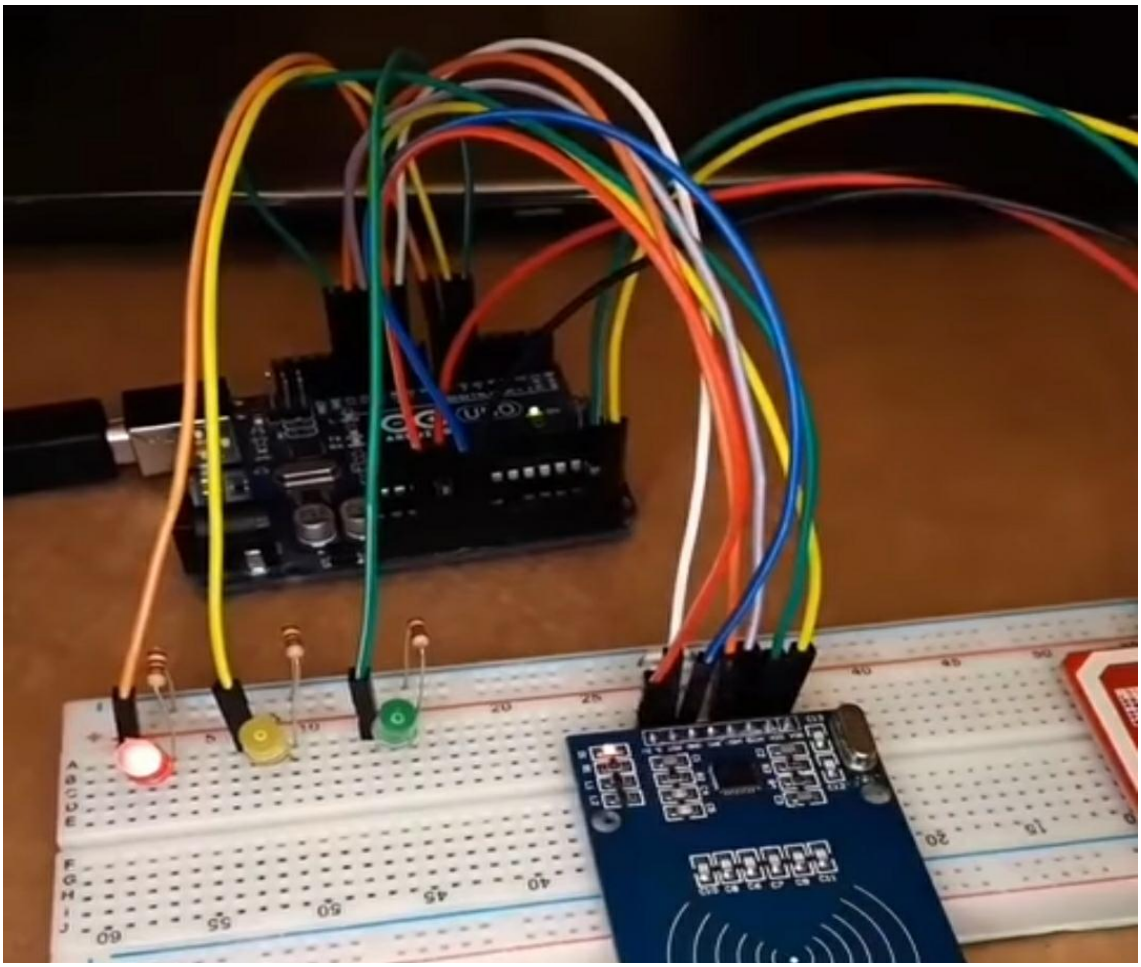
Marcio Vinicius

Introdução ao

Arduino



SUMÁRIO

1. A importância da Eletrônica

Atualmente a eletrônica possui uma imensa importância para as Indústrias. Quando assistimos uma programação na TV, ou quando ouvimos uma música na rádio, não percebemos como a eletrônica interfere em nossas vidas. Na década de 40, com o desenvolvimento da tecnologia de circuitos integrados, criou-se a base que deu origem aos computadores. Atualmente o avanço da Eletrônica tem tomado proporções assustadoras e isso se deve aos grandes avanços tecnológicos do setor.

A **eletrônica** é o ramo da ciência que estuda os circuitos formados por componentes elétricos e eletrônicos. O principal objetivo dessa tecnologia é representar, armazenar, transmitir e processar informações. Já no século 20, a eletrônica se separou das tecnologias voltadas para elétrica e eletromecânica.

Na prática, um técnico em eletrônica é capaz de trabalhar com aparelhos como televisores, rádios, computadores entre outros aparelhos domésticos e industriais. Sua capacidade vai muito além da manutenção, atuando na projeção e execução de novos aparelhos.

Enfim, a eletrônica adquiriu tanta importância que boa parte da nossa rotina está integrada a ela. Nossas necessidades mais básicas como alimentação, higiene e saúde estão permeadas de aparelhos eletrônicos além de muitos outros benefícios.

2.Primeira lei de Ohm

A primeira Lei de Ohm é de extrema importância para entendermos sobre a programação de Arduino, a primeira Lei de Ohm foi desenvolvida por Georg Simon Ohm, No dia

16 de março de 1787, nasceu em Erlangen, na Bavária (Alemanha), Georg Simon Ohm, físico e matemático que muito contribuiu com a física, principalmente para a eletrodinâmica, onde estabeleceu a lei batizada com seu nome.
Ohm iniciou sua carreira como professor de matemática no Colégio dos Jesuítas, na cidade de Colônia, em 1825. Sua intenção era de se tornar professor universitário, então continuou com seus trabalhos e pesquisas, dedicando-se à eletricidade.
Ohm realizou experiências com fios condutores de diferentes espessuras e comprimentos. Verificou que a resistência elétrica do condutor era inversamente proporcional à área da secção transversal do fio e diretamente proporcional ao seu comprimento. A partir de suas observações, definiu o conceito de resistência elétrica.

A primeira lei de Ohm determina que a diferença de potencial entre dois pontos de um resistor é proporcional à corrente elétrica que é estabelecida nele. Além disso, de acordo com essa lei, a razão entre o potencial elétrico e a corrente elétrica é sempre constante para resistores ôhmicos representada na formula abaixo

V=r.i

Aonde V é a tensão ou potencia elétrica, r é a resistência elétrica e i é a corrente elétrica. A potencia elétrica. Essa

grandeza é escalar e é medida em uma unidade Volts. A diferença de potencial Elétrico entre dois pontos de um circuito, por sua vez, indica que ali existe uma resistência elétrica, essa diferença decorre do consumo da energia dos elétrons, uma vez que essas partículas transferem parte de sua energia aos átomos da rede cristalina, quando conduzidas por meios que apresentam resistência à sua condução. A dissipação de energia pode ser explicada A partir de um efeito chamado efeito Joule.

