UFAC FUNDAPE CITS

Aplicações IoT
Internet das Coisas

Prof. André Nasserala
andre.nasserala@ufac.br

Apresentação

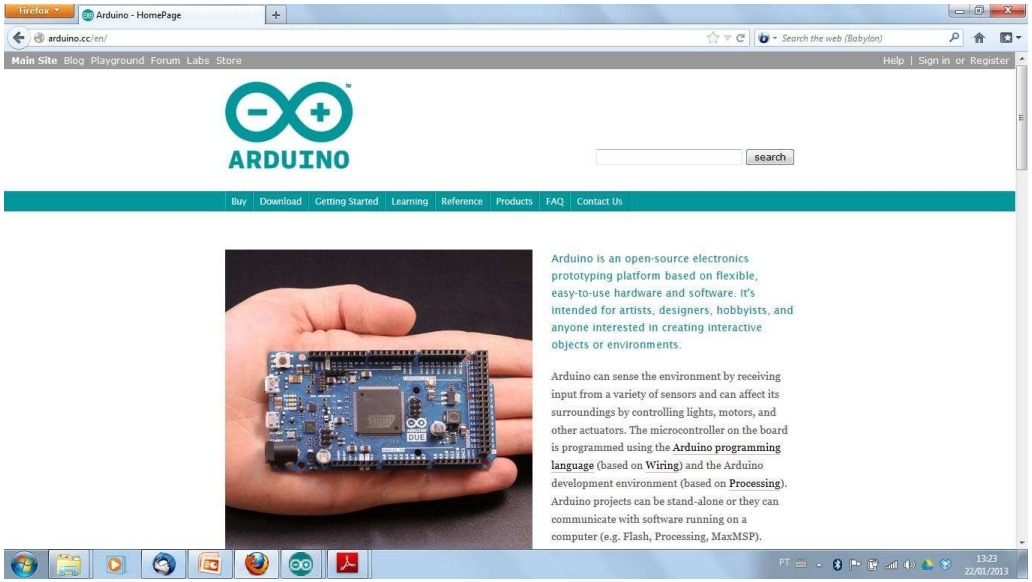
- O Arduino é uma plataforma utilizada para prototipação de circuitos eletrônicos.
- O projeto do Arduino teve início em 2005 na cidade de Ivrea, Itália.
- O Arduino é composto por uma placa com microcontrolador Atmel AVR e um ambiente de programação baseado em C++.
- Tanto o hardware como o ambiente de programação do Arduino são livres, ou seja, qualquer pessoa pode modificá-los e reproduzi-los.
- O Arduino também é conhecido como plataforma de computação física.

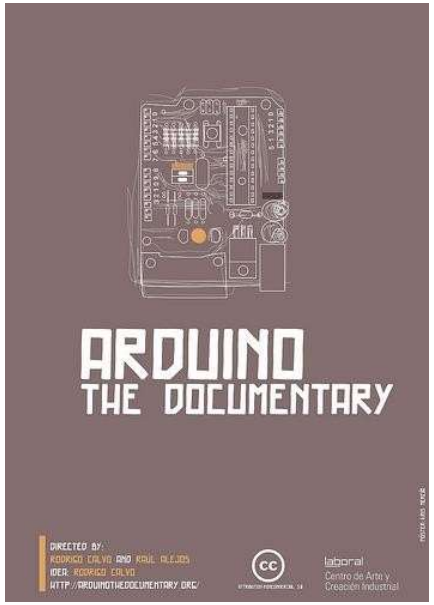
Apresentação

- Tipos de Arduino
- Existem vários tipos de Arduino com especificidades de hardware. O site oficial do Arduino lista os seguintes tipos:
  - Arduino UNO
  - Arduino Leonardo
  - Arduino Due
  - Arduino Esplora
  - Arduino Mega
  - Arduino Ethernet
  - Arduino Mini
  - Arduino LilyPad
  - Arduino Micro
  - Arduino Nano
  - Arduino Pro
  - Arduino Fio

Apresentação

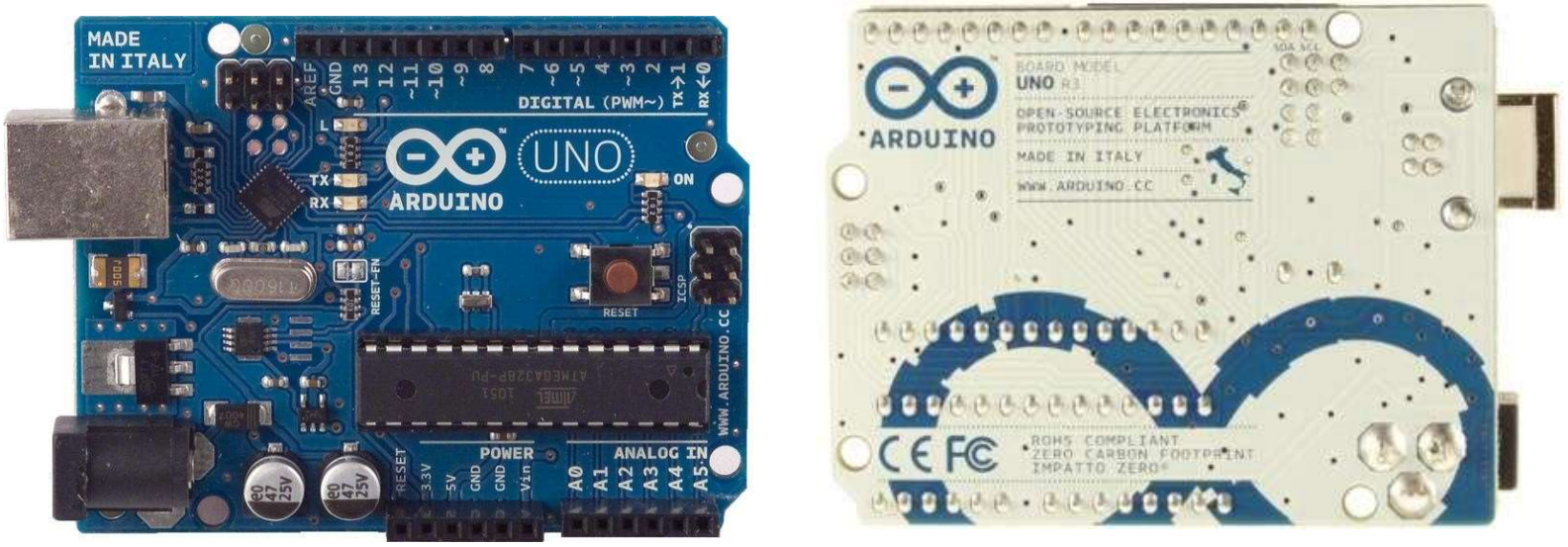
- Referências na WEB
- O site oficial do Arduino é http://arduino.cc
  - Um documentário sobre o Arduino pode ser assistido em: http://arduinothedocumentary.org/





Arduino UNO

- Vista da placa do Arduino UNO Rev 3 (frente e verso)



Arduino UNO

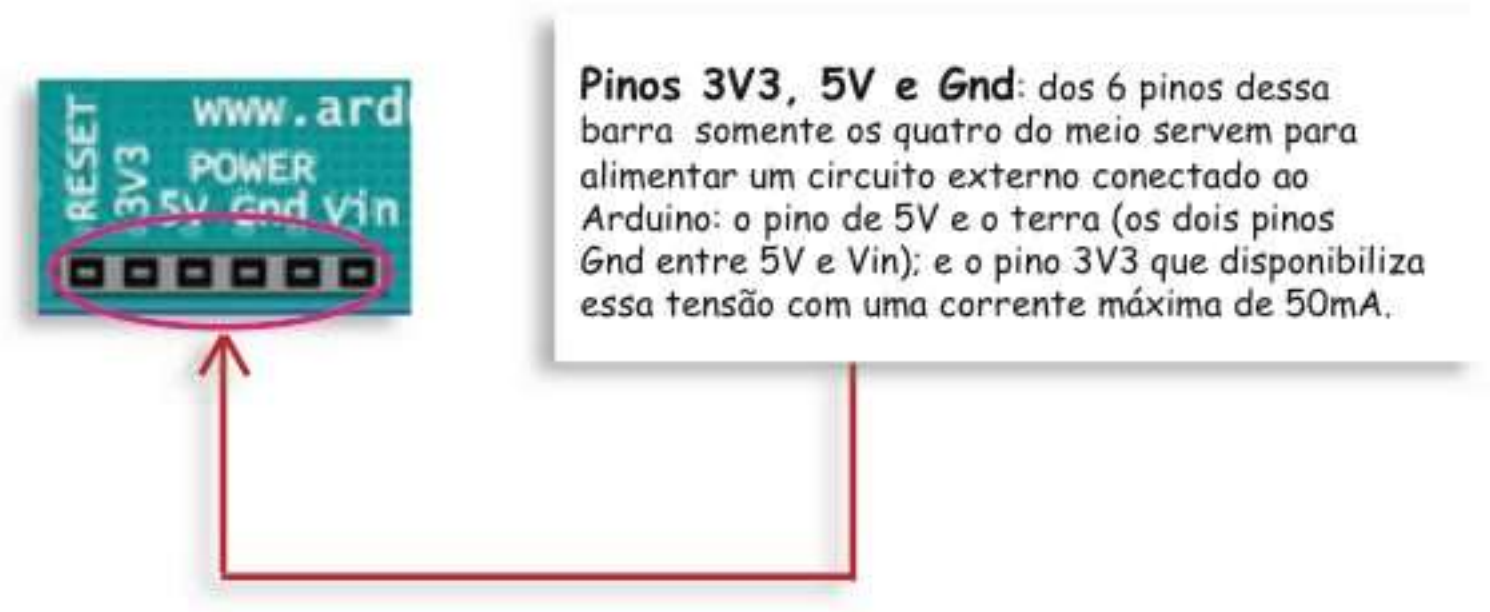
- Características
  - Microcontrolador: ATmega328
  - Tensão de operação: 5V
  - Tensão recomendada (entrada): 7-12V
  - Limite da tensão de entrada: 6-20V
  - Pinos digitais: 14 (seis pinos com saída PWM)
  - Entrada analógica: 6 pinos
  - Corrente contínua por pino de entrada e saída: 40 mA
  - Corrente para o pino de 3.3 V: 50 mA
  - Quantidade de memória FLASH: 32 KB (ATmega328) onde 0.5 KB usado para o bootloader
  - Quantidade de memória SRAM: 2 KB (ATmega328)
  - Quantidade de memória EEPROM: 1 KB (ATmega328)
  - Velocidade de clock: 16 MHz

Arduino UNO

- Alimentação
  - O Arduino UNO pode ser alimentado pela porta USB ou por uma fonte externa DC.
  - A recomendação é que a fonte externa seja de 7 V a 12 V e pode ser ligada diretamente no conector de fonte ou nos pinos Vin e Gnd.

