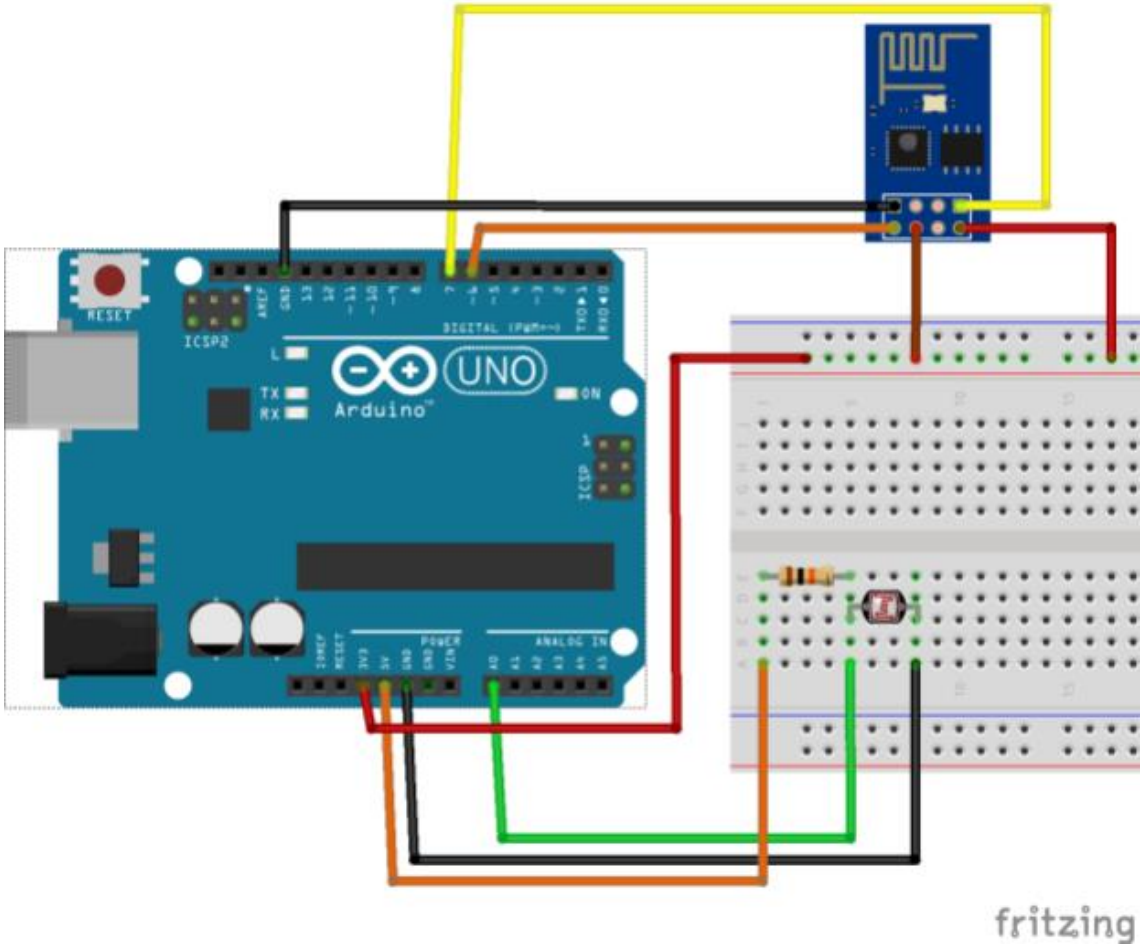
Controle via Wi-Fi: circuito

Agora que você já aprendeu a configurar o módulo ESP-01S, vamos fazer um projeto conectando a placa Arduino à Internet.

Para o projeto apresentado no vídeo, você precisará de:

- Arduino
- Módulo ESP-01S
- 5 Jumpers macho-fêmea
- 4 Jumpers macho-macho
- 1 LDR
- 1 resistor de 10k

Para começar, monte o seguinte circuito:



Observe que, neste caso, o RX da ESP conecta-se ao TX da Arduino e vice-versa (ao contrário do circuito de configuração utilizado anteriormente).

Em seguida, podemos conectar a Arduino ao computador e transferir o código que consta na próxima unidade.