A corrente elétrica **i** mede o fluxo de cargas pelo corpo em Ampères, ou em C/s. A corrente elétrica é diretamente proporcional à resistência elétrica dos corpos: quanto maior a resistência elétrica de um corpo, menor será a corrente elétrica a atravessá-lo.

Além da primeira lei de Ohm existe a sua segunda lei que relaciona as propriedades físicas que interferem na resistência elétrica de um corpo condutor e homogêneo. Essa lei informa que a resistência elétrica de um corpo é diretamente proporcional ao seu comprimento e resistividade é inversamente proporcional à sua área transversal.

Exercício

1)Um resistor de 100Ohm é percorrido por uma corrente elétrica de 20 mA. A ddp entre os terminais do resistor, em volts, é igual a:

A)5,0

B)2,0

C)2,0.10

2)Um resistor ôhmico, quando submetido a um ddp de 40V, é atravessado por uma corrente elétrica de intensidade 20 A. Quando a corrente que o atravessa for igual a 4 A, a ddp, em volts, nos seus terminais, será:

A)16

B)12

C)8

3)Ao ser estabelecida uma ddp de 30V entre os terminais de um resistor, estabelece-se uma corrente elétrica de 3A. Qual a resistência entre os terminais?

A)5

B)3

C)10

4)Os valores nominais de uma lâmpada incandescente, usada em uma lanterna, são: 6,0V; 20 mA.Isso significa que a resistência elétrica do seu filamento é de:

A)300Ohm, sempre, com a lâmpada acesa ou apagada.

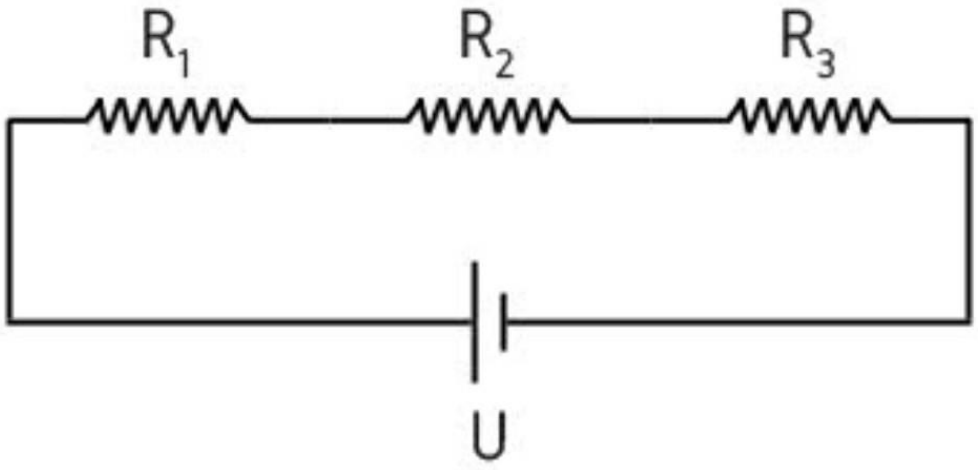
B)300Ohm com a lâmpada acesa e tem um valor bem maior quando apagada.

C)300Ohm com a lâmpada acesa e tem um valor bem menor quando apagada.

3. Associação de resistores

A associação de resistores é um circuito que apresenta dois ou mais resistores. Há três tipos de associação: em paralelo, em série e mista.

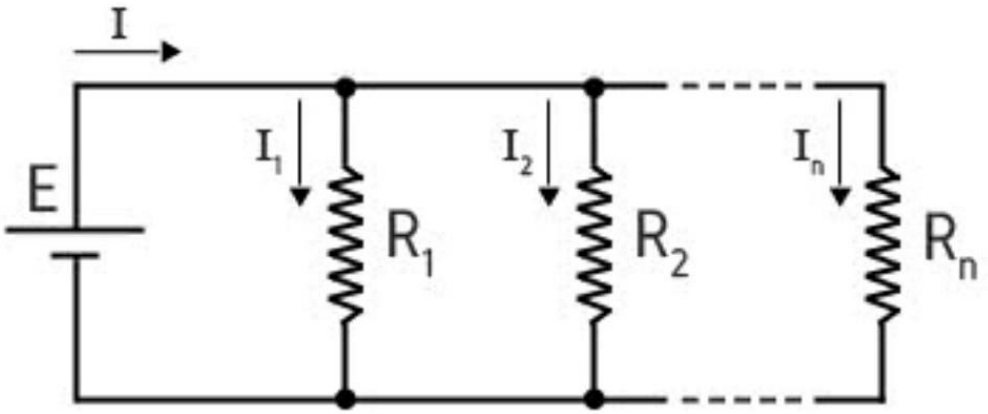
Na associação de resistores em série, os resistores são ligadas em sequência. Isso faz com que a corrente elétrica seja mantida ao longo do circuito, enquanto a tensão elétrica varia demonstrado na imagem abaixo



A resistência equivalente de um determinado circuito corresponde à soma das resistências de cada resistor que estiver presente no circuito

$$Req=R1+R2+R3+...+Rn$$

De forma resumida na ligação em série, as resistências somam-se, os potenciais elétricos somam-se é a corrente

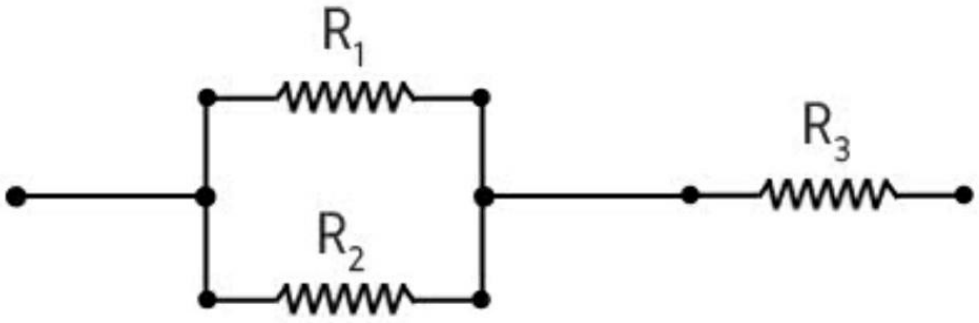


elétrica é igual para todos os resistores representado na imagem abaixo

Na associação de resistores em paralelo, os resistores encontram-se ligados ao mesmo potencial elétrico, porém, a corrente elétrica que atravessa cada resistor pode ser diferente, caso os resistores tenham resistências elétricas diferentes.

Assim, o inverso da resistência equivalente de um circuito é igual a soma dos inversos das resistências de cada resistor presente no circuito. Todos os resistores encontram-se ligados sob o mesmo potencial elétrico e a resistência equivalente é menor que a menor das resistências.

