AULA 10 - IoTA

Programação física 2
O que vamos aprender 2
Por que precisamos usar programação? 2
Instalação da Arduino IDE 2
1. Instalação da IDE 2
2. Instalação do driver 2
Criando o nosso primeiro programa 2
Como programar a placa Arduino usando a IDE 2
Alternativa à IDE: MBlock 3
Começando 4
Programando 6
Finalizando 8
Estrutura geral do código Arduino 8
O conceito atrás do Blink 9
Apresentação da Arduino 9
Principais partes de uma placa Arduino 9
Conectando a Arduino a um outro circuito 9
Como explorar exemplos da IDE Arduino 10
Referências: 11

Programação física

O que vamos aprender

https://youtu.be/y1EVu-UGQC8

Por que precisamos usar programação?

https://youtu.be/LT6uKENV95o

Instalação da Arduino IDE

Antes de começar a usar e programar sua placa Arduino, você deverá seguir os dois passos abaixo: instalar a IDE e instalar o driver. Veja aqui como proceder.

1. Instalação da IDE

Arduino IDE é o software que permite que você programe sua placa Arduino.
Para instalá-lo, faça o download do arquivo correspondente ao seu sistema operacional nos links abaixo:

- Windows Installer
- Mac OS X 10.7 Lion ou mais recente
- Linux 32 bits
- Linux 64 bits

Ao clicar nos links, você será direcionado para uma página na qual são solicitadas doações para a manutenção do projeto Arduino. Caso não esteja interessado em fazer uma doação, clique no botão "Just Download" para iniciar seu download.
Após fazer o download dos arquivos, siga os procedimentos padrão para instalação do software. Caso você precise de outra configuração de instalação, visite a página https://www.arduino.cc/en/Main/Software

2. Instalação do driver

Além de instalar a IDE, você deve também instalar o driver que permitirá a comunicação do seu computador com a placa Arduino.
Se estiver utilizando Mac ou Linux, não é necessário fazer a instalação do driver. Caso esteja utilizando o Windows, siga o seguinte procedimento:
Conecte sua placa Arduino ao seu computador usando o cabo USB.
Uma janela surgirá (Assistente para Adicionar Novo Hardware, ou "Found New Hardware Wizard"), e o Windows deverá reconhecer e instalar automaticamente os drivers.

Criando o nosso primeiro programa

https://youtu.be/TxFC1mhPmc4

Como programar a placa Arduino usando a IDE

Conforme apresentado, para programar sua placa Arduino você precisará utilizar a IDE. Após fazer as instalações necessárias, siga os seguintes passos para criar seu primeiro programa:

- Conecte a placa Arduino a uma entrada USB do seu computador
- Abra o software “Arduino” (Arduino IDE).
- Verifique se sua placa foi identificada. Ela deve aparecer na lista de portas disponíveis em Ferramentas > Porta (ou Tools > Port).

- Em Ferramentas > Porta, selecione a porta na qual seu Arduino está conectado (dependendo da sua placa, uma identificação aparecerá à direita do nome da porta).
- Verifique se a sua placa está definida corretamente em Ferramentas > Placa (ou Tools > Board). Neste curso, estamos trabalhando com a placa Arduino Uno ou similar e, portanto, selecionaremos a opção “Arduino/Genuino Uno”.
- Insira o código que deseja passar para o Arduino na área de programação. Para começar, você pode copiar o código abaixo e colá-lo na área de programação, substituindo o conteúdo pré-existente.
- Clique em Carregar (ou Upload, no ícone com a seta para a direita)
- Se tudo der certo, você verá a mensagem “Done uploading” abaixo da área de programação.