Controle via Wi-Fi: código

```
/*
 * Configura um servidor na ESP-01S; a ideia é, pelo navegador, permitir:
 *   - controlar um atuador (liga ou desliga LED da Arduino);
 *   - ler um sensor (abre uma página com valor de leitura de conversão AD).
 */
#include "SoftwareSerial.h"
#define TIMEOUT 5000 // mS
#define LED 13
SoftwareSerial mySerial(6, 7); // RX, TX
const int button = 8;
const int LDR = A0;
int LDR_value = 0;
void setup()
{
 pinMode(LED,OUTPUT);
 Serial.begin(9600);
 mySerial.begin(9600);
 SendCommand("AT+RST", "Ready");
 delay(5000);
 SendCommand("AT+CWMODE=1","OK");
 SendCommand("AT+CIFSR", "OK");
 SendCommand("AT+CIPMUX=1","OK");
 SendCommand("AT+CIPSERVER=1,80","OK");
}
void loop(){
 LDR_value = analogRead(LDR);
 String IncomingString="";
 boolean StringReady = false;
 while (mySerial.available()){
   IncomingString=mySerial.readString();
   StringReady= true;
 }
 if (StringReady){
   Serial.println("Received String: " + IncomingString);
 if (IncomingString.indexOf("LED=ON") != -1) {
   digitalWrite(LED,HIGH);
   mySerial.println("AT+CIPSEND=0,18");
   delay(100);
   mySerial.println("<h1>LED Aceso</h1>");
   delay(1000);
   SendCommand("AT+CIPCLOSE=0","OK");
  }

 if (IncomingString.indexOf("LED=OFF") != -1) {
   digitalWrite(LED,LOW);
   mySerial.println("AT+CIPSEND=0,20");
   delay(100);
   mySerial.println("<h1>LED Apagado</h1>");
   delay(1000);
   SendCommand("AT+CIPCLOSE=0","OK");
  }
 if(IncomingString.indexOf("LDR") != -1)    {
   char valueStr[4];
   sprintf(valueStr,"%04d",LDR_value);
   Serial.println(valueStr);
   mySerial.println("AT+CIPSEND=0,4");
   delay(100);
   mySerial.println(valueStr);
   delay(1000);
   SendCommand("AT+CIPCLOSE=0","OK");
```

```
  }
 }
}
boolean SendCommand(String cmd, String ack){
 mySerial.println(cmd); // Send "AT+" command to module
 if (!echoFind(ack)) // timed out waiting for ack string
  return true; // ack blank or ack found
}
boolean echoFind(String keyword){
 byte current_char = 0;
 byte keyword_length = keyword.length();
 long deadline = millis() + TIMEOUT;
 while(millis() < deadline){
  if (mySerial.available()){
   char ch = mySerial.read();
   Serial.write(ch);
   if (ch == keyword[current_char])
    if (++current_char == keyword_length){
     Serial.println();
     return true;
   }
  }
 }
 return false; // Timed out
}
```

Controle via Wi-Fi: testes

Após montar o circuito e fazer o upload do código, podemos abrir o navegador (ex.: Firefox, Google Chrome) para testar o nosso programa.
Lembra-se do IP que você anotou anteriormente? Cole- o no navegador para testar alguns comandos.

- Cole no seu navegador o endereço IP seguido de "/LDR" ao final (sem as aspas). O que você observa?
- Escreva "LED=ON" (sem as aspas) e observe o que acontece com o LED interno da placa Arduino.
- Agora escreva "LED=OFF". O que acontece?

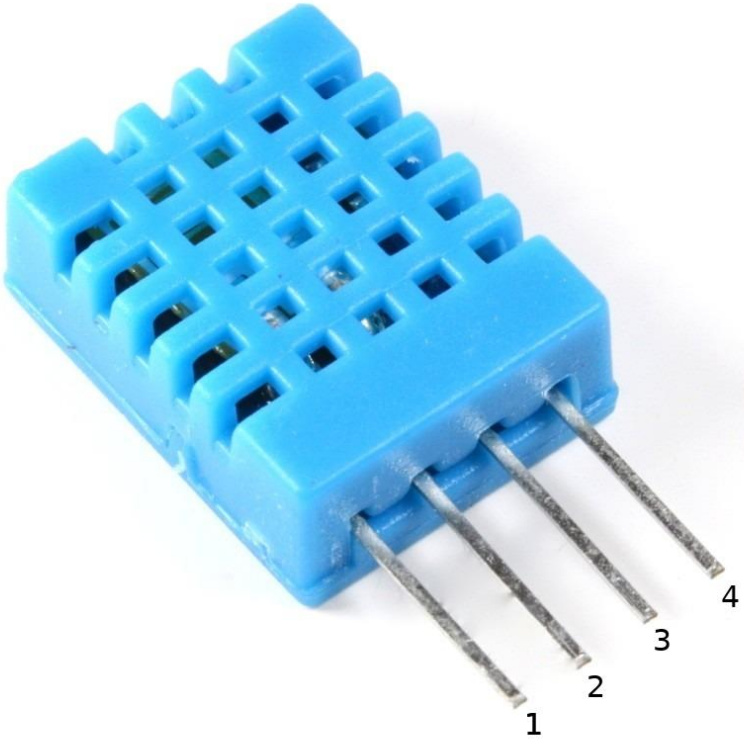
Observação: Você precisará recarregar a página sempre que quiser atualizar o valor de leitura do LDR. Não se preocupe se não visualizar a informação imediatamente, pois o envio destes dados pode demorar alguns segundos.