A associação mista de resistores, pode haver tanto ligações em paralelo quanto ligações em série. Para solucioná-la é bem simples, apenas basta resolver separadamente, os resistores que encontram-se ligados em paralelo e os resistores que encontram-se ligados em série representado como o circuito a seguir



4. Oque é Arduino

O Arduino foi desenvolvido em 2005 por um grupo de 5 pesquisadores : **Gianluca Martino, David Mellis, Massimo Banzi, David Cuartielles e Tom Igoe**. O objetivo proposto era elaborar um dispositivo que fosse ao mesmo tempo barato, funcional e fácil de programar, sendo dessa forma acessível a estudantes e projetistas amadores. Além disso, foi adotado o conceito de hardware livre, o que significa que qualquer um pode montar, modificar, melhorar e personalizar o Arduino, partindo do mesmo hardware básica.

Assim, foi criada uma placa composta por um microcontrolador Atmel, circuitos de entrada/saída e que pode ser conectada

facilmente à um computador e programada via IDE(**Integrated Development Environment**) utilizando uma linguagem baseada em C/C++, sem a necessidade de equipamentos extras além de um cabo USB.

O arduino é caracterizado por varias elementos em sua estrutura demonstrado abaixo

Principais elementos de uma placa Arduino Uno

1- **Microcontrolador:** conhecido como o cérebro do arduino. Um computador inteiro dentro de um pequeno chip. Este é o dispositivo programável que roda o código que enviamos à placa.
2- **Conector USB:** conecta a placa ao computador. É por onde o computador e o Arduino se comunicam com o auxilio de um cabo USB, Além de ser uma opção de alimentação da placa.
3- **Pinos de entrada e saída:** Pinos que podem ser programados para agirem como saídas ou entradas fazendo

com que o arduino interaja com o meio externo. O UNO R3 possui 14 portas digitais, 6 pinos de saída analógica e 6 pinos de entrada(PWM).

4- **Pinos de alimentação:** Fornecem diversos valores de tensão que podem ser utilizados para energizar os componentes do seu projeto. **Obs: Devem ser usados com cuidado, para que não sejam forçados a fornecer valores de corrente superiores ao suportado pela placa.**
5- **Botão de reset:** Botão que reinicia a placa.
6- **Conversor Serial:** USB e LEDs TX/RX: é responsável para que o computador e o microcontrolador conversem, é necessário que existe um chip que traduza as informações vindas de um para o outro. Os LEDs TX e RX acendem quando o Arduino está transmitindo e recebendo dados pela porta serial respectivamente.
7- **Conector de Alimentação:** responsável por receber a energia de alimentação externa, que pode ter uma tensão de no mínimo 7 Volts e no máximo 20. Volts e uma corrente mínima de 300ma
8- **LED de alimentação:** indica se a placa está energizada.
9- **LED Interno:** LED conectado ao pino digital 13

Quando as pessoas vão mexer com o software do Arduino oque chama mais atenção é oque trás mais duvida como primeira impressão são as bibliotecas no software do Arduino, existem três tipos diferentes de bibliotecas de software disponíveis:

- **core**(biblioteca essencial)

- **padrão**

- **Adicionais**(de terceiros)

A **biblioteca core** vem instalada na IDE do Arduino e é imprescindível para o desenvolvimento de programas, desde os mais simples como acionar um LED até projetos complexos, como realizar automação de uma residência. Desta forma, a programação do Arduino fica muito simplificada, pois o programador não tem a necessidade de entender como o código da biblioteca funciona internamente, basta estudar e entender como utilizá-lo.

As **bibliotecas padrão** são incluídas na instalação do IDE do Arduino, pois o Arduino possui recursos de memória limitados, e assim essas bibliotecas somente são incluídas de forma explícita quando você necessita delas. As bibliotecas padrão do Arduino são:

Ethernet – permite conectar o Arduino à Internet ou à rede local usando um shield Ethernet.

EEPROM – Usada para ler e gravar dados em uma memória EEPROM no Arduino. No Uno a EEPROM tem tamanho de 1024 bytes e no Mega, de 4096 bytes.

Firmata – Esta biblioteca permite a comunicação entre o Arduino e aplicações em um computador via protocolo de comunicação serial.

GSM – Conecta a uma rede GSM/GPRS usando um shield GSM.

LiquidCrystal – Com essa biblioteca podemos controlar displays de cristal líquido (LCD).

SD – Essa biblioteca Biblioteca muito importante, usada para que seja possível escrever e ler dados em cartões de memória SD/SDHC.

Servo – Controlar motores servo.

SPI – Comunicação com dispositivos usando o barramento SPI que significa Serial Peripheral Interface.

SoftwareSerial – Permite a comunicação serial usando os pinos digitais da placa.

Stepper – Controlar os motores de passo

TFT – Permite desenhar imagens e formas e escrever texto em uma tela TFT

WiFi – Permite conexão à rede local e Internet por meio de um shield WiFi.

Wire – Biblioteca que permite enviar e receber dados por meio de uma interface TWI/I2C (interface a dois fios) em uma rede de dispositivos ou sensores.

As **bibliotecas de Terceiros** são disponibilizadas por desenvolvedores diversos que contribuem

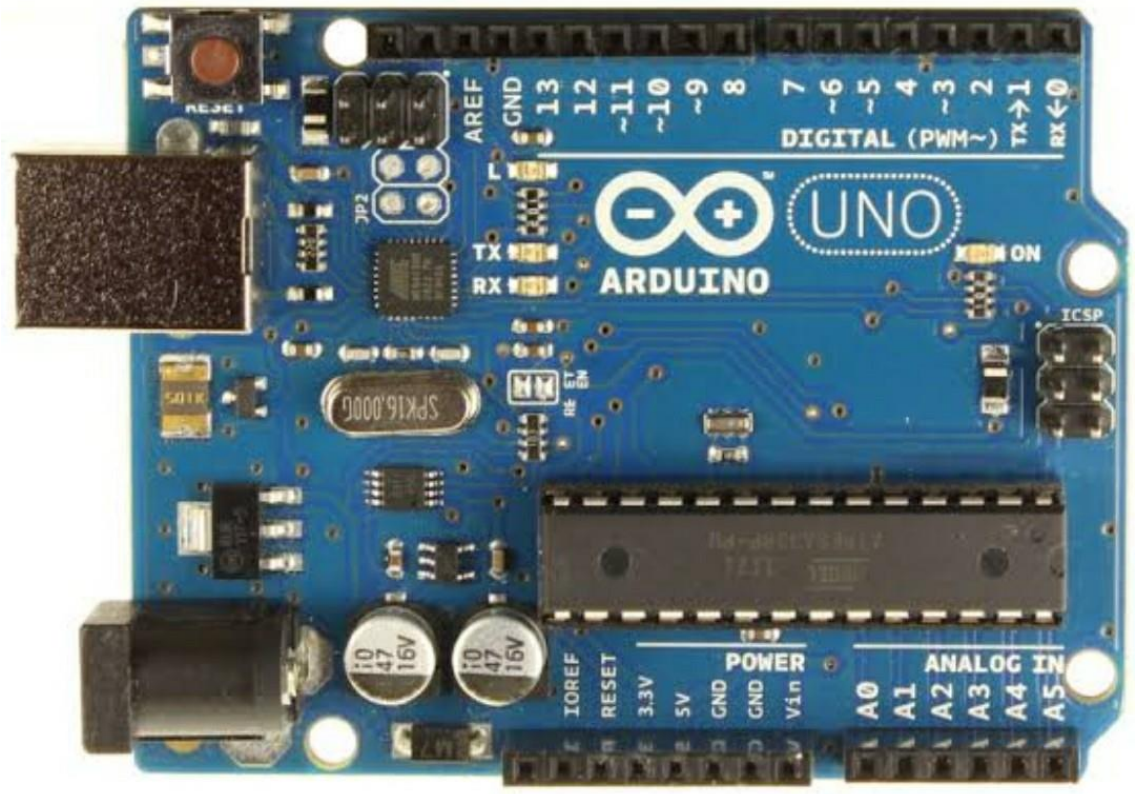
voluntariamente com software para a plataforma, e não são distribuídas por padrão com IDE do Arduino. Para usá-los, você precisa baixá-los e então efetuar sua instalação por meio do IDE. Elas oferecem funções