Código do programa "Blink"

```
// the setup function runs once when you press reset or power the board
void setup() {
  // initialize digital pin 13 as an output.
  pinMode(13, OUTPUT);
}
// the loop function runs over and over again forever
void loop() {
  digitalWrite(13, HIGH);   // turn the LED on (HIGH is the voltage level)
  delay(1000);              // wait for a second
  digitalWrite(13, LOW);    // turn the LED off by making the voltage LOW
  delay(1000);              // wait for a second
}
```

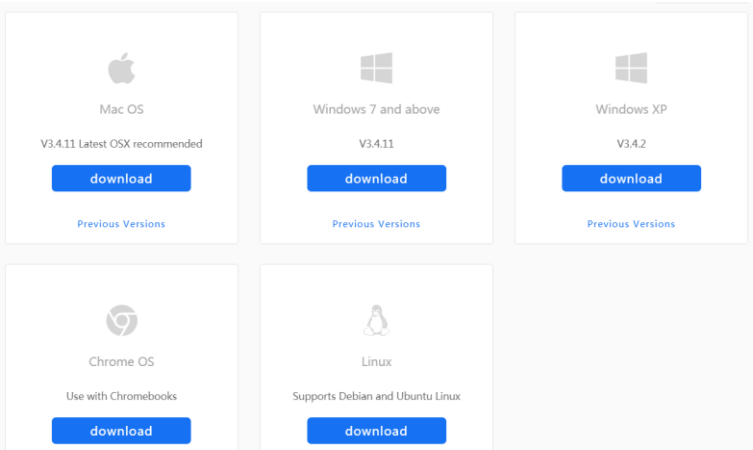
Alternativa à IDE: MBlock

O que é o MBlock?

MBlock é um programa baseado em Scratch que permite que você controle e programe sua placa Arduino por meio de uma interface de programação em blocos.
Assim, caso você não se sinta confortável para começar a programar a Arduino diretamente pela IDE, pode iniciar pelo MBlock para se familiarizar com a lógica de programação para Arduino utilizando blocos similares aos do Scratch. No futuro, quando estiver mais confortável e sentir que esta interface não atende mais às suas necessidades, pode passar à IDE para criar projetos mais complexos.

Instalando o Mblock

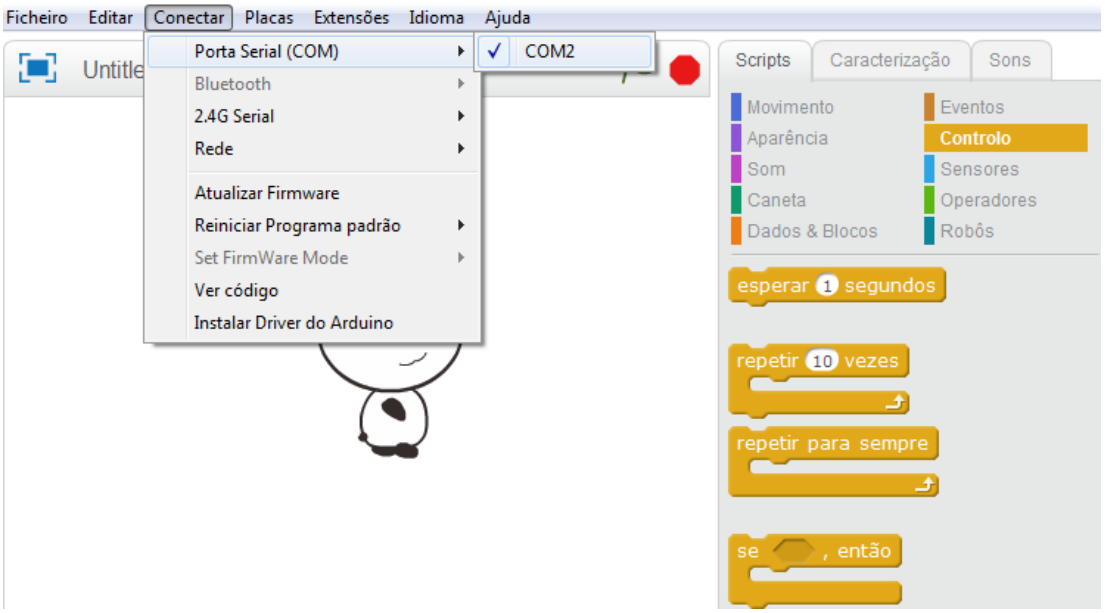
Para instalar o Mblock, dirija-se ao site http://www.mblock.cc/software/mblock/mblock3/ e selecione o seu sistema operacional para fazer o download.