Conectando a Arduino à internet

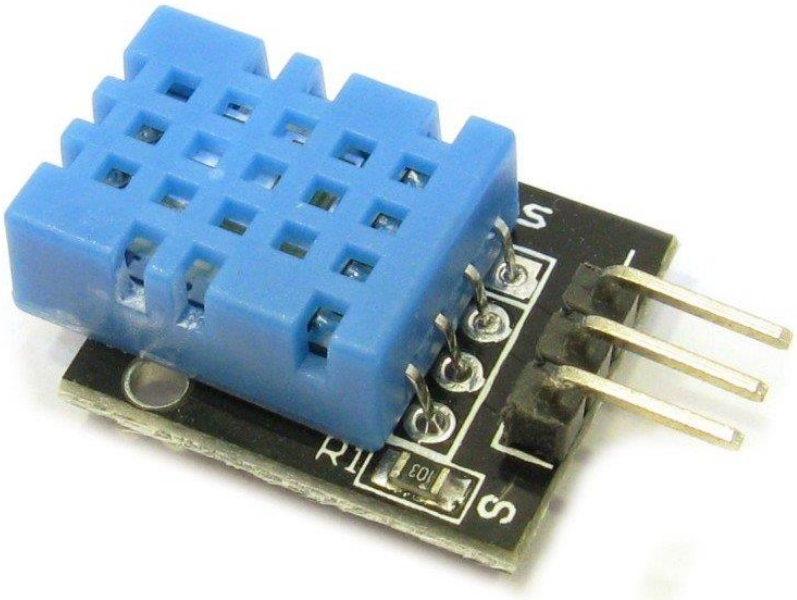
https://youtu.be/W6IC_27ja6k

O sensor DHT11

O DHT11 é um sensor de temperatura e umidade muito utilizado por seu baixo custo e relativa facilidade de uso. Ele tem três componentes principais: um sensor capacitivo de umidade, um termistor e um chip simples que converte as leituras analógicas realizadas para um sinal digital, que é o sinal que lemos usando nosso microcontrolador.
O DHT11 realiza leituras de umidade entre 20-80, com 5% de precisão, e leituras de temperatura entre 0-50°C, com ±2°C de precisão. Ele tem quatro terminais. O primeiro terminal (1) é VCC, o segundo (2) é o terminal de dados e o quarto (4) é terra ou GND. Seu terceiro terminal não está conectado à nada.



É também possível encontrá-lo em módulos que expõem apenas os terminais de VCC, dados e GND, como o usado no vídeo dessa aula:



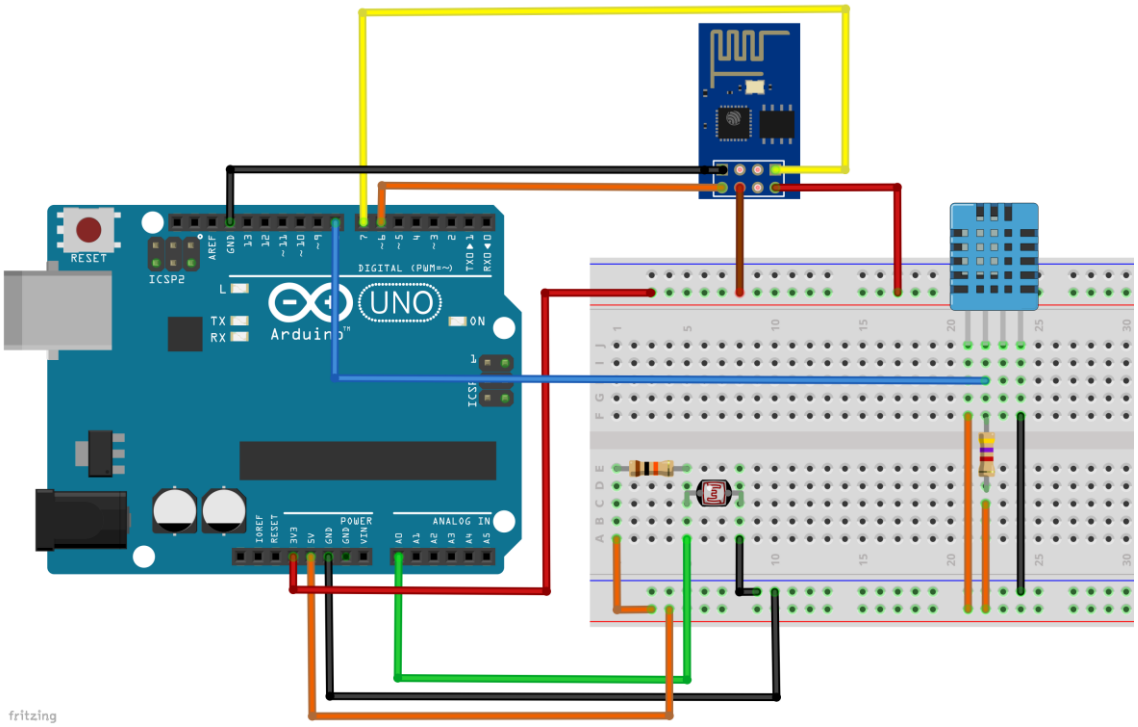
Sensor DHT11: circuito

Vamos montar um circuito que lê a umidade, a temperatura e a luminosidade de um ambiente e que envia essas informações para um servidor na Internet, usando a Arduino e o módulo WiFi ESP-01S.