adicionais a bibliotecas existentes ou novas funcionalidades não presentes em nenhuma biblioteca padrão, permitindo estender o uso do Arduino de forma praticamente ilimitada.

5. Tipos de Arduino

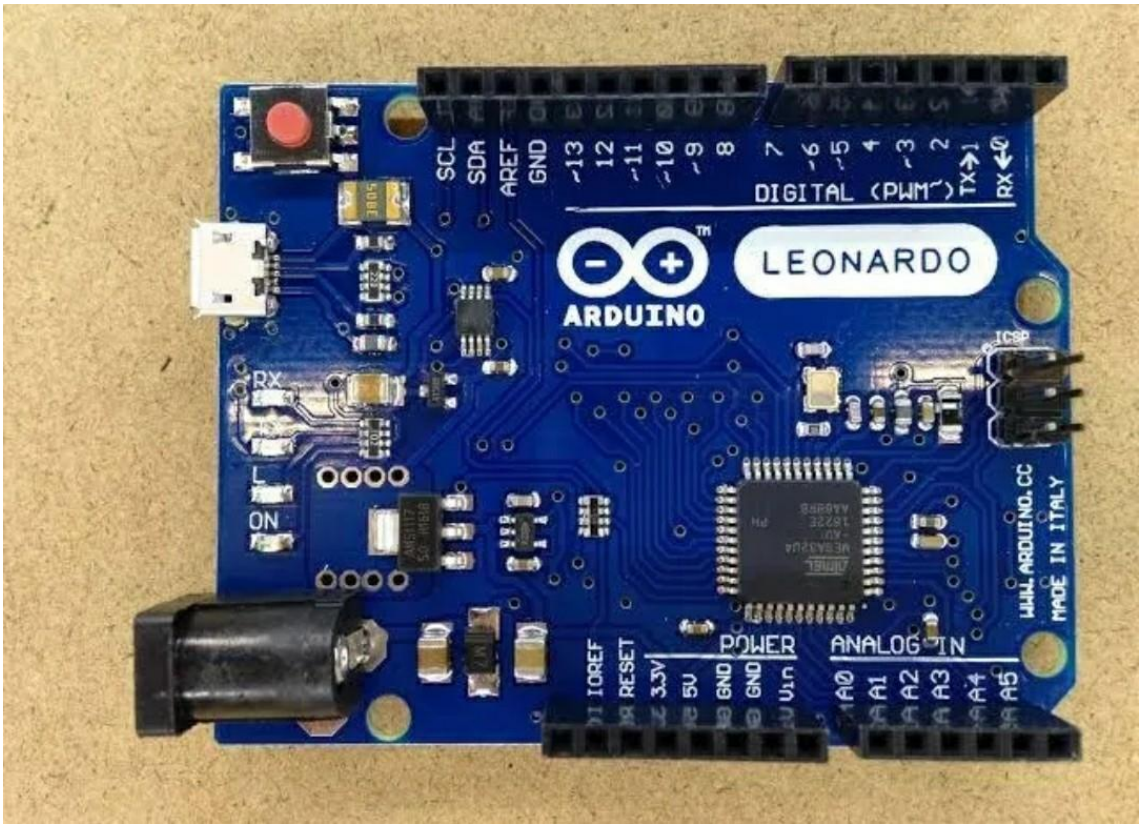
Existem diversos tipos de Arduino, devido a existir essa grande diversidade de Arduino muitas pessoas acabam ficando com duvida sobre qual Arduino comprar. Aqui irei demonstrar os tipos de Arduino e qual vai do gosto de cada pessoa demonstrado nas definições abaixo:

Arduino UNO: Dentre os tipos de Arduino, este costuma se a primeira opção para quem vai comprar um Arduino, pois possui um bom número de portas disponíveis, e grande compatibilidade com os shields disponíveis no mercado. Possui processador ATMEGA328, 14 portas digitais, sendo que 6 delas podem ser usadas como saidas PWM, e 6 portas analógicas. A alimentação (selecionada automaticamente), pode vir da conexão USB ou do conector para alimentação externa (recomendável 7 à 12 Vdc).



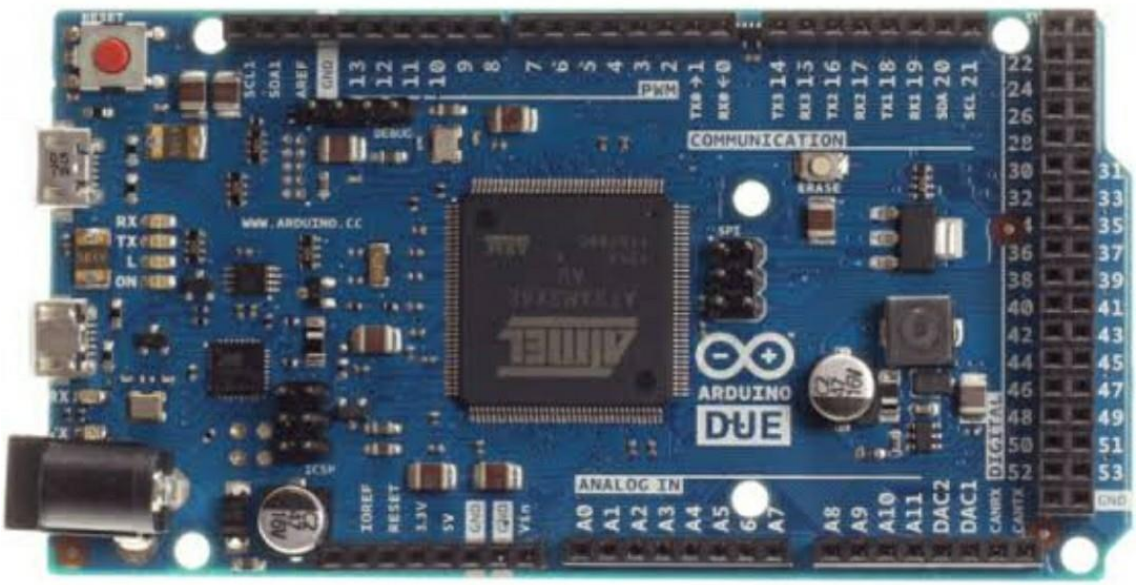
Arduino Leonardo: O Arduino Leonardo possui algumas semelhanças ao Arduino UNO mas você irá observar que vai haver grandes diferenças entre os dois, com microcontrolador **Atmega32u4**, possuindo 20 portas digitais, das quais 7 podem ser usadas como PWM, e 12 como portas analógicas e Esta placa também possui clock de 16 Mhz e conexão pra alimentação externa. Diferentemente do Arduino Uno, possui conector micro-usb para ligação ao computador.