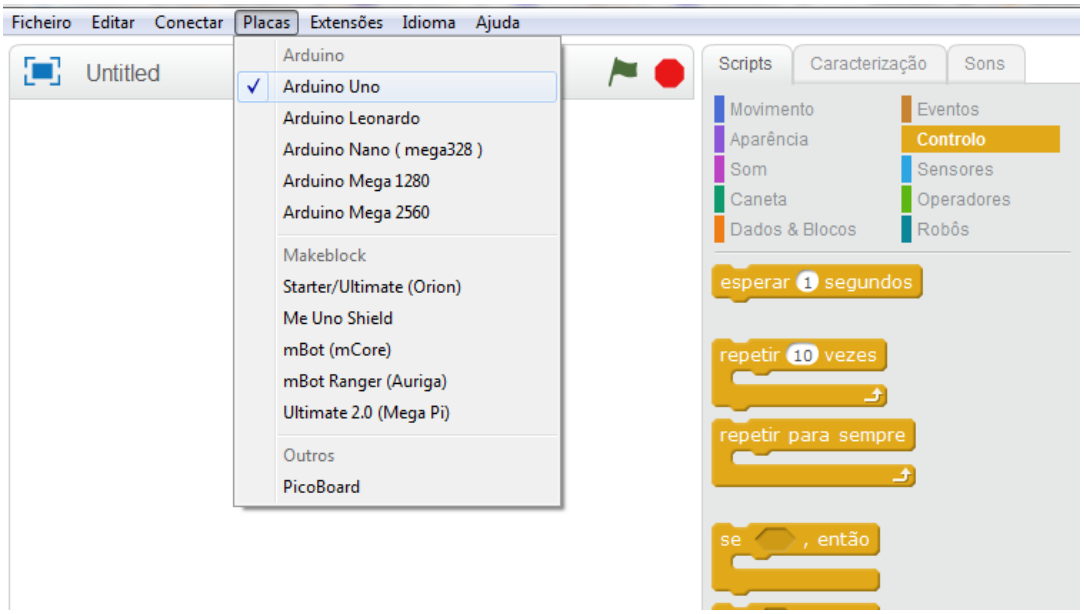
Em seguida, execute o arquivo salvo (.exe) para instalar o MBlock.
Atenção! Estamos usando aqui a versão 3 do Mblock (v3.4.11, especificamente). A versão 5 ainda está em desenvolvimento e não permite controlar placas Arduino.