Arduino UNO

- Pinos



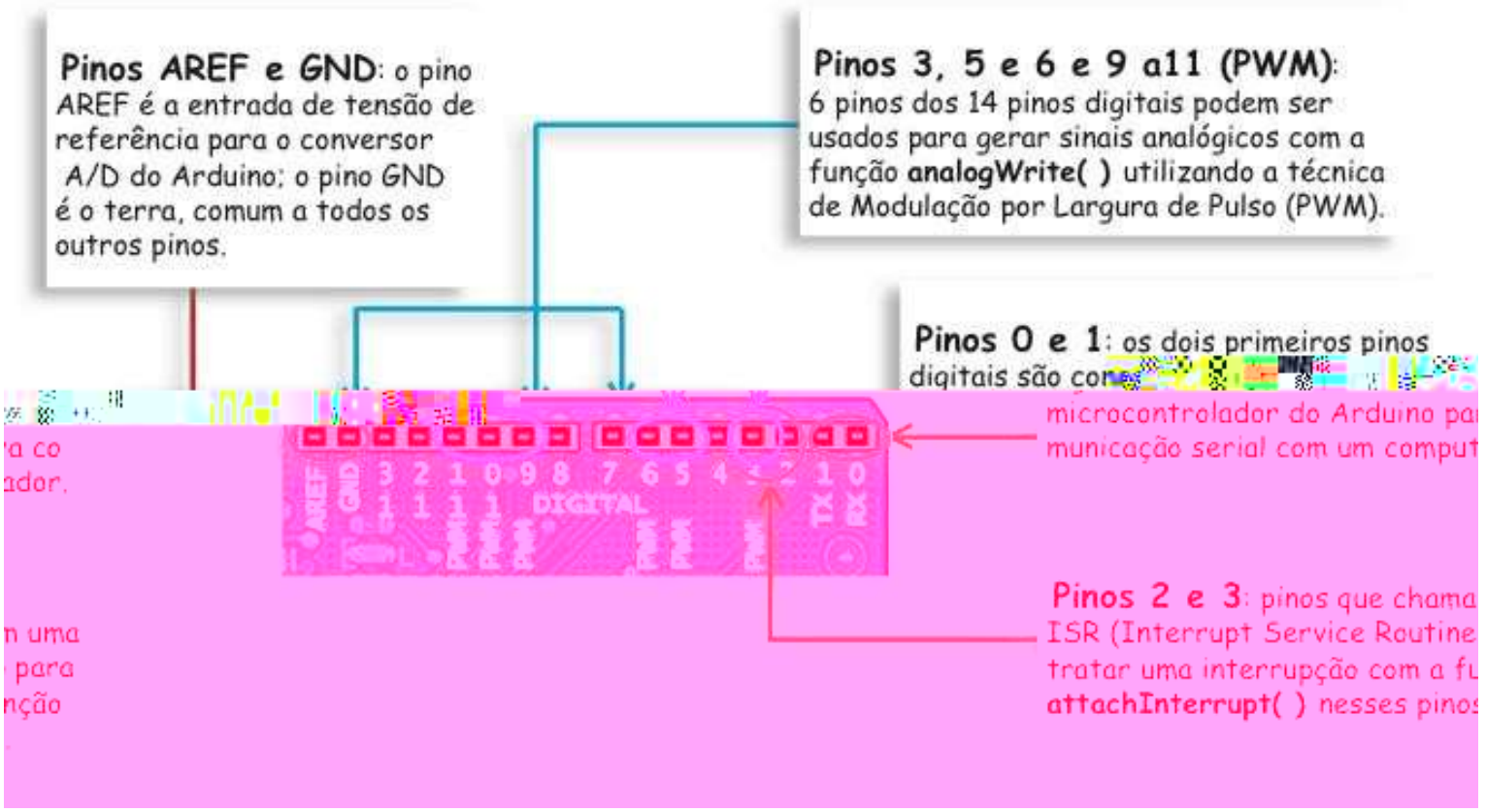
Arduino UNO

- Pinos

Pinos 0 a 5: esses 6 pinos aceitam tensões entre zero e 5 volts CC que vão ao conversor A/D de 10 bits no microcon trolador do Arduino. O pino AREF, numa das barras de pinos digitais, é a entrada de tensão de referência para esse conversor.

Arduino UNO

- Pinos



Ambiente de Desenvolvimento

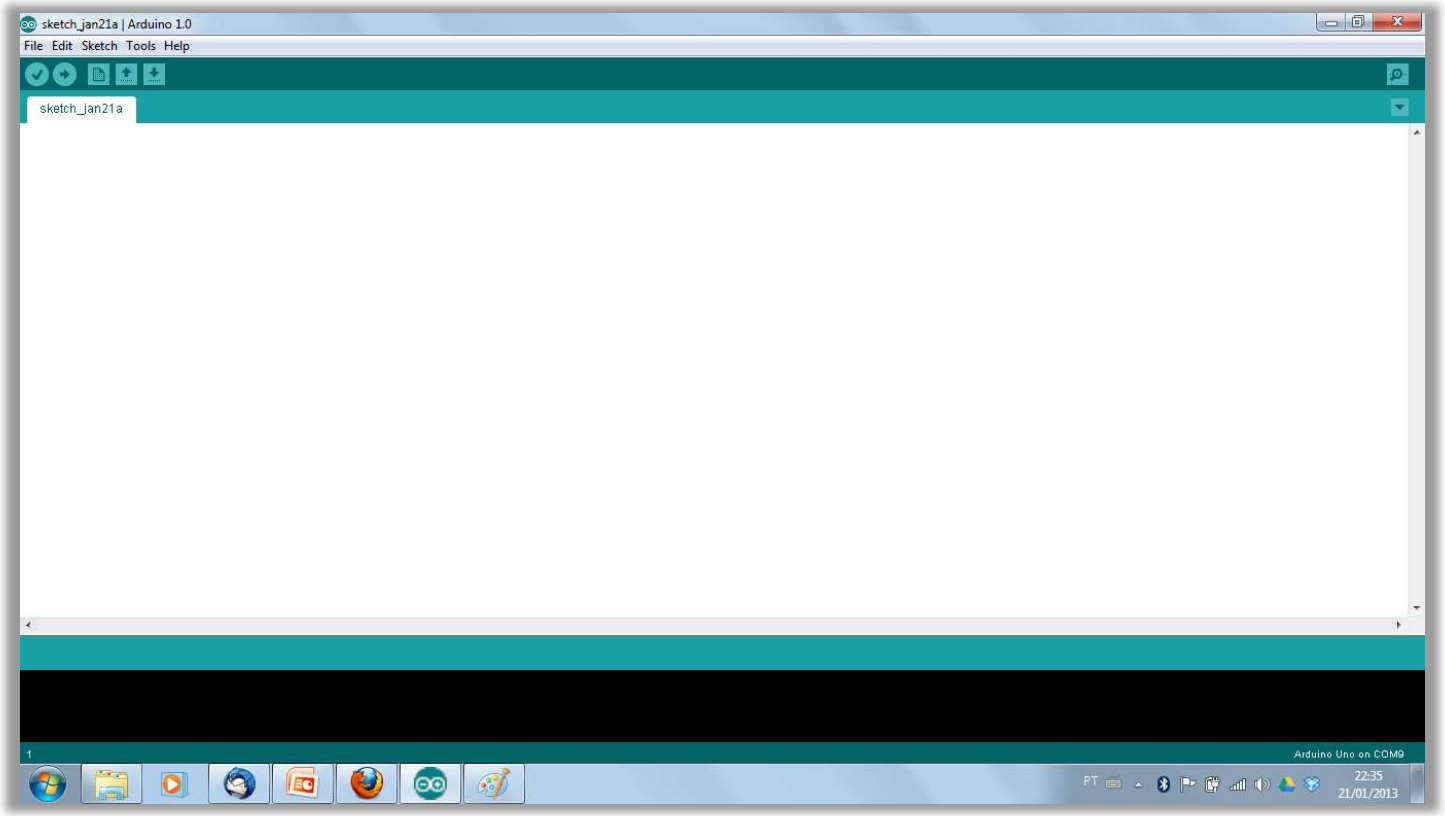
- O ambiente de desenvolvimento do Arduino (IDE) é gratuito e pode ser baixado no seguinte endereço: arduino.cc.

- As principais funcionalidades do IDE do Arduino são:

- Escrever o código do programa;
- Salvar o código do programa;
- Compilar um programa;
- Transportar o código compilado para a placa do Arduino.