Outra característica dessa placa é o chip de conexão USB integrado ao microcontrolador, o que elimina a necessidade de um chip adicional de comunicação na placa, e permite que o Arduino Leonardo seja reconhecido pelo computador como se fosse um mouse ou um teclado, e não necessariamente como uma porta serial.

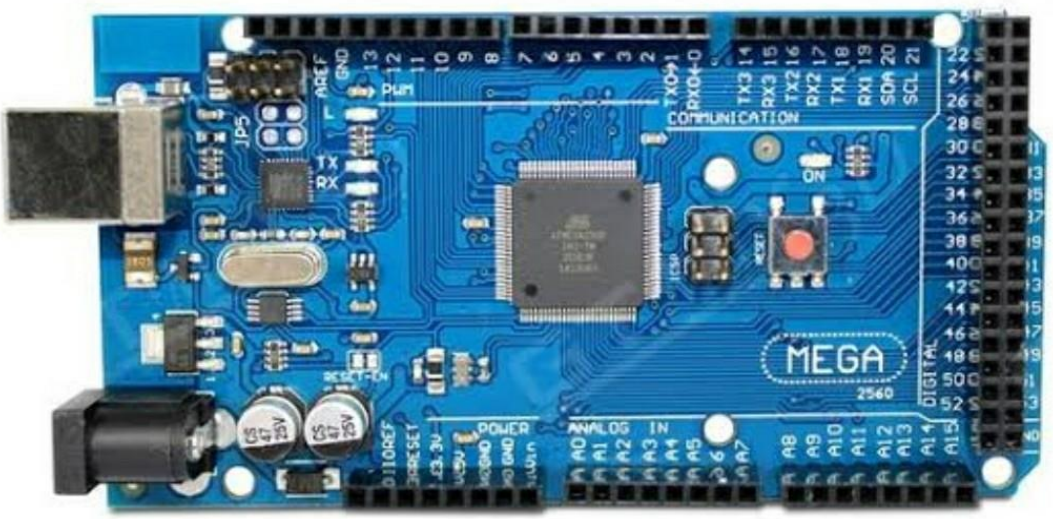


Arduino Mega 2560: Possui um microcontrolador ATmega2560 e 54 portas digitais, das quais 15 podem ser usadas como PWM, além de 15 portas analógicas. Clock de 16 MHz, conexão USB e conector para alimentação externa. Ideal para projetos mais elaborados que exijam grande número de entradas e saídas.

Arduino Due: Entre todos os tipos de Arduino está é a placa com maior capacidade de processamento, baseada em um microcontrolador ARM de 32bits e 512kb de memória totalmente disponível para programas/aplicações. O bootloader já vem gravado de fábrica em uma memória ROM dedicado. É caracterizada por possuir 54 portas digitais, das quais 12 podem ser usadas como PWM, e 12 portas analógicas. Possuí também 4 chips controladores de portas seriais, conexão USB e conector para alimentação externa. As ligações desta placa exigem especial atenção pois as portas trabalham à 3.3V, o que pode comprometer o uso dos Shields disponíveis no mercado, que geralmente trabalham com 5V.



Arduino Mega ADK: é baseado no ATmega2560, esta placa possui uma conexão USB dedicada à ligação com dispositivos baseados em Android, como telefones celulares. Possui 54 portas digitais, das quais 15 podem ser usadas como PWM, 16 portas analógicas, 4 chips dedicados à comunicação serial, clock de 16 MHz e conexão ao computador via USB. Também possui conector para alimentação externa.



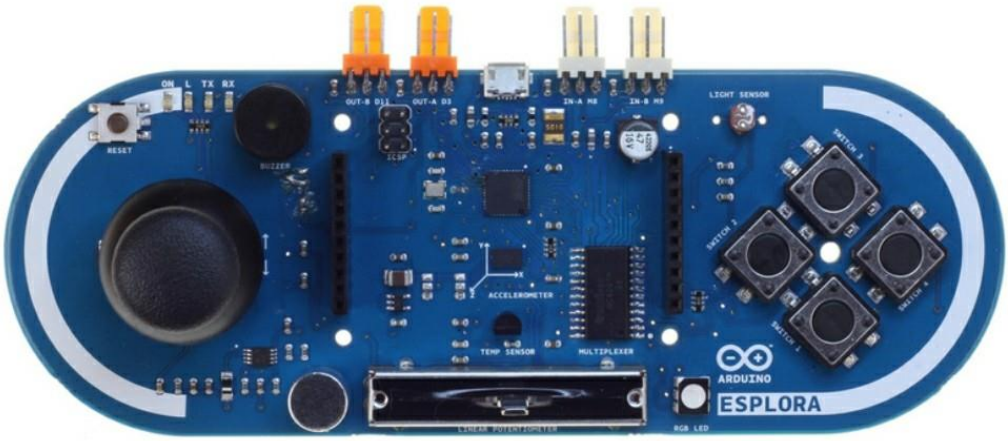
Arduino Nano: Baseada no microcontrolador ATmega328 para versão 3.x ou ATmega168 para versão 2.x, uma grande diferença que esse Arduino possui em relação á outras placas, não possui conector para alimentação externa, sendo alimentada por um conector USB Mini-B. É uma placa desenvolvida pela Gravitech.

Arduino Pro Mini: Placa compacta, ideal para projetos permanentes e que não necessitem de grande poder de processamento ou constante atualização. O microcontrolador utilizado por esta placa é o ATmega168 que, dependendo da versão da placa, roda à 8 MHz na versão 3,3V ou 16 MHz na placa de versão 5V.

O Arduino Pro Mini possui 14 portas digitais, sendo que 6 podem usadas como PWM, e 8 portas analógicas. Não possui conexão USB ou conector para alimentação externa. Para comunicação com o computador, pode ser adquirido um módulo USB separadamente, ou utilizar uma placa Arduino para programação.

Arduino Esplora: O seu formato nós lembra muito um controle de videogame e em certas situações pode até ser usado como um, dependendo da criatividade do desenvolvedor), O Arduino Esplora é totalmente diferente de todos os outros tipos de Arduino, principalmente por possuir diversos sensores na sua construção. Nessa placa vem embutido um buzzer, um joystick, um potenciômetro deslizante, um sensor de temperatura, um acelerômetro, um LED RGB, um sensor

de luz LDR, 4 pushbuttons e um microfone. Além de tudo, ainda possui um soquete para tela LCD.