Os componentes necessários serão:

- O circuito montado na aula passada
- 4 jumpers
- 1 resistor de 4.7K Ohms
- 1 sensor DHT11

Monte um circuito como o da figura abaixo:



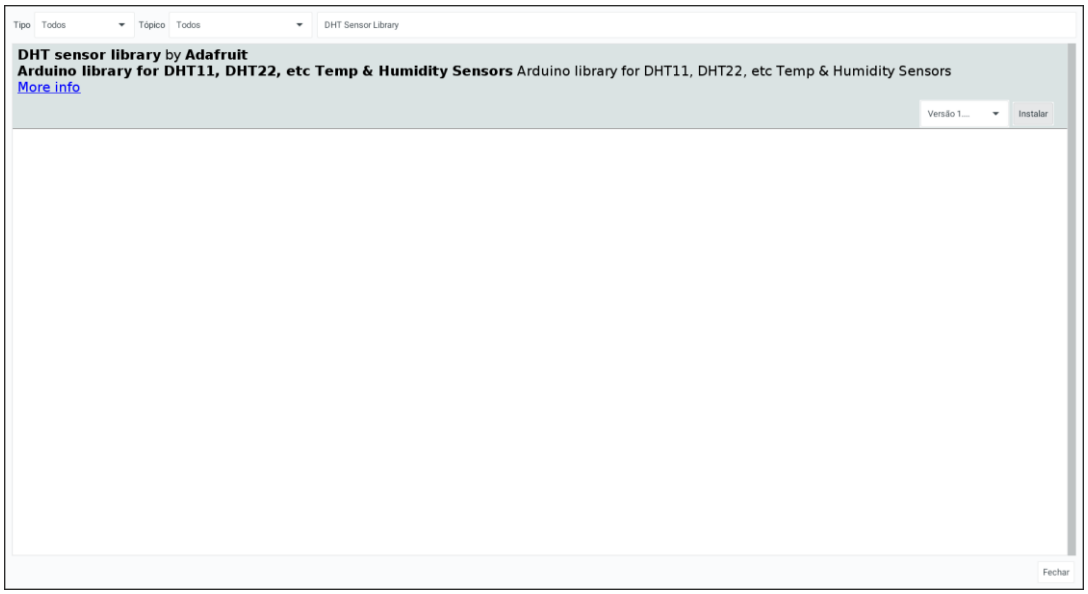
Note que caso você tenha um módulo do sensor DHT11 com três terminais ao invés do sensor com 4 terminais, a montagem deve ter o primeiro terminal ligado à VCC, o segundo ligado ao resistor e o terceiro ligado à GND.

Sensor DHT11: bibliotecas e código

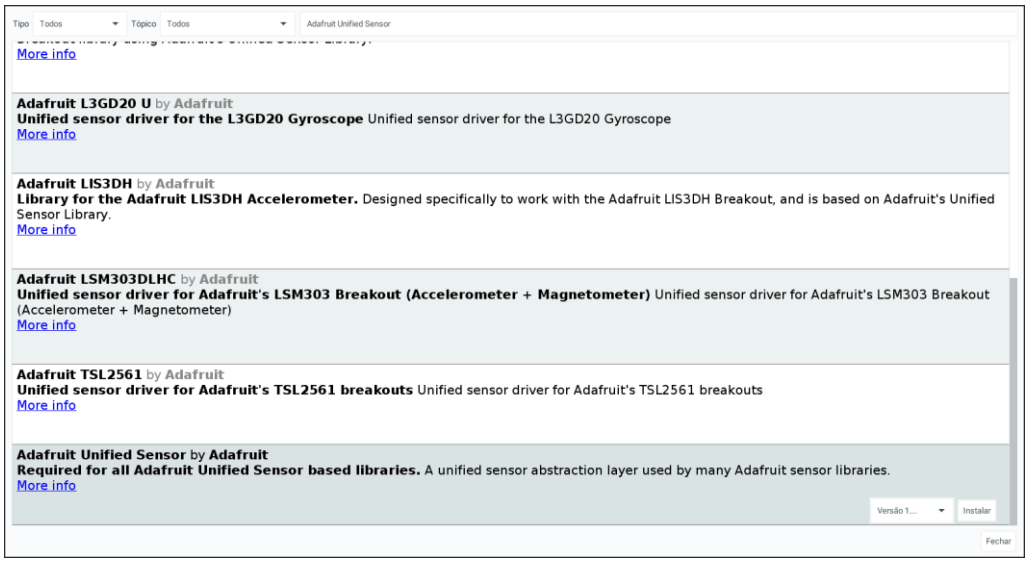
Instalação de Bibliotecas

Para realizar a leitura dos dados enviados pelo DHT11, usaremos a biblioteca DHT sensor library. Precisaremos também instalar a biblioteca Adafruit Unified Sensor, que é usada pela DHT sensor library. Para isso, realize os seguintes passos:

1. Abra o ambiente de desenvolvimento da Arduino
2. No menu superior, selecione Sketch -> Incluir Biblioteca -> Gerenciar Bibliotecas
3. Na caixa de pesquisa, digite 'DHT sensor library',
4. Encontre a biblioteca DHT sensor library na lista de bibliotecas e clique em instalar.



5. Na caixa de pesquisa, digite 'Adafruit Unified Sensor',
6. Encontre a biblioteca Adafruit Unified Sensor no final da lista de bibliotecas e clique em instalar.