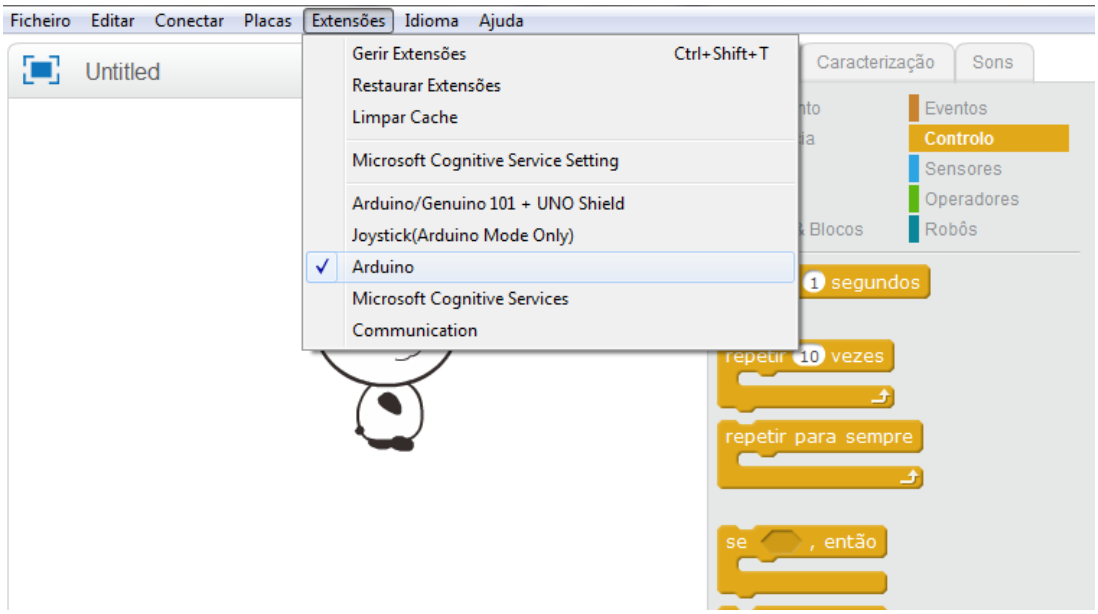
Começando

Após instalar e abrir o MBlock, você pode alterar o idioma para português, clicando na barra superior em Language > Português.
Em seguida, após conectar sua placa Arduino ao computador, selecione a porta à qual ela está conectada (no nosso caso, por exemplo, COM2), clicando na barra superior em Conectar > Porta Serial (COM)



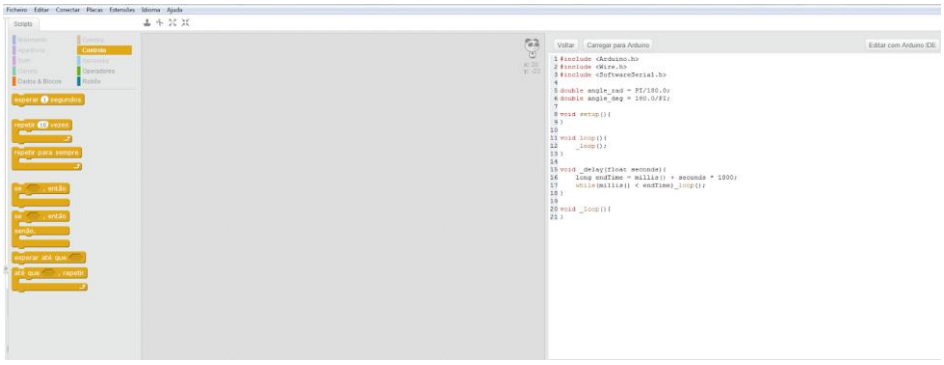
Por fim, selecione o seu tipo de placa em Placas e Extensões:





Em Editar, clique em Modo Arduino:

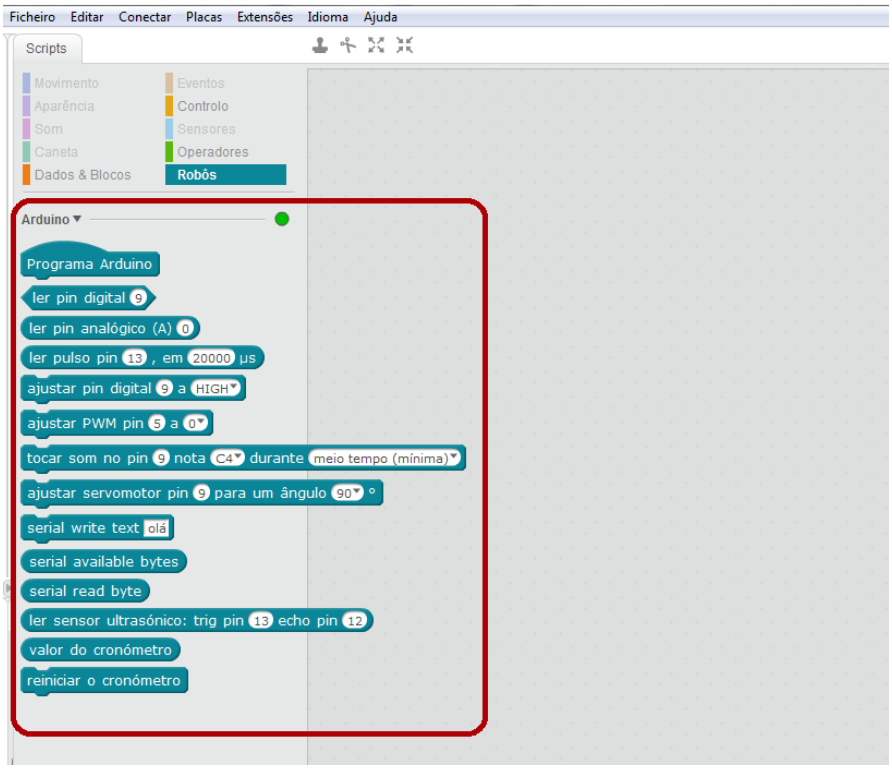
Você perceberá que um novo quadro aparecerá à direita da sua tela, que deve ficar mais ou menos assim:



Este quadro que aparece no canto direito fará a "tradução" da linguagem de blocos para a linguagem da IDE da Arduino. Assim, após criar o seu programa com blocos similares aos do Scratch, você pode observar à direita a tradução dela para a linguagem entendida pela Arduino, o que pode ser uma ótima forma de aprender.

Programando

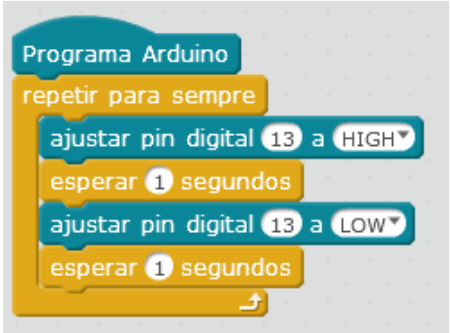
Para começar, podemos recriar o programa Blink, que faz o LED 13 da Arduino piscar em intervalos de 1 segundo.
Na categoria "Robôs", você encontrará os blocos abaixo:



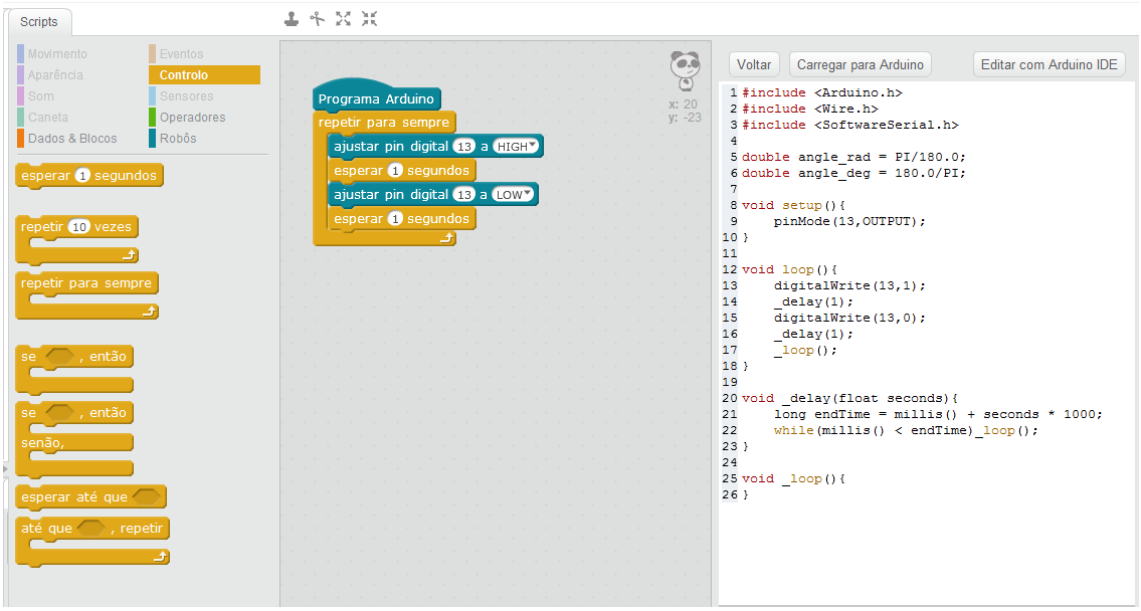
Para o nosso programa que pisca o LED, utilizaremos a seguinte lógica:

- ligar o LED (conectado ao pino 13)
- aguardar 1 segundo
- desligar o LED (conectado ao pino 13)
- aguardar um segundo

E colocaremos tudo isso dentro de um loop, ou seja, dentro de um bloco "repetir para sempre", que fará com que esses comandos se repitam indefinidamente.
Com os blocos disponíveis, criamos então a seguinte programação (note que os programas para Arduino sempre devem começar com o bloco "Programa Arduino":



Observe que, no quadro à direita, teremos agora um novo trecho de código:



Nele, a parte que corresponde aos blocos que utilizamos encontra-se no seguinte trecho:

```
void loop(){
  digitalWrite(13,1);
  _delay(1);
  digitalWrite(13,0);
  _delay(1);
  _loop();
}
```

No trecho acima, digitalWrite (13,1) corresponde ao comando para ligar o pino 13 e digitalWrite (13,0) ao comando para desligá-lo. O _delay(1), por sua vez, corresponde a uma espera de 1 segundo. E, tudo que está dentro do loop ficará se repetindo indefinidamente, como no bloco "repetir para sempre".

Mas atenção! Se você for programar diretamente na IDE da Arduino, o delay será dado sempre em milissegundos, e por isso, para aguardar 1 segundo, você deve usar delay(1000) - o MBlock faz essa conversão de forma automática e "oculta" para facilitar a sua compreensão.

Passando o código para a placa Arduino

Agora que já criamos o programa, podemos carregá-lo para a placa Arduino. Para isso, clique no botão "Carregar para Arduino" e observe o que ocorre.

Se quiser, você também pode abrir o programa na IDE para modificá-lo. Para isso, clique em "Editar com Arduino IDE".

Finalizando

O MBlock pode ser uma ferramenta interessante para explorar a programação de Arduino por meio de uma linguagem mais intuitiva. No entanto, você observará que ela tem suas limitações. Por isso, sugerimos que se você estiver com dificuldade de compreender os programas iniciais deste curso, vá para o programa MBlock e explore por lá algumas possibilidades, fazendo testes para entender a função dos blocos e como eles se relacionam à programação da IDE. Mas, quando os conteúdos ficarem mais avançados, sugerimos que passe a trabalhar diretamente com a IDE para se familiarizar com ela e aproveitar todo o potencial da sua Arduino.
Por fim, ressaltamos que neste curso o nosso foco será a programação via IDE. O MBlock é sugerido como uma ferramenta auxiliar de transição, mas não ofereceremos tutoriais ou suporte ao seu uso ao longo do curso.