6. Tipos de variáveis

Em um programa a variável é um espaço de memória com tamanho pré-definido associado ao tipo da variável criada. Por exemplo, se você precisa fazer uma conta aritmética bem simples em seu programa, você pode criar uma variável do tipo inteiro com o nome de *X*, assim pode ser demonstrado a partir da seguinte linha de código:

```
X= 2+3;
```

O valor 5 vai preencher o espaço de memória associado à variável *X*. Existem diferentes tipos de Variáveis, no Arduino as variáveis podem ser representadas a partir dos seguintes tipos:

Booleanos: é bem idêntico ao padrão bool, o book como também o Booleanos pode armazenar dois valores: *false* ou *true* aonde cada variável tem a ocupação de um byte na memória.

Char: é usado para armazenar um caractere, todos os caracteres são armazenados como números. Você pode ver a codificação na tabela ASCII. O tipo de dado *char* ocupa ao menos 8 bits.

Byte: armazena valores numéricos de 8-bit sem sinal, de 0 a 255.

Int: um int armazena um valor de 16-bit, variáveis int armazenam números negativos com uma técnica chamada *Complemento de 2*. O bit mais significativo, as vezes chamada de o "*bit de sinal*", indica que o número é negativo. O restaurante dos bits são invertidos e 1 é adicionado.

Unsigned int: como o int armazenam um valor de 2bytes.Em vez de guardar números, no entanto, esses

apenas armazenam valores positivos, garantindo um intervalo útil de 0 a 65, 525.

Long: Variáveis long são variáveis de tamanho extendido para armazenamento de números, armazenam 32 bits.

Unsigned long: número inteiro de 16 bits sem sinal

Float: floats possuem apenas 6-7 dígitos decimais de precisão. Isso para o número total de dígitos, não de dígitos à direita do ponto decimal.

Double: número real de precisão dupla

String: sequência de características

Void: é um topo de vazio e não tem tipos.

Dessa forma, dependendo das necessidades matemáticas e características da lógica do seu programa, você vai precisar de declarar variáveis de um tipo ou de outro.

7. Pinos Digitais com Interrupções

O primeiro parâmetro de *attachInterrupt()* é o número da interrupção. É recomendado usar *digitalPinToInterrupt(pino)* para converter o número do pino digital para o número específico da interrupção. Por exemplo, se você usar o pino 2, passe *digitalPinToInterrupt (2)* como o primeiro parâmetro de *attachInterrupt().*

Interrupções são úteis para fazer coisas automaticamente em programas de microcontroladores, e podem ajudar a resolver problemas de temporização. Boas tarefas para se usar uma interrupção podem incluir a leitura de um codificador rotativo, ou monitorar entradas de dados pelo usuário.

Supomos que você quisesse ter certeza que um programa sempre captura os pulsos de um codificador rotativo, sem nunca perder um pulso, seria muito complicado criar um programa que pudesse fazer qualquer outra coisa além disso, porque o programa precisaria checar constantemente os pinos do codificador, para capturar os pulsos exatamente quando eles ocorreram. Outros sensores são similarmente dinâmicos também, como um sensor de som que pode ser usado para capturar um clique, ou um sensor infravermelho para capturar a queda de uma moeda. Em todas essas situações, usar uma interrupção pode permitir o microcontrolador trabalhar em outras tarefas sem perder a interrupção.

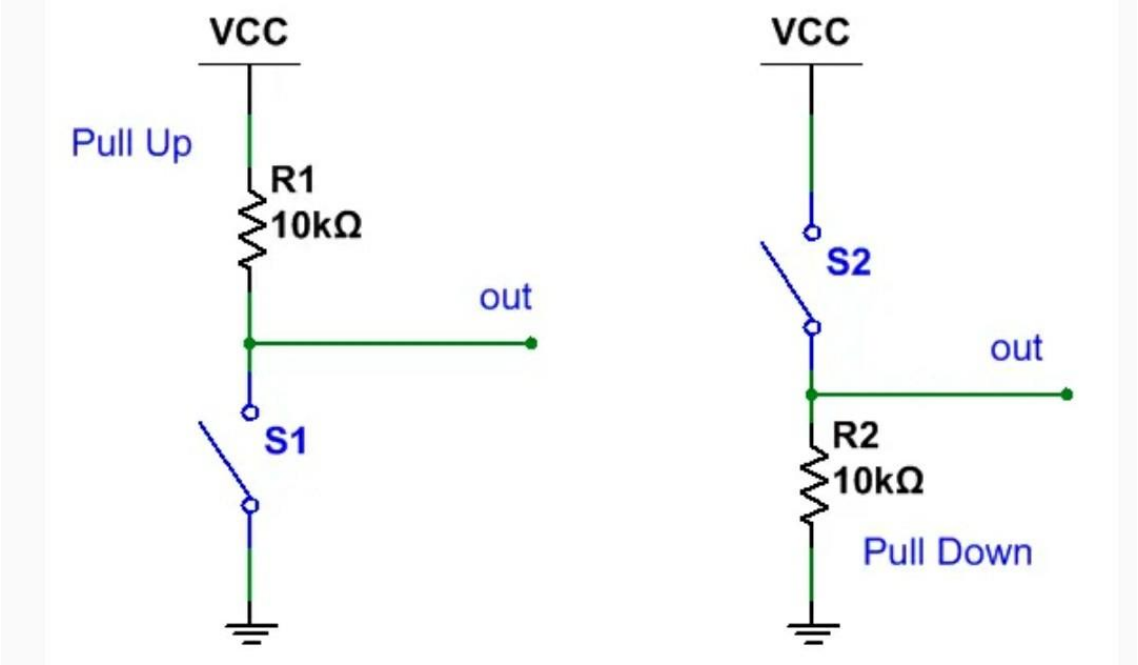
Normalmente você deve usar *digitalPinToInterrupt(pino)*, em vez de colocar diretamente o número da interrupção no seu sketch. Os pinos específicos para interrupções, e seu mapeamento para o número da interrupção variam para cada tipo de placa. O uso direto dos números das interrupções podem parecer mais simples, porém pode causar problemas de compatibilidade quando seu sketch for executado em uma placa diferente.