Ambiente de Desenvolvimento

- Interface principal do ambiente de desenvolvimento



Funções setup() e loop()

- Primeiro programa: Blink LED

blink_led

```
void setup()
{
  pinMode(13, OUTPUT);
}

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

Funções setup() e loop()

- As duas principais partes (funções) de um programa desenvolvido para o Arduino são:

- **setup():** onde devem ser definidas algumas configurações iniciais do programa. Executa uma única vez.
- **loop()**: função principal do programa. Fica executando indefinidamente.

- Todo programa para o Arduino deve ter estas duas funções.

Funções setup() e loop()

- Formato das funções setup() e loop()

setuploop

```
void setup()
{

}

void loop()
{

}
```

Constantes

- A criação de constantes no Arduino pode ser feita de duas maneiras:
- Usando a palavra reservada const
- Exemplo:
- const int x = 100;

- Usando a palavra reservada define
- Exemplo:
- #define X 100;

Constantes

- No Arduino existem algumas constantes previamente definidas e são consideradas palavras reservadas.
- As constantes definidas são:
- **true** – indica valor lógico verdadeiro.
- **false** – indica valor lógico falso.
- **HIGH** – indica que uma porta está ativada, ou seja, está em 5V.
- **LOW** – indica que uma porta está desativada, ou seja, está em 0V.
- **INPUT** – indica que uma porta será de entrada de dados.
- **OUTPUT** – indica que uma porta será de saída de dados.

Comentários

- Muitas vezes é importante comentar alguma parte do código do programa.
- Existem duas maneiras de adicionar comentários a um programa em Arduino.

- A primeira é usando //, como no exemplo abaixo:
- // Este é um comentário de linha

- A segunda é usando /* */, como no exemplo abaixo:
- /* Este é um comentário de bloco. Permite acrescentar comentários com mais de uma linha */

Comentários

- Primeiro programa comentado

comentarios

```
/*************************************************
*       OFICINA DE ROBÓTICA - LARM - UFSC        *
*                                                *
* Blink Led: Primeiro programa em Arduino.       *
*            Pisca um led conectado à porta 13.  *
*************************************************/

// função usada para configurações iniciais
void setup()
{
  pinMode(13, OUTPUT);
}

// principal função do programa - laço infinito
void loop()
{
  digitalWrite(13, HIGH);
  delay(1000);
  digitalWrite(13, LOW);
  delay(1000);
}
```

Portas Digitais

- O Arduino possui tanto portas digitais quanto portas analógicas.

- As portas servem para comunicação entre o Arduino e dispositivos externos, por exemplo: ler um botão, acender um led ou uma lâmpada.

- Conforme já mencionado, o Arduino UNO, possui 14 portas digitais e 6 portas analógicas (que também podem ser utilizadas como portas digitais).

Portas Digitais

- As portas digitais trabalham com valores bem definidos. No caso do Arduino esses valores são 0V e 5V.
- 0V indica a ausência de um sinal e 5V indica a presença de um sinal.

- Para escrever em uma porta digital basta utilizar a função **digitalWrite(pino, estado).**

- Para ler um valor em uma porta digital basta utilizar a função **digitalRead(pino).**

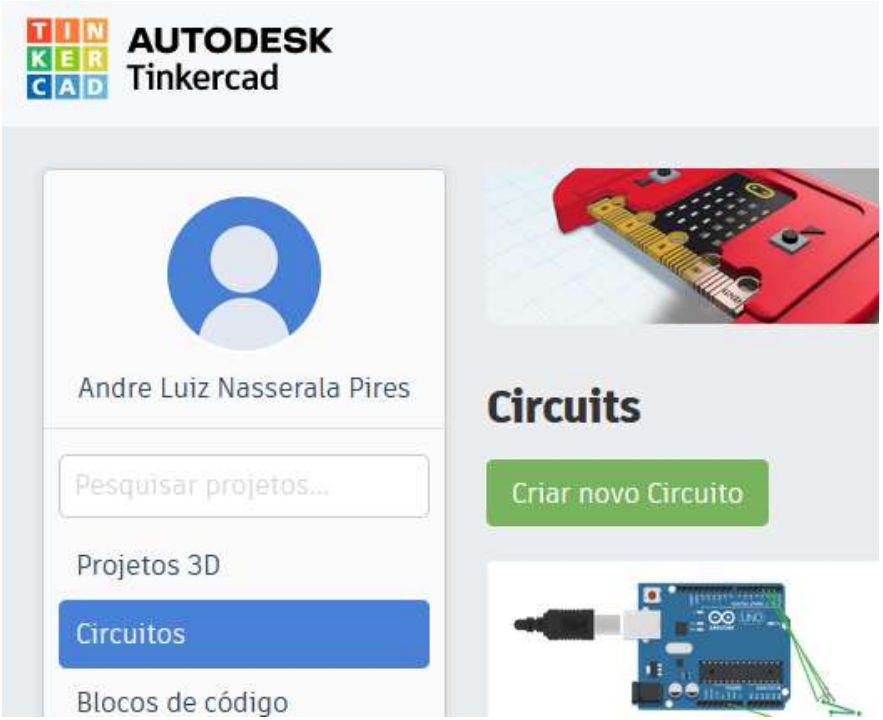
Portas Digitais

- As portas digitais são usadas para entrada e saída de dados.
- Para definir se uma porta será usada para entrada ou para saída de dados, é necessário explicitar essa situação no programa.
- A função pinMode(pino, estado) é utilizada para definir se a porta será de entrada ou saída de dados.
- **Exemplos**:
- Define que a porta 13 será de saída
- pinMode(13, OUTPUT)
- Define que a porta 7 será de entrada
- pinMode(7, INPUT)

Primeiro Programa

- Usaremos o simulador:
- https://www.tinkercad.com/

- Entre e façam uma conta de aluno.



Primeiro Programa

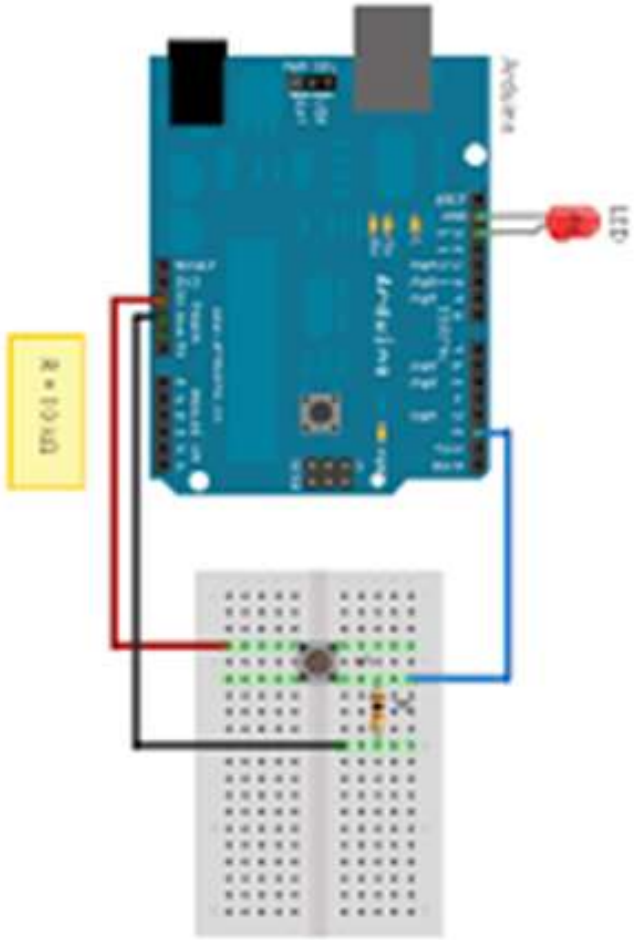
```
void setup()
{
   pinMode(2, OUTPUT);
}

void loop()
{
   digitalWrite(2, HIGH);
   delay(1000);
   digitalWrite(2, LOW);
   delay(1000);
}
```

Push-Button

- Os push-buttons (chaves botão) e leds são elementos presentes em praticamente qualquer circuito eletrônico.
- As chaves são usadas para enviar comandos para o Arduino e os Leds são elementos de sinalização luminosa.
- Esses dois componentes são trabalhados por meio das entradas e saídas digitais do Arduino.

Push-Button

- Neste exemplo vamos fazer uma aplicação básica que você provavelmente vai repetir muitas vezes.
- Vamos ler o estado de um push-button e usá-la para acender ou apagar um led.
- Ou seja, sempre que o botão for acionado, vamos apagar ou acender o Led.



Push-Button

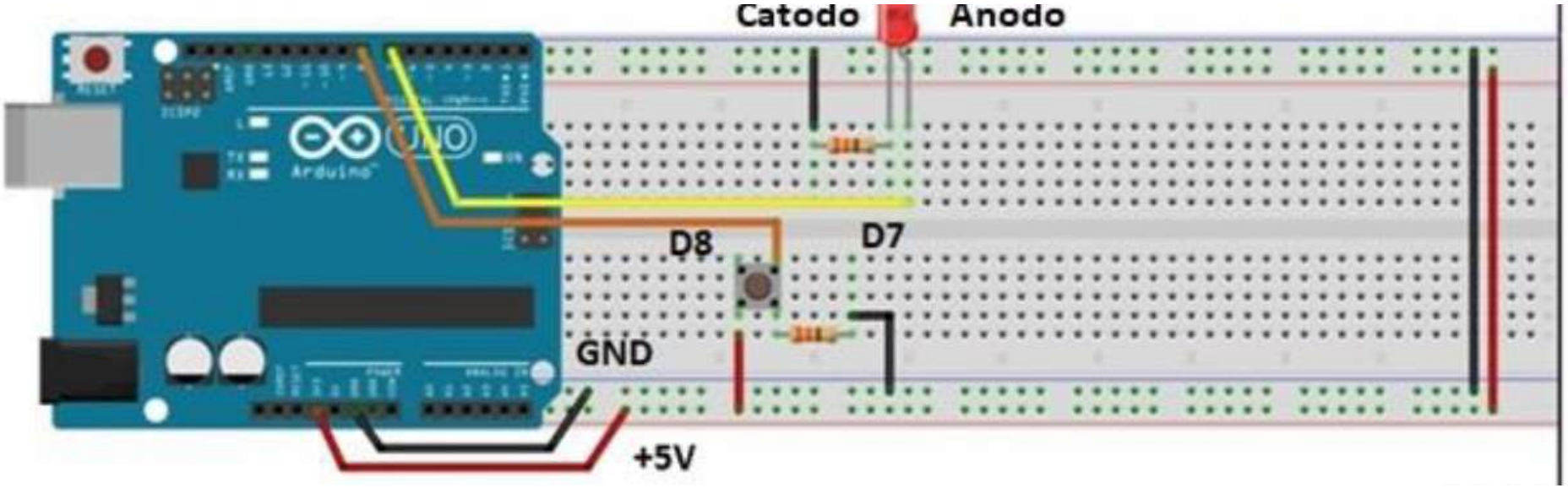
- Lista de materiais:

- Para esse exemplo você vai precisar:

- 2 resistores de 330 ohms;
- 1 Led vermelho (ou de outra cor de sua preferência);
- Push-button (chave botão);
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

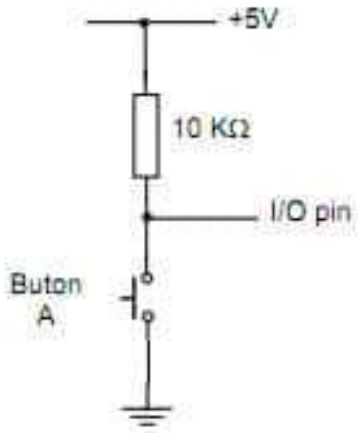
Push-Button

- Diagrama de circuito:

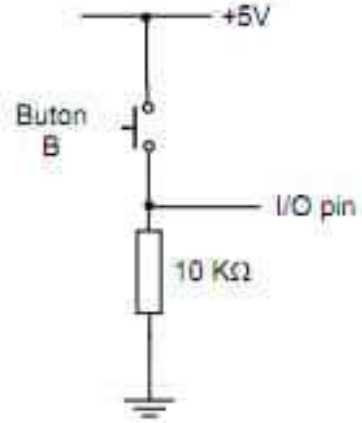


Push-Button

- Os resistores pull-up e pull-down garantem um nível lógico estável quando por exemplo uma tecla não está pressionada.
- Geralmente utiliza-se um resistor de 10KΩ para esse propósito.
- A seguir é exibida a ligação desses resistores no circuito para leitura de tecla do botão:



Pull-Up



Pull-Down

Push-Button

- Esse pequeno código abaixo mostra como ler entradas digitais e como acionar as saídas digitais.
- Na função void Setup(), é preciso configurar qual pino será usado como saída e qual será usado como entrada.
- Depois de configurar os pinos, para acioná-los basta chamar a função digitalWrite(pino,HIGH).
- A função digitalWrite() aciona ou desaciona um pino digital dependendo do valor passado no argumento:

- Se for “HIGH”, o pino é acionado.
- Se for “LOW”, o pino é desligado.

Push-Button

- Na função void Loop(), fizemos um if no qual a função digitalRead é usada para saber se o pushButton está acionado ou não.

- Caso ele esteja acionado, nós acendemos o Led, caso ele esteja desligado, nós desligamos o led.

- Carregue o código abaixo e pressione o botão para acender o LED.

Push-Button