Código
Com as bibliotecas instaladas, você poderá transferir o programa abaixo para sua placa Arduino para testar o funcionamento do módulo DHT11:

```
// testeDHT.ino
// Programa para teste do DHT11, baseado no exemplo
// escrito por ladyada
#include "DHT.h"
int pinoDHT = 8;        // Pino em que conectamos o pino central do DHT11
int tipoDHT = DHT11;    // Estamos usando o sensor DHT11
// Inicializa o sensor
DHT dht(pinoDHT, tipoDHT);
void setup() {
  Serial.begin(9600);
  Serial.println("Teste do DHT11");
  dht.begin();
}
void loop() {
```

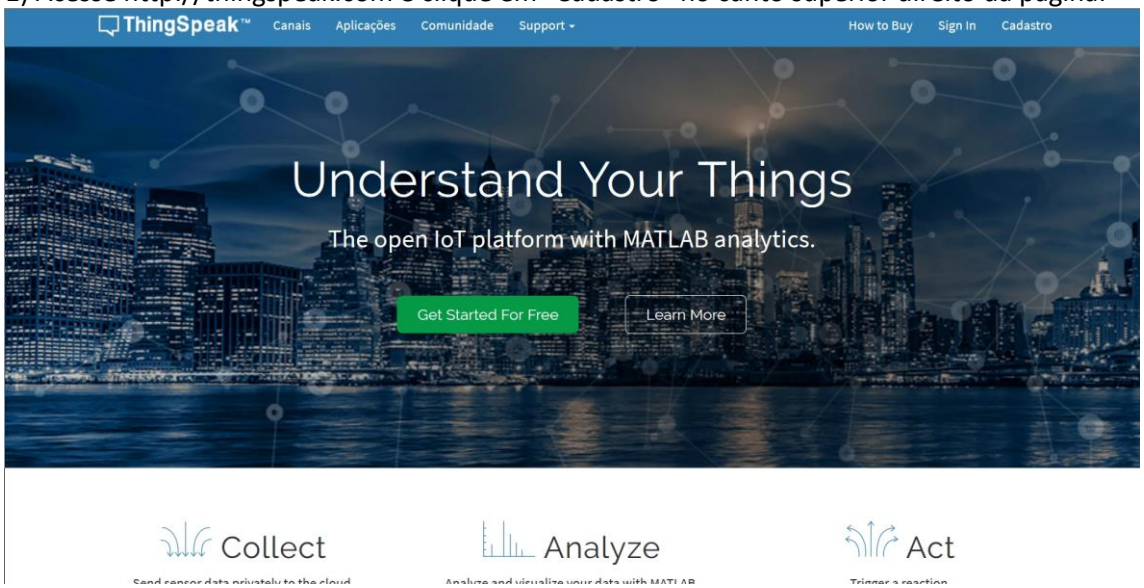
```
    // Espera alguns segundos entre leituras
    delay(2000);
    // A leitura de umidade e temperatura leva cerca de 250ms
    float h = dht.readHumidity();
    //Lê a temperatura como graus Celsius
    float t = dht.readTemperature();
    // verifica se alguma das leituras falhou (para tentar novamente).
    if (isnan(h) || isnan(t)) {
      Serial.println("Falha na leitura do sensor DHT11");
      return;
    }
    // imprime os resultados
    Serial.print("Umidade relativa do ar: ");
    Serial.print(h);
    Serial.print(" %\t");
    Serial.print("Temperatura: ");
    Serial.print(t);
    Serial.println(" *C ");
}
```

Thingspeak: criação de conta e canal

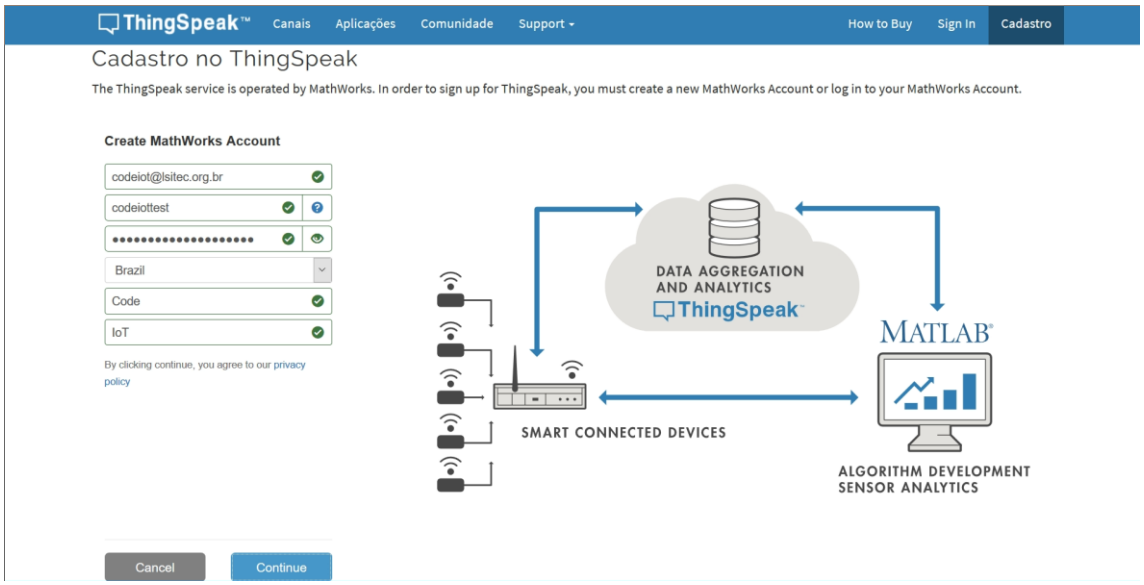
Criação de conta

O ThingSpeak é uma plataforma para a Internet das Coisas (IoT) que permite a coleta e armazenamento na nuvem de dados de sensores para a criação de aplicações em IoT. Para enviarmos as informações dos sensores que conectamos à Arduino nas últimas unidades, precisaremos criar uma conta gratuita no Thingspeak, com os seguintes passos:

1) Acesse http://thingspeak.com e clique em "Cadastro" no canto superior direito da página.