Estrutura geral do código Arduino

Para entender a estrutura geral dos códigos de Arduino, vamos analisar o exemplo "Blink", que faz o LED piscar em intervalos de 1 segundo:

```
void setup() {
  pinMode(13, OUTPUT);
}
void loop() {
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

O programa começa com a função setup(), na qual são definidas as configurações iniciais do programa. Tudo que estiver dentro das chaves { } fará parte da função.
Em seguida, temos o comando pinMode(13, OUTPUT); que configura o pino 13 como saída (em inglês, output). Usamos este comando pois o pino 13 (ao contrário dos pinos A0-A5) pode ser utilizado tanto como saída quanto como entrada. Assim, precisamos especificar que ele deve ser tratado como saída, já que um LED está conectado a ele.

pinMode é uma função, e as informações que estão dentro dos parênteses são seus argumentos - ou seja, informações necessárias para que ela execute sua tarefa. Neste caso, temos dois argumentos: (número do pino, tipo de sinal: OUTPUT OU INPUT).
Em seguida, temos a função loop(), que será repetida indefinidamente enquanto a placa estiver ligada. É aqui que especificamos o comportamento geral do programa.
Na função loop temos os seguintes comandos:

- digitalWrite(13, HIGH): o primeiro argumento (13) especifica o pino ao qual conectamos o LED, e o segundo argumento (HIGH ou 1) informa que este será configurado com a tensão de 5V - ligando o LED.
- digitalWrite(13, LOW): o segundo argumento (LOW ou 0) configura a tensão do pino 13 como 0 - desligando o LED.
- delay(1000): comando de espera em milissegundos. Neste caso, o Arduino aguarda 1 segundo antes de passar ao próximo comando.

Se estivéssemos programando a Arduino em uma linguagem de programação em blocos como o Scratch, teríamos no loop algo similar ao bloco de comandos abaixo:

Nas próximas aulas trataremos em mais detalhes desta estrutura, das funções e de seus parâmetros.

O conceito atrás do Blink

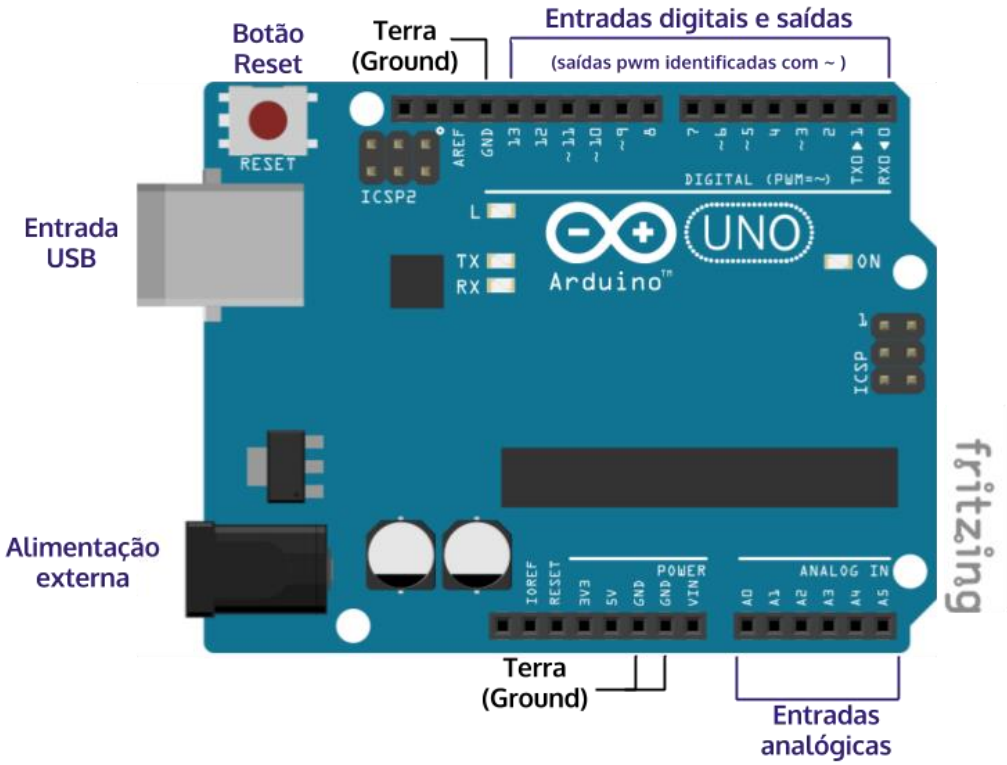
https://youtu.be/--RTiYNxr5Y

Apresentação da Arduino

https://youtu.be/zni672kZv5g

Principais partes de uma placa Arduino

A imagem abaixo apresenta a identificação das principais partes de uma placa Arduino, conforme apresentado no vídeo.



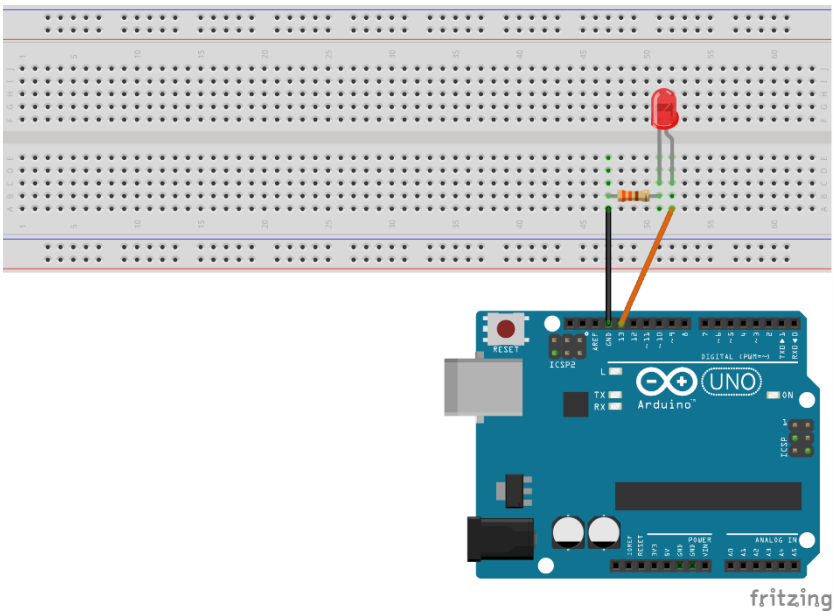
Conectando a Arduino a um outro circuito

https://youtu.be/HCKHoqGEz_c

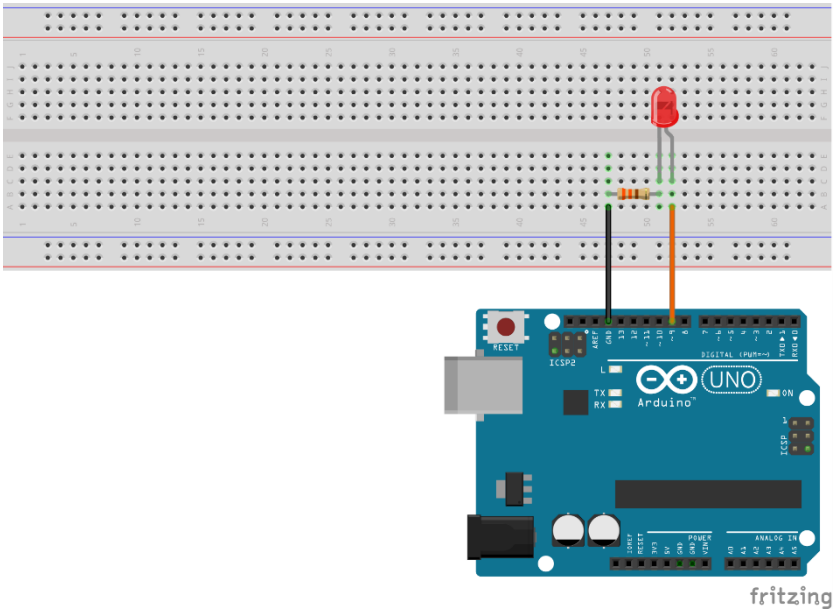
Como explorar exemplos da IDE Arduino

O programa apresentado no vídeo que faz o LED piscar é um dos exemplos básicos que podem ser obtidos diretamente da IDE do Arduino. Para encontrá-lo, vá até Arquivo > Exemplos > 01.Basics > Blink (ou File > Examples > 01.Basics > Blink).

Tente você também: monte o circuito abaixo, abra o exemplo Blink e faça o upload para sua Arduino.



Assim como o exemplo “Blink”, que faz o LED piscar, há diversos outros exemplos disponíveis na IDE que poderão ser úteis ao longo do curso.
Para experimentar um outro exemplo, conecte o LED ao pino 9 da sua placa, conforme esquema abaixo:



Faça o upload para a placa do exemplo "Fading" (disponível em Arquivo > Exemplos > 03. Analog > Fading), e observe o que acontece.

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 01/09/2020.
    - Licença disponível em https://creativecommons.org/licenses/by-nc-nd/4.0/deed.pt_BR - Acessado em 01/09/2020.
    - Plano de aula disponível em https://codeiot.org.br/assets/courseware/v1/b1730072a11dfbe6e66db2834c6bd652/asset-v1:LSI-TEC+IOT104+2018_S2+type@asset+block/Plano_de_Aula_Progamacao_Fisica_Semana_1.pdf - Acessado em 01/09/2020.