```
#define PinButton 8 // define pino digital D8
#define ledPin 7 // define pino digital D7
void setup()
{
  pinMode(PinButton, INPUT); // configura D8 como entrada digital
  pinMode(ledPin, OUTPUT); // configura D7 como saída digital
}
void loop()
{
  if ( digitalRead(PinButton) == HIGH)  { // se chave = nivel alto
    digitalWrite(ledPin, HIGH); // liga LED com 5V
  }
  else { // senão chave = nivel baixo
   digitalWrite(ledPin, LOW); // desliga LED com 0V
  }
  delay(100); // atraso de 100 milissegundos
}
```

Melhorando o Código

- Criar um circuito para atribuir duas funções em um único botão (push button).
- Neste circuito o botão servirá como um interruptor para ligar e desligar um componente eletrônico.
- No nosso exemplo vamos ligar e desligar um led utilizando o push button como interruptor.
- **Observação**: Os push-button apenas mudam seu estado enquanto estamos pressionando, voltando ao seu estado original quando o botão é liberado.
- Teremos uma rotina para atribuir a um só botão duas funções de ligar e desligar um componente eletrônico qualquer.

Melhorando o Código

- Criar um circuito para atribuir duas funções em um único botão (push button).

- Neste circuito o botão servirá como um interruptor para ligar e desligar um componente eletrônico.

- No nosso exemplo vamos ligar e desligar um led utilizando o push button como interruptor.

Melhorando o Código

- **<u>Como o projeto deve funcionar:</u>**

- Quando você rodar o programa, o led ficará apagado.

- Ao pressionar e soltar o botão o led se acenderá.

- Ao pressionar e soltar novamente o botão o led se apagará, invertendo-se assim o seu estado.

Melhorando o Código

```
#define PinButton 8 // define pino digital D8
#define ledPin 7 // define pino digital D7
int estado = 0; // variável para leitura do pushbutton
int guarda_estado = LOW; // variável para armazenar valores do pushbutton
void setup()
{
  pinMode(PinButton, INPUT); // configura D8 como entrada digital
  pinMode(ledPin, OUTPUT); // configura D7 como saída digital
  }
void loop()
{
  estado = digitalRead(PinButton);  // le o estado pushbutton: ligado (HIGH) ou desligado (LOW)
  if (estado == HIGH) {// verifica se o botão (pushbutton) está pressionado
    guarda_estado = !guarda_estado;  // inverte valor da variável variable_buttonEstado
    delay(500); //esperera o tempo de 500ms para evitar que haja várias vezes alterações
  }
  if (guarda_estado == HIGH) {
    digitalWrite(ledPin, HIGH); // liga o led
  }
  else {
      digitalWrite(ledPin, LOW); // desliga o led
  }
}
```

Sensor de luz LDR

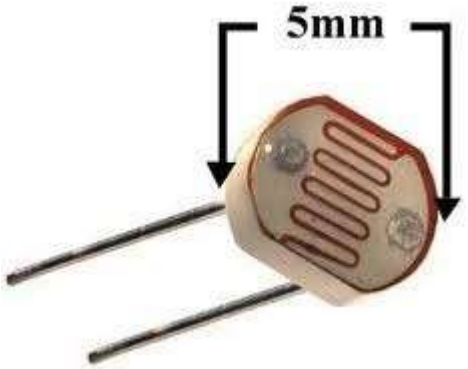
- O sensor LDR é um sensor de luminosidade.
- LDR é um Light Dependent Resistor, ou seja, um resistor cuja resistência varia com a quantidade de luz que incide sobre ele.
- Esse é seu princípio de funcionamento.

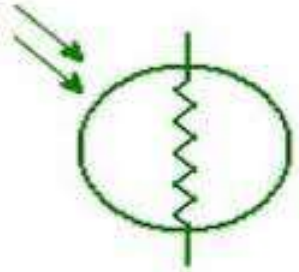
Sensor de luz LDR

- É importante considerar a potência máxima do sensor, que é de 100 mW.
- Ou seja, com uma tensão de operação de 5V, a corrente máxima que pode passar por ele é 20 mA.
- Felizmente, com 8K ohms (medido experimentalmente com o ambiente bem iluminado), que é a resistência mínima, a corrente ainda está longe disso, sendo 0,625mA.
  - Nas suas medições, pode ser que você encontre um valor de resistência mínimo diferente, pois depende da iluminação local.
- Dessa forma, podemos interfacear o sensor diretamente com o Arduino.

Especificações do LDR

- Modelo: GL5528
- Diâmetro: 5mm
- Tensão máxima: 150VDC
- Potência máxima: 100mW
- Temperatura de operação: -30°C a 70°C
- Comprimento com terminais: 32mm
- Resistência no escuro: 1 MΩ (Lux 0)
- Resistência na luz: 10-20 KΩ (Lux 10)





- Este sensor de luminosidade pode ser utilizado em projetos com arduino e outros microcontroladores para alarmes, automação residencial, sensores de presença e vários outros.

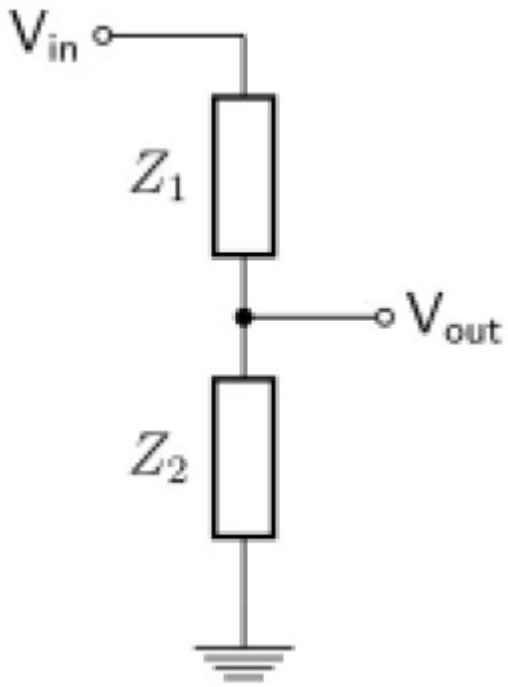
Divisor de Tensão

- O divisor de tensão consiste em dois resistores ligados em série (Z1 e Z2) , em que o sinal de 5V é aplicado a o terminal de um deles.
- O terminal do segundo resistor é ligado ao GND, e o ponto de conexão entre os dois resistores é a saída do divisor, cuja tensão é dada pela seguinte relação:

$$V_{\text{out}} = \frac{Z_2}{Z_1 + Z_2} \cdot V_{\text{in}}$$

Divisor de Tensão

- Em que Z1 e Z2 são os valores dos resistores da figura abaixo.



Divisor de Tensão

- Um divisor de tensão muito comum é fazer Z1 igual 330 ohms e Z2 igual 680 ohms.
- Vout = (680 /(330+680)*5
- Vout = 0,673 * 5
- Vout = 3,365 v
- Dessa forma a sáida Vout fica sendo 3.365 V.

- Você pode ligar um resistor de 330ohms como Z1 e dois de 330ohms como Z2.
- Os dois de 330 ligados em série formam um resistor de 660 ohm, o que resulta numa saída de 3.33V.

Exemplo

- No exemplo a seguir, vamos usar uma entrada analógica do Arduino para ler a variação de tensão no LDR e, consequentemente, saber como a luminosidade ambiente está se comportando.
- Veja na especificação que com muita luz, a resistência fica em torno de 10-20 KΩ, enquanto que no escuro pode chegar a 1MΩ.
- Para podermos ler as variações de tensão resultantes da variação da resistência do LDR, vamos usar o sensor como parte de um divisor de tensão.

Exemplo

- Assim, a saída do divisor será dependente apenas da resistência do sensor, pois a tensão de entrada e a outra resistência são valores conhecidos.

- No nosso caso, vamos usar um resistor de 10K e uma tensão de operação de 5V.

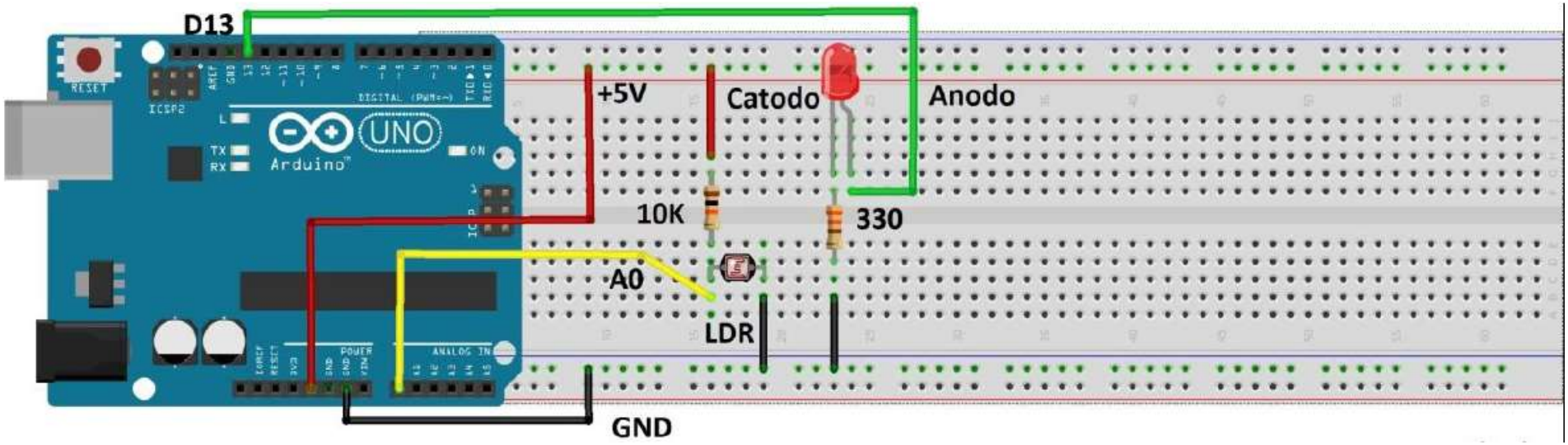
- Assim, o sinal que vamos ler no arduino terá uma variação de 2,2V (quando o LDR for 8K) e 5V (quando o LDR tiver resistências muito maiores que o resistor de 10K).

Exemplo

- Lista de materiais:

- Para esse exemplo você vai precisar:
- LDR;
- Resistor de 10k;
- 1 Arduino UNO;
- Protoboard;
- Jumpers de ligação;

Exemplo



Exemplo

- No diagrama, o sensor é ligado como parte de um divisor de tensão no pino analógico A0, de forma que a tensão de saída do divisor varia de acordo com a variação da resistência do sensor.

- Assim, vamos identificar as variações na intensidade de luz pelas variações na tensão do sensor.

- Quanto maior a intensidade de luz, menor a resistência do sensor e, consequentemente, menor a tensão de saída.

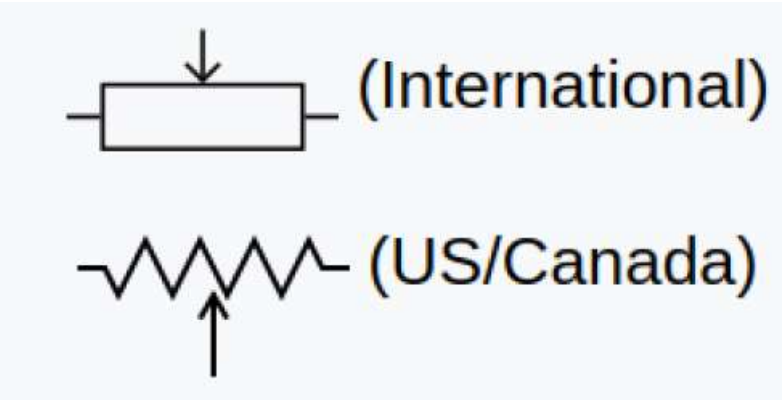
```
// Exemplo 4 - Sensor de luz LDR
// Apostila Eletrogate - KIT START