2) Insira os dados solicitados e clique em "Continue"

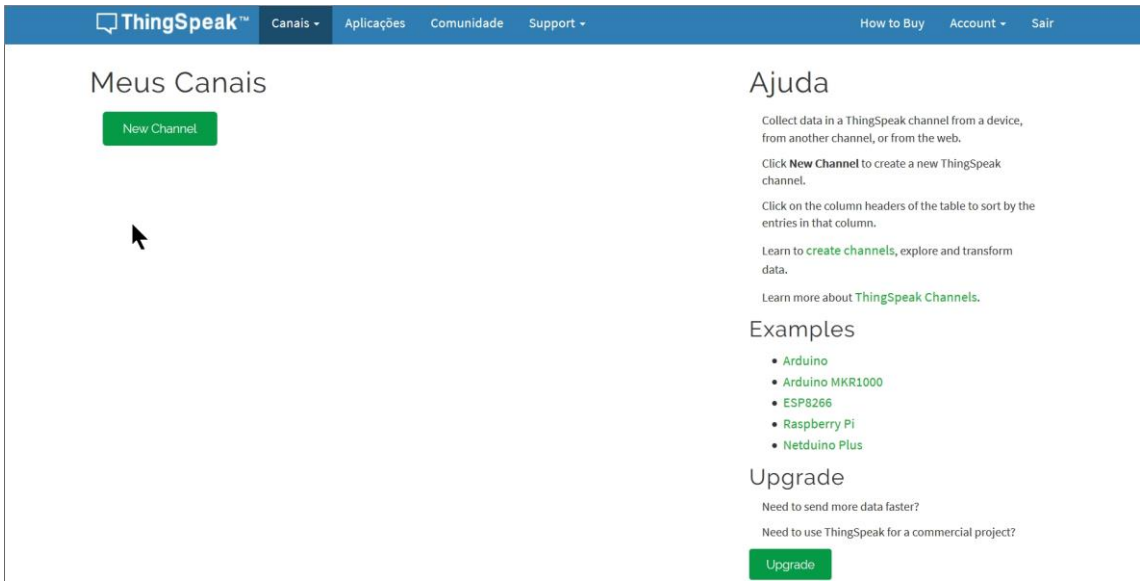


3) Você receberá uma mensagem no endereço de email informado (essa mensagem pode ir para sua caixa de Spam, não deixe de verificá-la também). Clique no link dessa mensagem para confirmar sua conta. Continue seu cadastro e aceite os termos de uso do serviço.

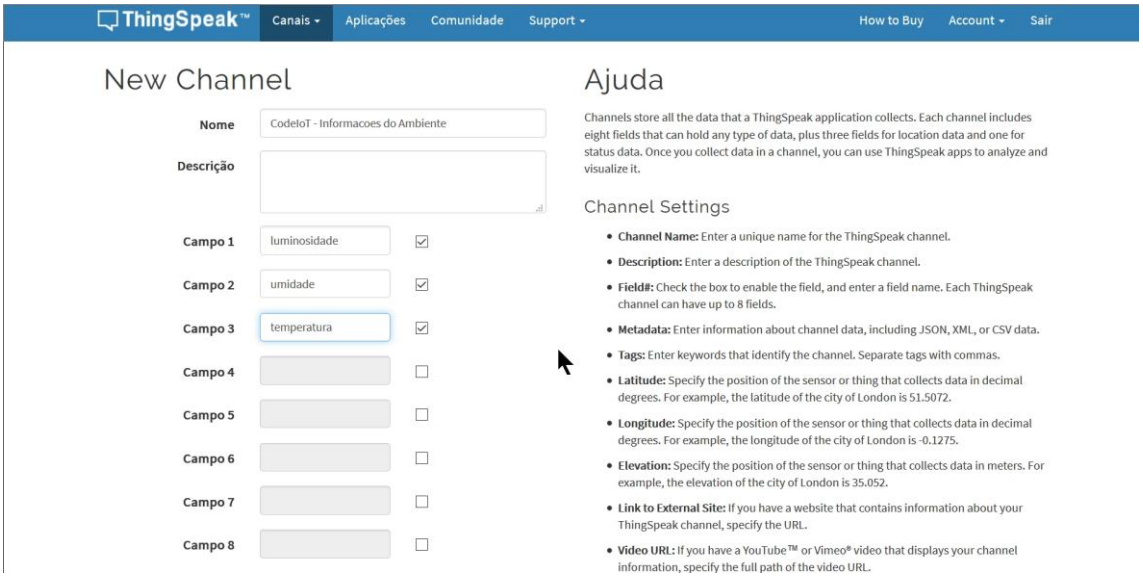
Criação de Canal

Precisamos criar um canal no Thingspeak para poder enviar e ler informações. Você pode criar quantos canais precisar, mas no nosso exemplo usaremos apenas um. Para criar esse canal, siga as instruções abaixo:

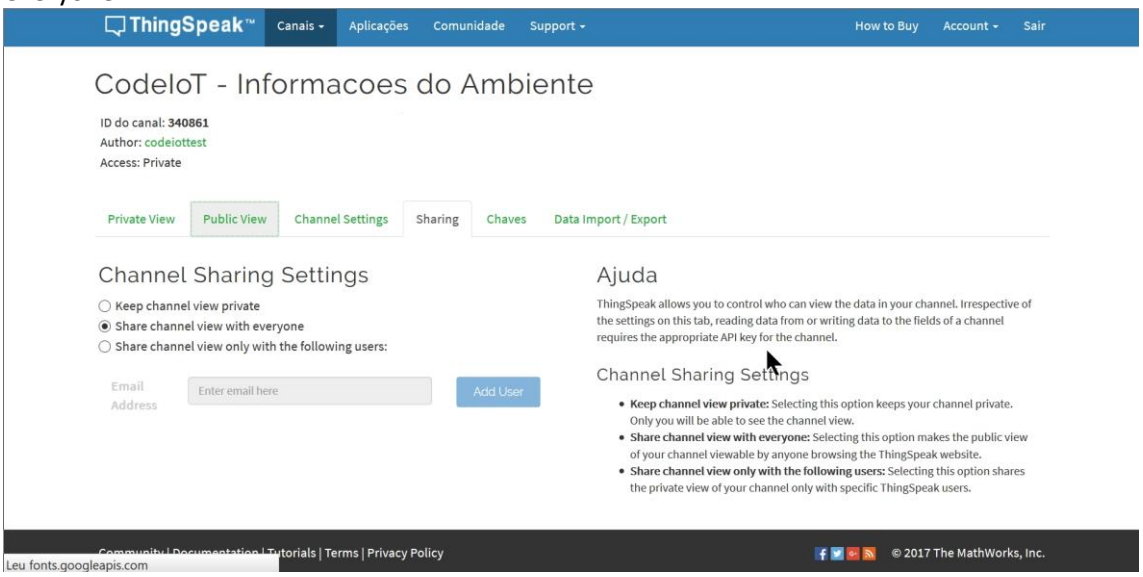
1) Ao fazer login no Thingspeak, você será direcionado para a tela de "Meus Canais". Clique em "New Channel".



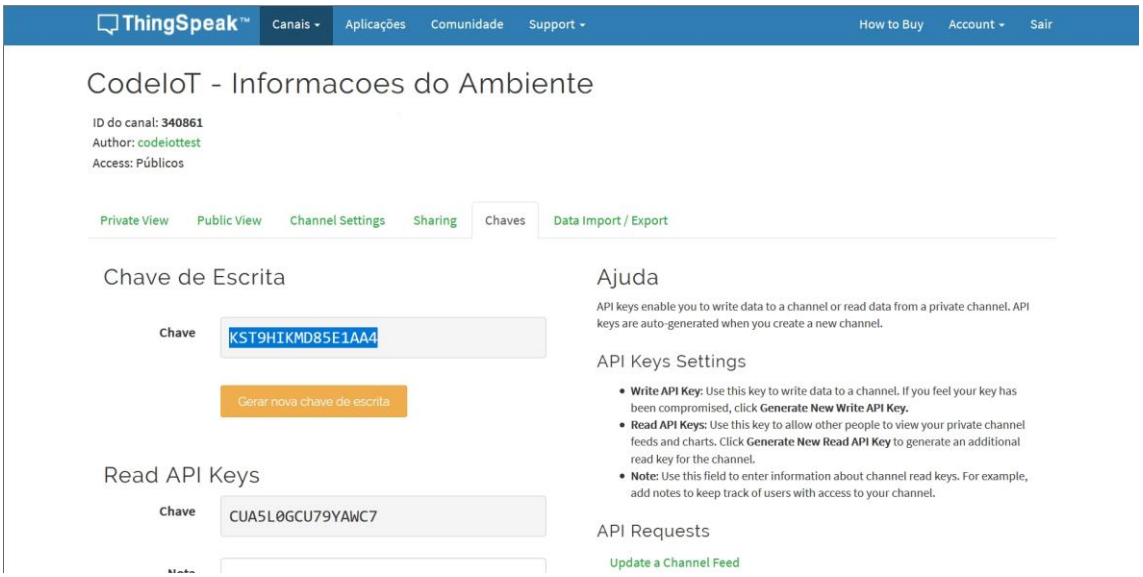
2) Escolha um nome para seu canal, e dê os nomes "luminosidade", "umidade" e "temperatura" para os primeiros três campos. Antes de dar o nome, é necessário marcar a caixa de seleção ao lado do campo. Por fim, clique em "Save Channel" no final da página.