8. Como resolver o efeito bouncing

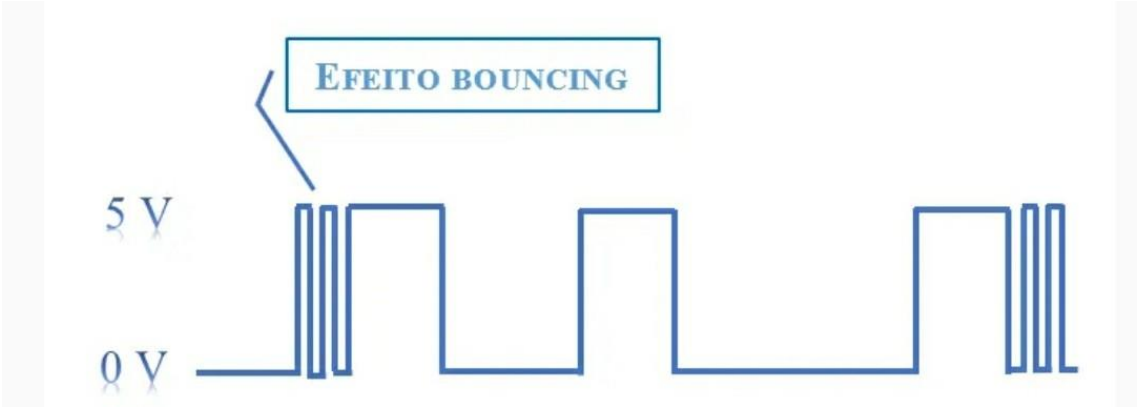
O efeito Bouncing se trata de idas e vindas, do nível lógico alto e o nível lógico baixo, antes de efetivamente estabilizar. Essas oscilações rápidas podem gerar acionamentos indevidos no nosso programa. Pois o mesmo tende a interpretar que você pressionou rapidamente várias vezes a chave, quando na verdade foi apenas uma vez. Vamos construir um circuito bem simples, um circuito muito utilizado para acionamentos simples de chaves. Neste circuito, iremos envolver uma chave mecânica e um resistor de 10K. Aqui o resistor é conhecido como resistor de pull-down e não um resistor de *pull-up*.

Ao pressionarmos a chave, estaremos fazendo uma ligação direta com o terra. Como não há nenhuma tipo de resistência no caminho, nosso pino então possuirá o valor

de nível lógico baixo. O mesmo acontece de modo inverso no circuito envolvendo o resistor de *pull-down*



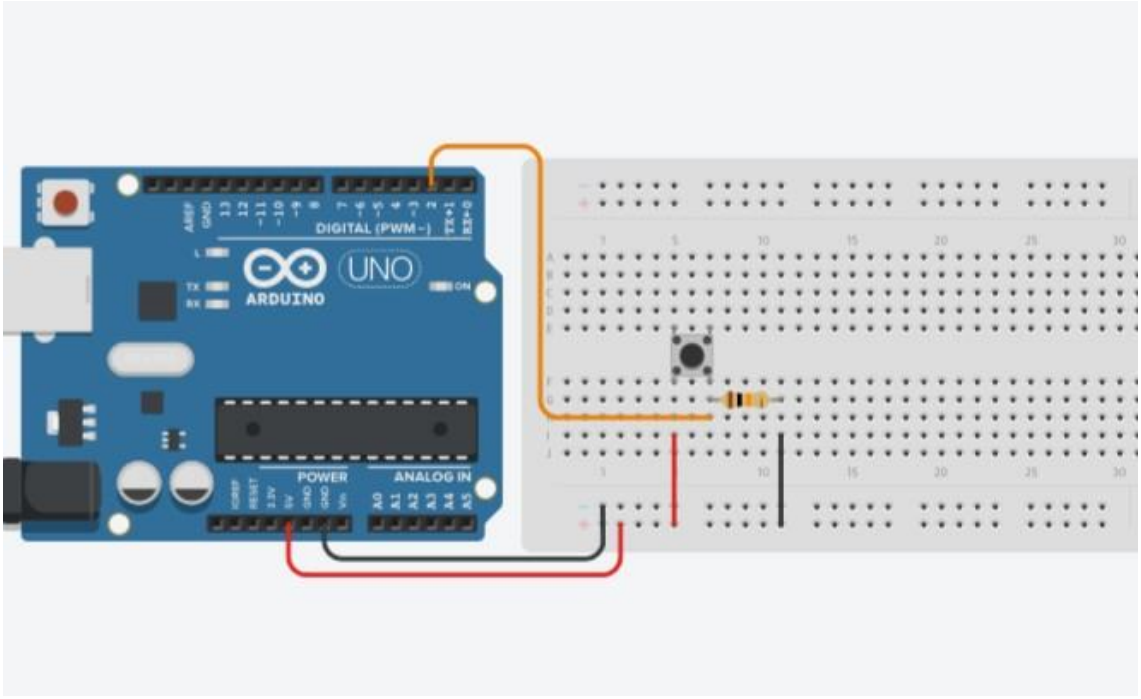
Podemos observar a diferença entre o circuito da direita e o circuito da esquerda, quando o resistor estiver ligado com o VCC será Pull-Up e quando o resistor estiver ligado ao terra será Pull-Down.



A imagem diz tudo. Como você pode ver, esse efeito é caracterizado por idas e vindas, do nível lógico alto e o nível lógico baixo, antes de efetivamente estabilizar. Entre outras palavras, essas oscilações rápidas podem gerar acionamentos indevidos no nosso programa. Pois o mesmo tende a interpretar que você pressionou rapidamente várias vezes a chave, quando na verdade foi apenas uma vez.

Uma das soluções para resolver o efeito Bouncing é simplesmente adicionar um capacitor ao seu circuito. O capacitor irá atenuar o sinal, sendo que a malha resistor/capacitor irá gerar um tempo de atraso no circuito. Devido a esse tempo, isso fará com que as rápidas oscilações indesejadas sejam atenuadas.

Outra solução que pode ser utilizada é o Software, é realizado utilizando um pequeno atraso no código, que seria o tempo suficiente para que a etapa do efeito bouncing passasse, e desta forma minimizar esse problema.



O código utilizado no projeto esta abaixo, todo comentado para facilitar seu entendimento.

```
const int buttonPin = 2; // Porta onde a chave está
conectada
const int ledPin = 13; // Led na porta 13

int buttonState; // O valor atual da chave no circuito
int ledState = HIGH; // Estado atual do pino do led
int lastButtonState = LOW; // O valor prévio da chave no
circuito

long lastDebounceTime = 0; // Variável utilizada na
temporização
long debounceDelay = 30; // tempo para estabilizar e
minimizar o efeito bouncing
void setup()
{
pinMode(buttonPin, INPUT); // Define o botão ou chave como
entrada
pinMode(ledPin, OUTPUT); // Define o led como saída
```

```
digitalWrite(ledPin, ledState); // Define estado atual do
led
}

void loop(){

// faz a leitura da chave e armazena na variável
int reading = digitalRead(buttonPin);

// Verifica se houve alterações com o valor prévio da chave
if (reading != lastButtonState) {
lastDebounceTime = millis(); // Reinicia a variável de
temporização
}
if ((millis() - lastDebounceTime) > debounceDelay) {

// Depois de esperar o tempo especificado no começo,
verifica se o estado atual da chave mudou
if (buttonState == HIGH) {
ledState = !ledState;
}
}

// Atribui o valor para o led
digitalWrite(ledPin, ledState);

// Atualiza a variável com o valor lido na chave
lastButtonState = reading;
}
```

Obs: Não faça todas as ligações com o seu ARDUINO desconectado da alimentação.