#define AnalogLDR A0                    // define pino analógico A0
#define Limiar 1.5                      // define constante igual a 1.5
#define ledPin 13                       // define pino digital D13

int Leitura = 0;                        // variavel inteiro igual a zero
float VoltageLDR;                       // variavel numero fracionario
float ResLDR;                           // variavel numero fracionario

void setup()
{
  pinMode(ledPin, OUTPUT);                  // configura D13 como saída digital
  Serial.begin(9600);                       // monitor serial - velocidade 9600 Bps
  delay(100);                               // atraso de 100 milisegundos
}

void loop()
{
  Leitura = analogRead(AnalogLDR);          // leitura da tensão no pino analogico
A0
  VoltageLDR =  Leitura * (5.0/1024);       // calculo da tensão no LDR
  Serial.print("Leitura sensor LDR = ");    // imprime no monitor serial
  Serial.println(VoltageLDR);               // imprime a tensão do LDR

  if (VoltageLDR > Limiar)                  // se a tensão LDR maior do que limiar
    digitalWrite(ledPin, HIGH);             // liga LED com 5V
  else                                      // senão a tensão LDR < limiar
    digitalWrite(ledPin, LOW);              // desliga LED com 0V
  delay(500);                               // atraso de 500 milisegundos
}
```

Potenciômetro

- O potenciômetro nada mais do que um resistor cujo valor de resistência pode ser ajustado de forma manual.
- Existem potenciômetros slides e giratórios.
- Na figura abaixo mostramos um potenciômetro giratório dos mais comumente encontrados no mercado.

Potenciômetro

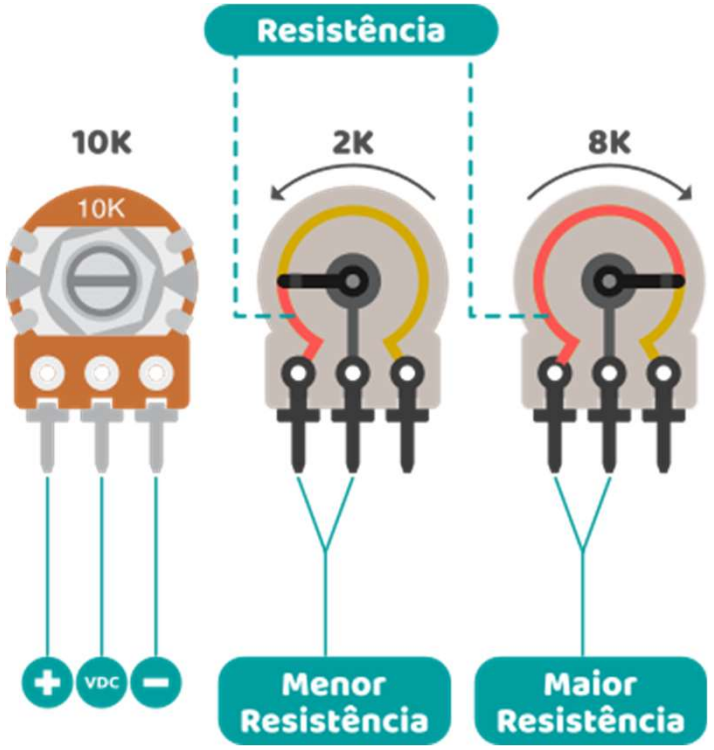
- Em ambos, ao variar a posição da chave manual, seja ela giratória ou slide, o valor da resistência entre o terminal de saída e um dos outros terminais se altera.



- O símbolo do potenciômetro é o mostrado na seguinte imagem:

Exemplo - Potenciômetro

- Nesse exemplo, vamos ligar a saída de um potenciômetro a uma entrada analógica da Arduino UNO.
- Dessa forma, vamos ler o valor de tensão na saída do potenciômetro e vê-la variando de 0 a 1023.
- Mas como assim, 0 a 1023?
- Isso se deve ao seguinte:



Exemplo - Potenciômetro

- Vamos aplicar uma tensão de 5V nos terminais do potenciômetro.

- A entrada analógica do Arduino, ao receber um sinal de tensão externo, faz a conversão do mesmo para um valor digital, que é representado por um número inteiro de 0 a 1023.

- Esses números são assim devido à quantidade de bits que o conversor analógico digital do Arduino trabalha, que são 10 bits (2 elevado a 10 = 1024).

Exemplo - Potenciômetro

- Ou seja, o arduino divide o valor de tensão de referência em 1024 unidades (0 a 1023) de 0,00488 volts.
- Assim, se a tensão lida na entrada analógica for de 2,5V, o valor capturado pelo arduino será metade de 2,5/0,00488 = 512. Se for 0V, será 0, e ser for 5V, será 1023, e assim proporcionalmente para todos os valores.
- Assim, digamos que o valor de tensão se dado por V.
- O valor que o Arduino vai te mostrar é:

- Valor = (V/5)*1024

Exemplo - Potenciômetro

- Em que 5V é o valor de referência configurado no conversor analógico-digital (uma configuração já padrão, não se preocupe com ela) e 1024 é igual 2 elevado a 10.
- No nosso código, queremos saber o valor de tensão na saída do potenciômetro, e não esse número de 0 a 1023, assim, rearranjar a equação para o seguinte:

- Tensão = Valor*(5/1024)

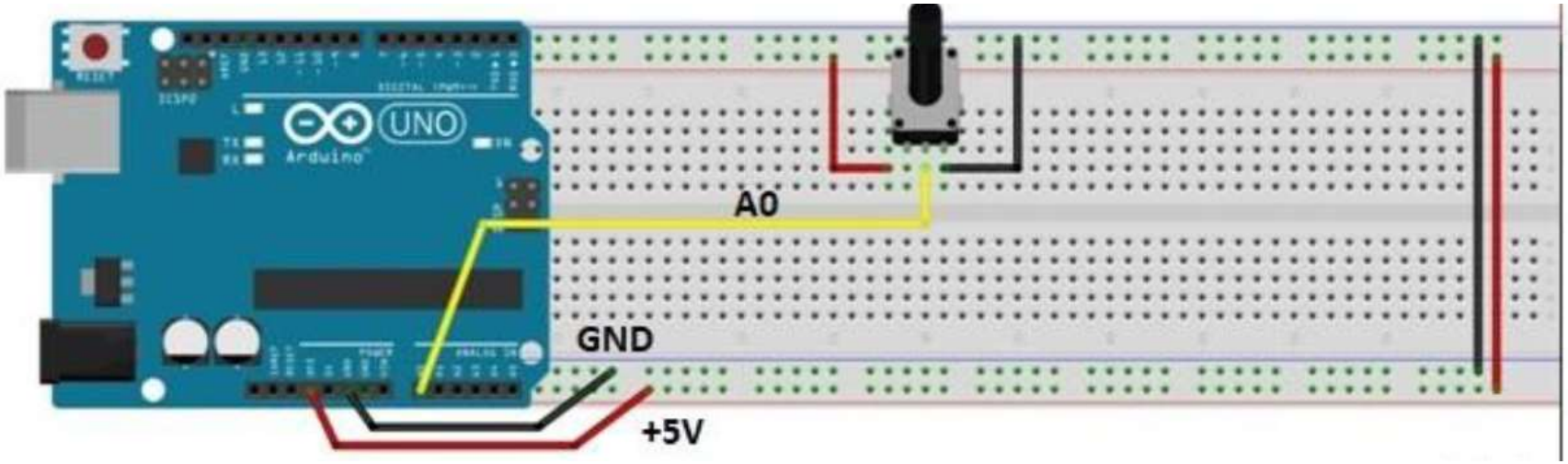
Exemplo - Materiais

- Lista de materiais:

- Para esse exemplo você vai precisar:
- Arduino UNO;
- Protoboard;
- Potenciômetro 10K;
- Jumpers de ligação;

Exemplo - Diagrama

- Diagrama de circuito
- Monte o circuito conforme diagrama abaixo e carregue o código de exemplo:



Exemplo - Diagrama

- O Potenciômetro possui 3 terminais, sendo que o do meio é o que possui resistência variável.
- A ligação consiste em ligar os dois terminais fixos a um tensão de 5V.
- Assim, o terminal intermediário do potenciômetro terá um valor que varia de 0 a 5V à medida que você dá gira seu knob.
- O terminal intermediário é ligado diretamente a uma entrada analógica do Arduino (A0).
- Como a tensão é de no máximo 5V, então não há problema em ligar direto.

Exemplo - Código

- Carregue o código abaixo no Arduino e você verá as leituras no Monitor serial da IDE arduino.