3) Por padrão, as informações do seu canal são privadas. Para poder compartilhar com seus colegas acesse a aba "Sharing" na página do canal e selecione a opção"Share channel view with everyone"



4) Para enviarmos dados para o Thingspeak precisamos informar no nosso Sketch qual é a chave de escrita do canal. Na página do canal, acesse "Chaves". Anote o valor de "Chave de escrita" que será mostrado em seguida. Você vai precisar dessa informação quando formos escrever nosso programa.



Arduino e Thingspeak : código

Código para comunicação com a plataforma Thingspeak

Transfira e execute o código abaixo na sua Arduino e verifique se seu canal no Thingspeak é atualizado com as informações lidas pelos sensores que montamos no circuito.
É importante notar que o código abaixo assume que seu módulo Wi-Fi ESP-01S já tenha recebido a configuração inicial e já esteja conectado à uma rede Wi-Fi, conforme descrito na aula 11.

```
// Thingspeak.ino: comunicacao entre Arduino e a Nuvem!
//
// Baseado no programa esp8266_test de Mahesh Venkitachalam

#include "DHT.h"
#include <stdlib.h>
#include <SoftwareSerial.h>
int pinoLED = 13;
int pinoLDR = A0;
int pinoDHT = 8;
int tipoDHT =  DHT11;
DHT dht(pinoDHT, tipoDHT); // Inicializa o sensor
SoftwareSerial EspSerial(6, 7); // Inicializa a porta de comunicacao com modulo WiFi
// substitua pela chave de leitura do seu canal no Thingspeak
String chaveDeEscrita = "INSIRA_AQUI_SUA_CHAVE_DE_ESCRITA_DO_THINGSPEAK";
void setup() {
  pinMode(pinoLED, OUTPUT);
  Serial.begin(9600); // Comunicacao com Monitor Serial
  EspSerial.begin(9600); // Comunicacao com Modulo WiFi
  dht.begin();
  EnviarComando("AT+RST", "Ready"); //Reset do Modulo WiFi
}
void loop() {
  float luminosidade = 1024.0f - analogRead(pinoLDR); // valores baixos: luminosidade baixa
  float umidade = dht.readHumidity();
  float temperatura = dht.readTemperature();
  // converte leiturasnpara String
  String strLuminosidade = floatToString(luminosidade);
  String strUmidade = floatToString(umidade);
  String strTemperatura = floatToString(temperatura);
  // Conexao com TCP com Thingspeak
```

```
    String cmd = "AT+CIPSTART=\"TCP\",\"";
    cmd += "184.106.153.149"; // Endereco IP de api.thingspeak.com
    cmd += "\",80";
    EspSerial.println(cmd);
    if(EspSerial.find("Error")){
      Serial.println("AT+CIPSTART error");
      return;
    }
    // preparacao da string GET
    String getStr = "GET /update?api_key=";
    getStr += chaveDeEscrita;
    getStr +="&field1=";
    getStr += String(strLuminosidade);
    getStr +="&field2=";
    getStr += String(strUmidade);
    getStr +="&field3=";
    getStr += String(strTemperatura);
    getStr += "\r\n\r\n";
    // envia o tamanho dos dados
    cmd = "AT+CIPSEND=";
    cmd += String(getStr.length());
    EspSerial.println(cmd);
    // envia comando de GET
    if(EspSerial.find(">")){
      EspSerial.print(getStr);
    }
    else{
      EspSerial.println("AT+CIPCLOSE");
      // alerta usuario
      Serial.println("AT+CIPCLOSE");
    }
    Serial.println("Comando enviado ao Thingspeak:");
    Serial.println(getStr);
    // pisca o LED a cada envio
    digitalWrite(pinoLED, HIGH);
    delay(200);
    digitalWrite(pinoLED, LOW);
    // o thingspeak precisa de pelo menos 16 segundos entre atualizacoes
    delay(16000);
}
String floatToString(float f) {
    char buf[16];
    return dtostrf(f, 4, 1, buf);
}
boolean EnviarComando(String cmd, String ack){
    EspSerial.println(cmd); // Envia comando "AT+" para o modulo
    if (!echoFind(ack)) // tempo de espera para string de ack
      return true; // ack vazio ou ack encontrado
}
boolean echoFind(String keyword){
  byte current_char = 0;
  byte keyword_length = keyword.length();
  long deadline = millis() + 5000; // Tempo de espera 5000ms
  while(millis() < deadline){
    if (EspSerial.available()){
      char ch = EspSerial.read();
      Serial.write(ch);
      if (ch == keyword[current_char])
        if (++current_char == keyword_length){
          Serial.println();
          return true;
      }
```

```
      }
    }
  return false;  // Tempo de espera esgotado
}
```

Referências:

- Material integralmente extraído e adaptado do curso **Programação Física**, da plataforma 'Code IoT', criado em parceria com a Samsung e LSI-TEC Escola Politécnica da USP - https://codeiot.org.br/courses/course-v1:LSI-TEC+IOT104+2020_O2/about - Acessado em 20/11/2020.
    - Licença disponível em https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR - Acessado em 20/11/2020.
    - Plano de aula disponível em https://codeiot.org.br/assets/courseware/v1/fa2b33021964a8f58fbca2b8e4ee677d/asset-v1:LSI-TEC+IOT104+2018_S2+type@asset+block/Plano_de_Aula_Progamacao_Fisica_Semana_4.pdf - Acessado em 20/11/2020.