```
// Exemplo 1 - Usando potenciometro para fazer leituras analógicas
// Apostila Eletrogate - KIT START

#define sensorPin A0      // define entrada analógica A0

int sensorValue = 0;      // variável inteiro igual a zero
float voltage;            // variável numero fracionario

void setup()
{
  Serial.begin(9600);     // monitor serial - velocidade 9600 Bps
  delay(100);             // atraso de 100 milisegundos
}

void loop()
{
  sensorValue = analogRead(sensorPin);       // leitura da entrada analógica A0
  voltage = sensorValue * (5.0 / 1024);      // cálculo da tensão

  Serial.print("Tensão do potenciometro: "); // imprime no monitor serial
  Serial.print(voltage);                     // imprime a tensão
  Serial.print("    Valor: ");               // imprime no monitor serial
  Serial.println(sensorValue);               // imprime o valor
  delay(500);                                // atraso de 500 milisegundos
}
```

Exemplo - Código

```
// Exemplo 1 - Usando potenciometro para fazer leituras analógicas
// Apostila Eletrogate - KIT START

#define sensorPin A0      // define entrada analógica A0

int sensorValue = 0;      // variável inteiro igual a zero
float voltage;            // variável numero fracionario

void setup()
{
  Serial.begin(9600);     // monitor serial - velocidade 9600 Bps
  delay(100);             // atraso de 100 milisegundos
}

void loop()
{
  sensorValue = analogRead(sensorPin);        // leitura da entrada analógica A0
  voltage = sensorValue * (5.0 / 1024);       // cálculo da tensão

  Serial.print("Tensão do potenciometro: "); // imprime no monitor serial
  Serial.print(voltage);                     // imprime a tensão
  Serial.print("    Valor: ");               // imprime no monitor serial
  Serial.println(sensorValue);               // imprime o valor
  delay(500);                                // atraso de 500 milisegundos
}
```

Exemplo - Código

- No código acima, nós declaramos a variável sensorValue para armazenar as leituras da entrada analógica A0 e a variável do tipo float Voltage para armazenar o valor lido convertido para tensão.
- Na função void Setup(), nós inicializamos o terminal serial com uma taxa de transmissão de 9600 kbps.
- Na função void Loop(), primeiro faz-se a leitura da entrada analógica A0 com a função analogRead(SensorPin) e armazenamos a mesma na variável sensorValue.
- Em seguida, aplicamos a fórmula para converter a leitura (que é um número entre 0 e 1023) para o valor de tensão correspondente.
- O resultado é armazenado na variável Voltage e em seguido mostrado na interface serial da IDE Arduino.

Exemplo - Código

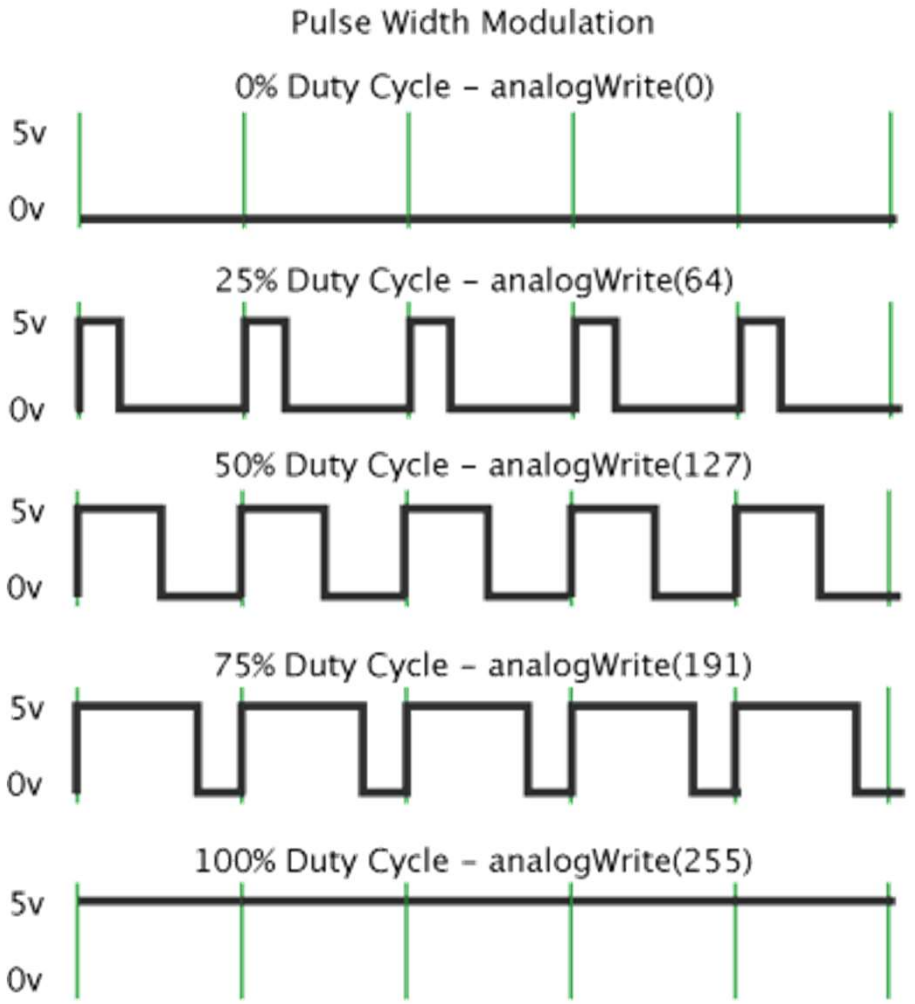
```
// Exemplo 1 - Usando potenciometro para fazer leituras analógicas
// Apostila Eletrogate - KIT START
#define sensorPin A0 // define entrada analógica A0
int sensorValue = 0; // variável inteiro igual a zero
float voltage; // variável numero fracionario
void setup()
{
  Serial.begin(9600); // monitor serial - velocidade 9600 Bps
  delay(100); // atraso de 100 milisegundos
}
void loop()
{
  sensorValue = analogRead(sensorPin); // leitura da entrada analógica A0
  voltage = sensorValue * (5.0 / 1024); // cálculo da tensão
  Serial.print("Tensão do potenciometro: "); // imprime no monitor serial
  Serial.print(voltage); // imprime a tensão
  Serial.print(" Valor: "); // imprime no monitor serial
  Serial.println(sensorValue); // imprime o valor
  delay(500); // atraso de 500 milisegundos
}
```

O que é PWM?

- A sigla PWM (**Pulse Width Modulation**) significa Modulação por Largura de Pulso.

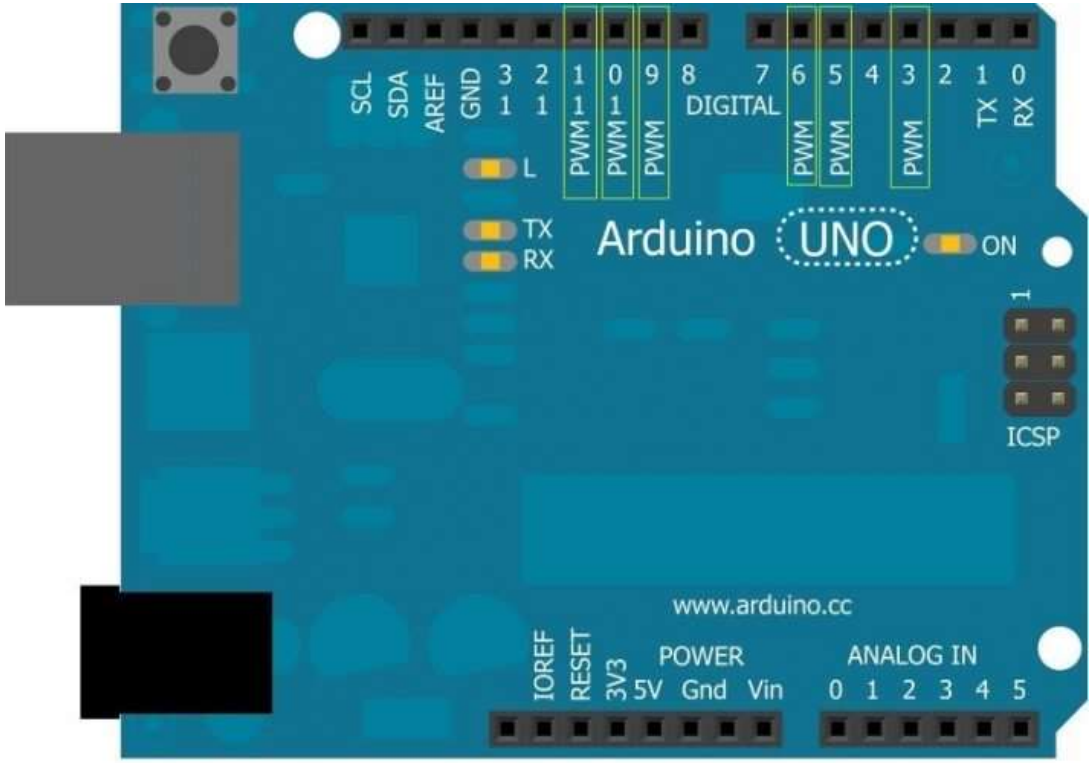
- O PWM funciona modulando o ciclo ativo (duty cicle) de uma onda quadrada.
- O conceito de funcionamento é simples.
- O controlador (fonte de tensão com PWM) entrega uma série de pulsos, gerados em intervalos de igual duração, que pode ser variada.
- Quanto mais largo o pulso, maior a quantidade de corrente fornecida à carga.

O que é PWM?


Pulse Width Modulation
0% Duty Cycle – analogWrite(0)
5v
0v
25% Duty Cycle – analogWrite(64)
5v
0v
50% Duty Cycle – analogWrite(127)
5v
0v
75% Duty Cycle – analogWrite(191)
5v
0v
100% Duty Cycle – analogWrite(255)
5v
0v

PWM no Arduino

- A placa Arduino Uno possui pinos específicos para saídas PWM e são indicados pelo carácter ‘~’ na frente de seu número, conforme exibido a seguir:



PWM no Arduino

- Observa-se na figura acima, que a Arduino Uno possui 6 pinos para saída PWM (3,5,6,9,10,11).

- Para auxiliar na manipulação desses pinos a plataforma possui uma função que auxilia na escrita de valores de *duty cycle* para esses pinos, assim você pode facilmente usar o PWM do Arduino UNO e outras placas.

Micro Servo - Servomotor

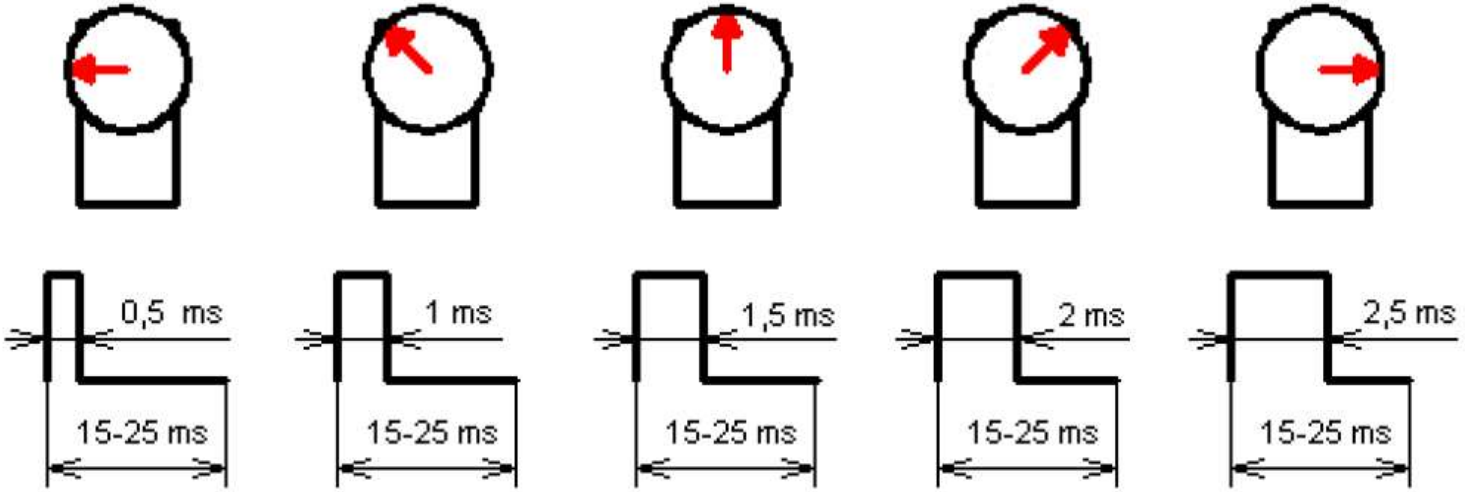
- Um servomotor é um equipamento eletromecânico que possui um encoder e um controlador acoplado.

- Diferentemente de motores tradicionais, como de corrente contínua, o servo motor apresenta movimento rotativo proporcional a um comando de forma a atualizar sua posição.

- Ao invés de girar continuamente como os motores de corrente contínua, o servo, ao receber um comando, gira até a posição especificada pelo mesmo.

Micro Servo - Servomotor

- Ou seja, o servo motor é um atuador rotativo para controle de posição, que atua com precisão e velocidade controlada em malha fechada.
- De acordo com a largura do pulso aplicado no pino de controle PWM, a posição do rotor é definida (0 a 180 graus) .
- Os pulsos devem variar entre 0,5 ms e 2,5 ms.



Micro Servo - Servomotor

- Existem dois tipos de Servomotores :
- Servomotor analógico
- Servomotor digital

- Servomotores analógicos são os mais comuns e mais baratos. O controle da posição do rotor utiliza um método analógico através da leitura de tensão sobre um potenciômetro interno.
- No caso dos servomotores digitais, mais caros e mais precisos, o controle da posição do rotor utiliza um encoder digital.

Micro Servo - Servomotor

- A figura abaixo mostra um típico servo motor analógico.

Micro Servo - Servomotor

- Os Servos são acionados por meio de três fios, dois para alimentação e um correspondente ao sinal de controle para determinar a posição.

- Em geral, os três fios são:
- Marron: GND,
- Vermelho: Alimentação positiva,
- Laranja: Sinal de controle PWM.

Micro Servo - Lista de materiais

- Antes de mais nada, vamos separar os componentes que precisamos para montar o servomotor junto com o arduino. A nossa lista de componentes é a seguinte:

- Micro Servo 9g SG90 TowerPro;
- Arduino UNO + cabo USB;
- Potenciômetro de 10k;
- Fonte 5V de protoboard para alimentar o servo;
- Jumpers para conexão no protoboard;
- Push button;

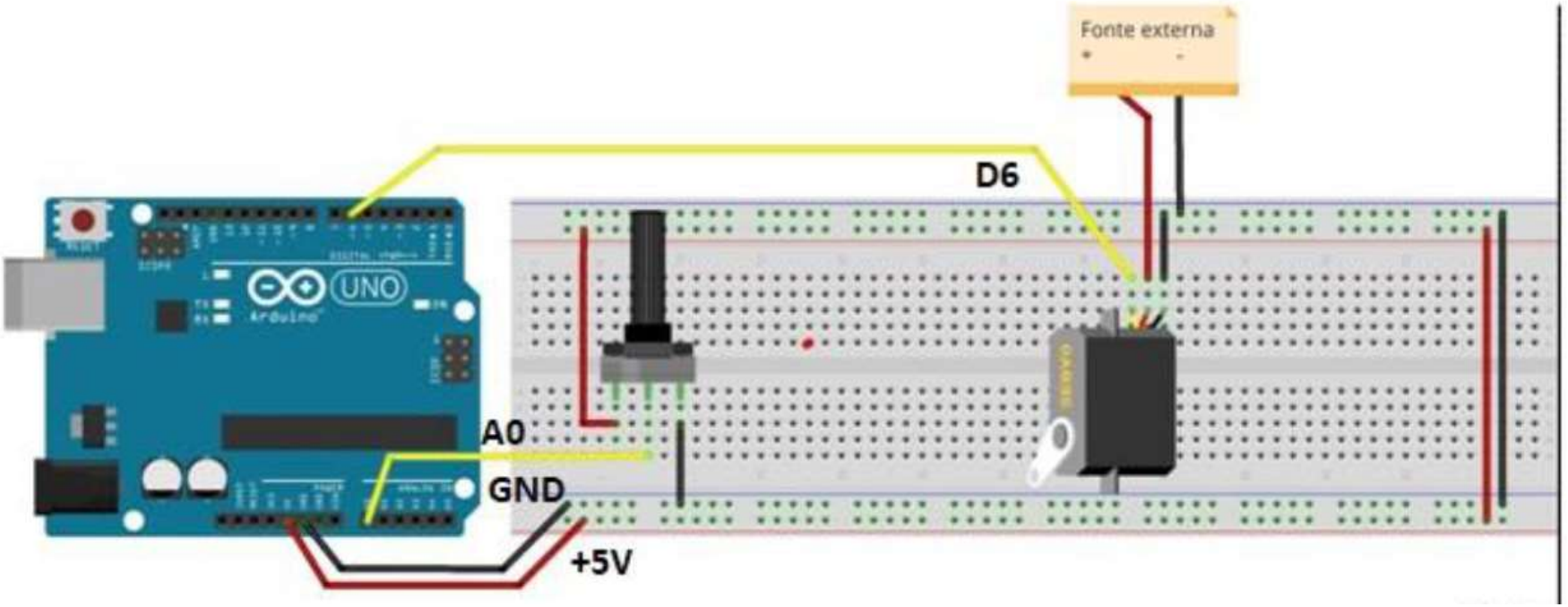
Micro Servo - Lista de materiais

- A montagem é simples:

- O servomotor em si deve ser ligado à alimentação conforme as cores apresentadas na introdução.
- De acordo com as especificações de tensão e corrente dos pinos do arduino UNO, os pinos de VCC e GND da placa conseguem fornecer até 200 mA.
- Ao utilizar servomotores, é recomendado que você utilize uma fonte externa, como a fonte para protoboard que inserimos na lista de materiais.

Micro Servo - Diagrama

- Diagrama de circuito
- A montagem para uma aplicação de controle do servo em qualquer posição, fica da seguinte forma:



Micro Servo - Código

- Carregue o código abaixo e gire o potenciômetro para o Servo motor girar:

```
// Exemplo 5 - Acionando o Micro Servo TowerPro
// Apostila Eletrogate - KIT START

#include <Servo.h>                      // usando bilbioteca Servo

Servo myservo;                          // cria o objeto myservo

#define potpin A0                       // define pino analógico A0

int val;                                // variavel inteiro

void setup()
{
  myservo.attach(6);                    // configura pino D6 - controle do Servo
}

void loop()
{
  val = analogRead(potpin);             // leitura da tensão no pino A0
  val = map(val, 0, 1023, 0, 179);      // converte a leitura em números (0-179)
  myservo.write(val);                   // controle PWM do servo
  delay(15);                            // atraso de 15 milisegundos
}
```

Micro Servo - Código

```
// Exemplo 5 - Acionando o Micro Servo TowerPro
// Apostila Eletrogate - KIT START

#include <Servo.h>                      // usando bilbioteca Servo

Servo myservo;                          // cria o objeto myservo

#define potpin A0                       // define pino analógico A0

int val;                                // variavel inteiro

void setup()
{
  myservo.attach(6);                    // configura pino D6 - controle do Servo
}

void loop()
{
  val = analogRead(potpin);             // leitura da tensão no pino A0
  val = map(val, 0, 1023, 0, 179);      // converte a leitura em números (0-179)
  myservo.write(val);                   // controle PWM do servo
  delay(15);                            // atraso de 15 milisegundos
}
```

Micro Servo - Código

- O aspecto mais importante desse software é a utilização da biblioteca servo.h.
- Esta biblioteca possui as funções necessárias para posicionar a servo para a posição desejada.
- Na função Void Setup() nós associamos o objeto servomotor, do tipo Servo, a um pino do arduino.
- Na mesma função, nós inicializamos o servo na posição 0º, utilizando o método write do objeto que acabamos de criar.
- Na função Void Loop(), o procedimento consiste em ler o valor do potenciômetro e usá-lo como referência para atualizar a posição do servo.

Micro Servo - Código

- A leitura analógica do potenciômetro retorna um valor entre 0 e 1023.
- Para controlar o servo nós usamos valores de 0 a 179, correspondendo ao meio giro de 180º do mesmo.
- Assim, é necessário usar a função Map() para traduzir a escala de 0-1023 para a escala de 0-179.
- Dessa forma, os valores lidos do potenciômetro podem ser usados como referência para determinar a posição do servomotor.

Micro Servo - Código

- Para atualizar a posição do servo a cada iteração do loop, nós chamamos o método write() do objeto servomotor, passando como parâmetro o valor lido do potenciômetro traduzido para a escala de 0-179.

- Assim, sempre que mexermos no potenciômetro, o servo motor irá atuar e atualizar a sua posição.

Micro Servo - Código

```
// Exemplo 5 - Acionando o Micro Servo TowerPro
// Apostila Eletrogate - KIT START
#include <Servo.h> // usando bilbioteca Servo
Servo myservo; // cria o objeto myservo
#define potpin A0 // define pino analógico A0
int val; // variavel inteiro
void setup()
{
  myservo.attach(6); // configura pino D6 - controle do Servo
}
void loop()
{
  val = analogRead(potpin); // leitura da tensão no pino A0
  val = map(val, 0, 1023, 0, 179); // converte a leitura em números (0-179)
  myservo.write(val); // controle PWM do servo
  delay(15); // atraso de 15 milisegundos
}
```

Sensor de Presença

- O Sensor de Movimento PIR que vamos usar é do modelo DYP-ME003, fabricado pela ElecFreaks.
- Esse sensor tem capacidade para detectar movimento de objetos dentro de um raio de até 7 metros, uma ótima distância para poder acender luzes e acionar outros dispositivos de automação no local.

Sensor de Presença

- O Sensor de Movimento PIR que vamos usar é do modelo DYP-ME003, fabricado pela ElecFreaks.
- Esse sensor tem capacidade para detectar movimento de objetos dentro de um raio de até 7 metros, uma ótima distância para poder acender luzes e acionar outros dispositivos de automação no local.
- O funcionamento do sensor é extremamente simples.
- Caso um objeto entre no raio de detecção, uma saída é ativada em nível alto, caso não haja nenhum movimento detectado dentro do range, a saída fica em nível baixo.

Sensor de Presença

- No verão o raio de detecção pode ser um pouco menor devido às temperaturas mais altas.
- O Datasheet fornece essa e algumas outras informações adicionais.
- Recomendamos fortemente que você conheça o datasheet do componente.

Aplicações

- O sensor PIR é ideal para projetos de automação residencial, comercial e de segurança.

- É muito usado em:

- Produtos de segurança;
- Dispositivos com detecção de presença;
- Iluminação automática;
- Acionamento automático de buzzers, lâmpadas e circuitos de automação residencial e comercial;
- Automação e controle industrial;

Descrição do Projeto

- No nosso projeto vamos usar o sensor PIR da ElecFreak para acionar um led.

- “Ler um sensor PIR (Passive Infrared) para detectar movimento dentro de um raio de até 7 metros e acionar um led caso seja detectado algum movimento. Após um certo tempo depois da última detecção, desligar o led”

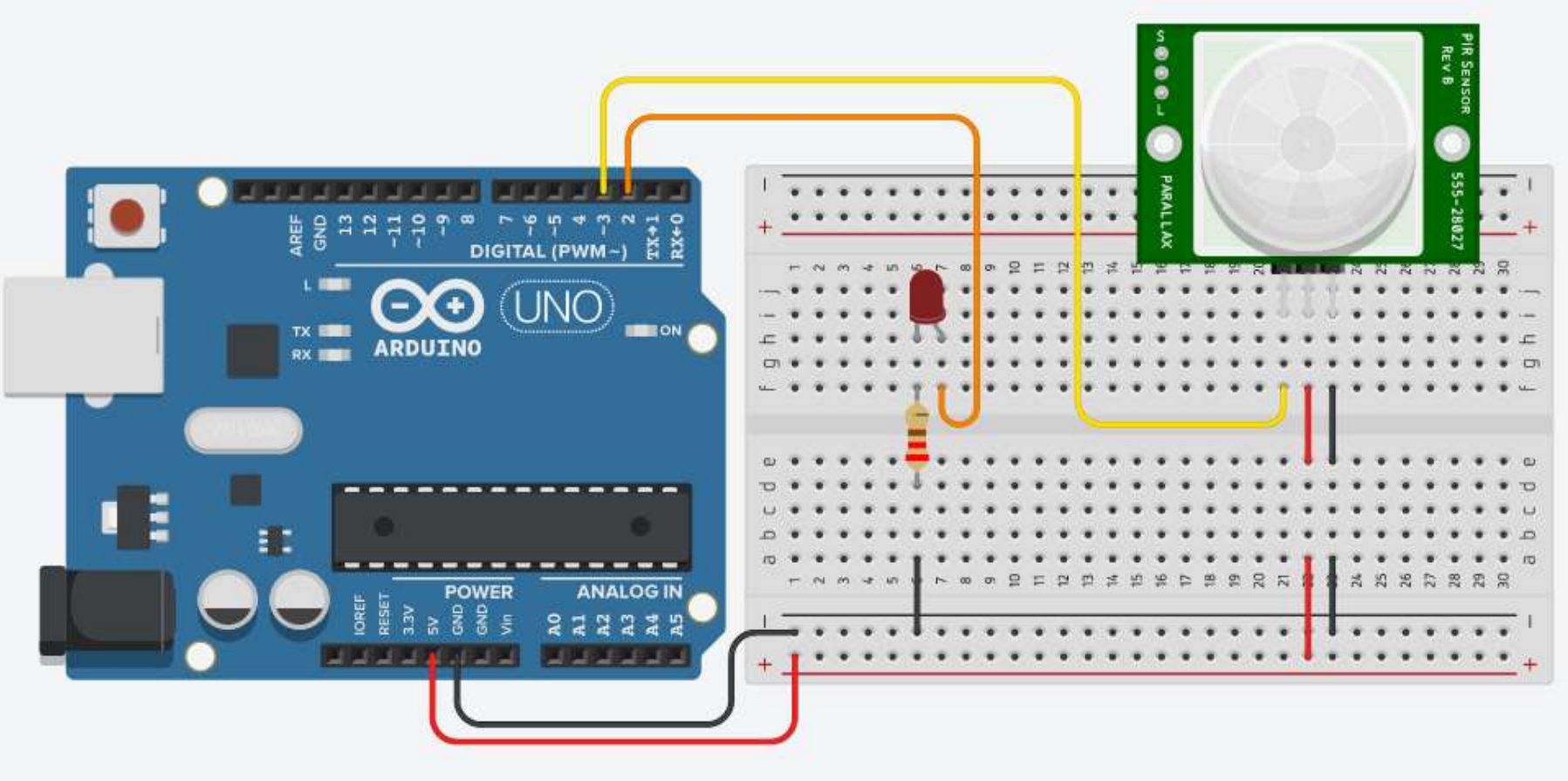
Aspectos de Hardware

- Materiais necessários para o projeto com Sensor de Presença PIR DYP-ME003 e Arduino Uno

- 1x Sensor de Movimento Presença PIR HC-SR501;
- 1x LED;
- 1x Resistor de 220 ohms;
- 1x Uno R3 + Cabo USB para Arduino;
- 1x Protoboard 400 Pontos;
- 8x Jumpers – Macho/Macho;

Aspectos de Hardware

- A montagem a seguir mostra como o circuito deve ser feito.



Aspectos de Software

- O software consiste no monitoramento constante do sensor PIR para saber se alguma variação de infravermelho(movimento de pessoas ou animações) foi identificado pelo sensor.

- Declaramos as variáveis para determinar os pinos de leitura(saída do sensor PIR) e de saída(aciona o Led), e na função void loop() fazemos a leitura do sinal de saído do sensor PIR, que é de natureza digital, isto é, ou é nível alto ou nível baixo.

Código

```
#define LED 2
#define SEN 3
int LeituraSensor;

void setup()
{
   pinMode(LED, OUTPUT);
   pinMode(SEN, INPUT);
   Serial.begin(9600);
}
```

Código

```
void loop()
{
  LeituraSensor = digitalRead(SEN); //Le o sensor
  if (LeituraSensor == LOW) //Nao há movimento
  {
    digitalWrite(LED, LOW);
  }
  else // há movimento
  {
    digitalWrite(LED, HIGH);
  }
  delay(2);
}
```

Bibliografia

- MONK, Simon. Programação com Arduino. Porto Alegre – RS. Editora: Bookman – 2017. ISBN: 9788582604465
- VIDAL, Vitor, Gustavo Murta. Arduino Start. Eletrogate – 2018. Belo Horizonte – MG. Disponível em: https://conteudo.eletrogate.com/apostila-arduino-start.
- MALVINO, Albert Paul. Eletrônica: Volume 1. 4.ed. São Paulo – SP: Makron Books, 1997. ISBN: 8534603782.
- SENAI, Senai SP. FUNDAMENTOS DE ELETRÔNICA - 1ªED. Editora: Senai SP – São Paulo 2015. ISBN: 9788583932086
- WILSON, J. A. e Milton Kaufman. Eletrônica Básica - Teoria e Prática - Volume 2. São Paulo: Editora: Rideel, 1980.
- PEREZ, Anderson Luiz Fernandes, Heron Pereira, Cristiano Pereira de Abreu, Renan Rocha Darós. Oficina de Robótica. UFSC – Programação Básica em Arduino - 2015. Disponível em: http://oficinaderobotica.ufsc.br/programacao-basica-em-arduino/.