9. Recursos watchdog e EEPROM no ARDUINO

O **watchdog** é um recurso simples mas com um objetivo vital para sistemas embarcados: garantir que o software volte para uma condição conhecida em caso de algum erro ou situação inesperada.

Trata-se de um *time* que, uma vez ligado, precisa ser rearmado dentro de um tempo limite, o microcontrolador é automaticamente reiniciado.

Para usar o watchdog precisamos descer um nível abaixo da biblioteca do ARDUINO, usando a biblioteca do compilador C. Por este motivo, as informações abaixo se aplicam somente aos modelos baseados nos microprocessadores AVR(*Atmega e ATtinny*). No seu programa você precisa incluir o header wdt.h:

```
#include <avr/wdt.h>
```

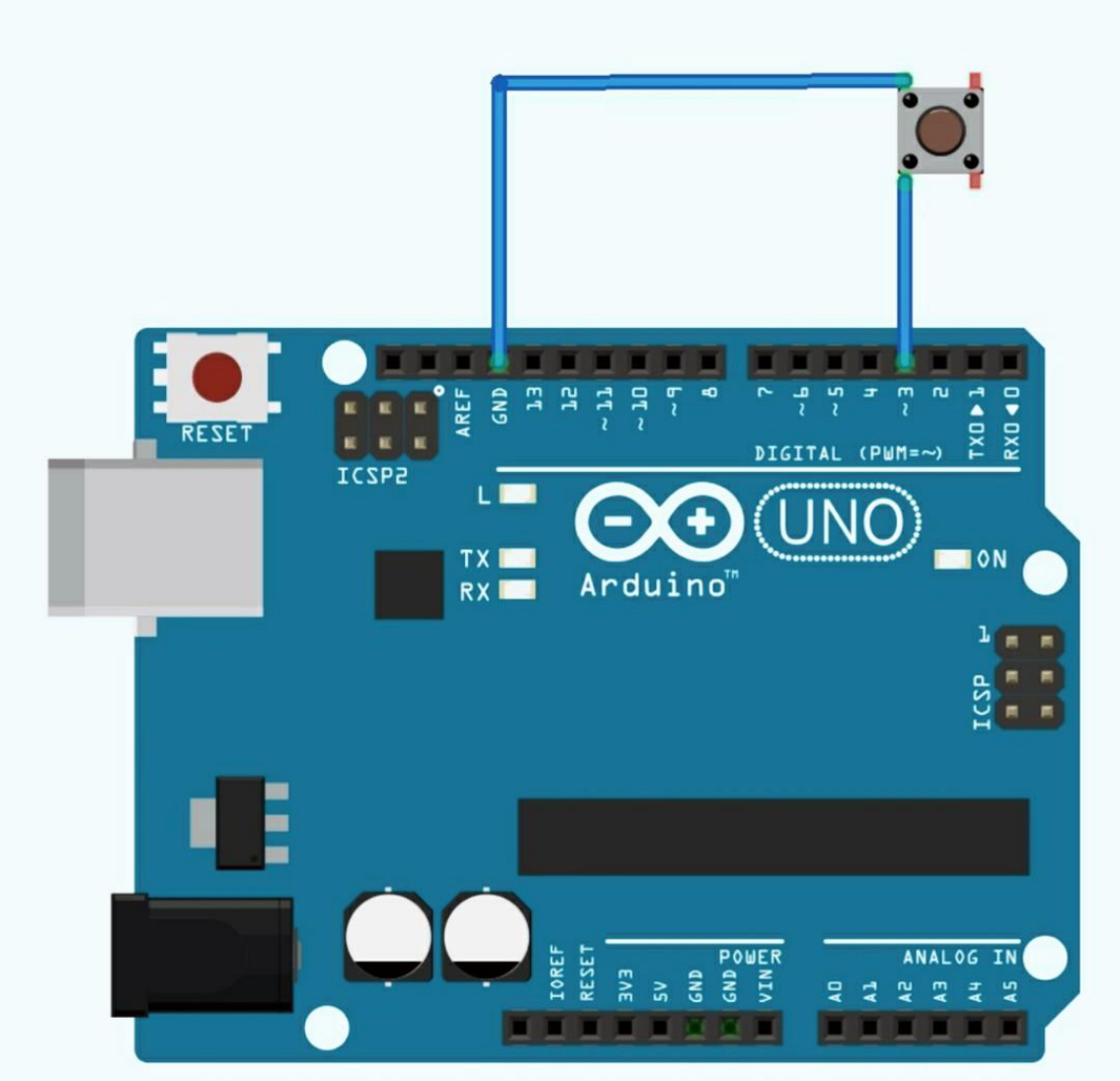
Para ligar o watchdog, chame a função wdt_enable, informando o tempo limite:

```
wdt_enable(WDTO_2S);
```

O valor no exemplo acima corresponde a 2 segundos; estão também disponíveis WDTO_8S, WDTO_4S, WDTO_1S, WDTO_,500MS,WDTO_60MS, WDTO_15MS. A temporização do watchdog é fornecida por um oscilador interno ao microcontrolador e independe do clock de execução das instruções.

Para rearmar o watchdog, basta chamar *wdt_reset().* Normalmente você irá colocar chamadas nos pontos normais de execução, para garantir que não ocorra o reset durante o funcionamento correto.

O exemplo do circuito abaixo permite ver o watchdog em funcionamento, para rodá-lo conecte um botão entre o pino 3 e GND. Se você mantiver o botão apertado por mais de 2 segundos o ARDUINO reiniciará.



O código utilizado no projeto acima é o seguinte:

```
#include <avr/wdt.h>

const int pinBotao = 3;
const int pinLED = 13;

void setupt() {
  wdt_enable(WDTO_2S);
  pinMode (pinBotao, INPUT_PULLUP);
  pinMode (pinLED, OUTPUT);
  Serial.begin(9600);
```

```
  Serial.println("Reset");
  digitalWrite (pinLED, HIGH);  // para indicar o reset
  delay (500);
  digitalWrite (pinLED, LOW);
}

void loop() {
  wdt_reset();
  while (digitalRead(pinBotao) == LOW) {
    delay (10);
  }
}
```

A **EEPROM** combina a não volatilidade da Flash(manutenção dos dados sem alimentação) com algo próximo à flexibilidade de leitura e escrita da RAM.

O motivo pela qual devemos usar o EEPROM é para armazenar informações que queremos que sobrevivam a um desligamento, como parâmetro ou variáveis importantes. O ATmega328 possui generosos 1K de EEPROM. Ao planejar o uso da EEPROM, é preciso levar em conta que existe um limite de 100.000 escritas em cada posição e que a escrita e leitura são bem mais demoradas que na RAM.

Dicionário técnico

Resistor*: Restringe o fluxo de eletricidade em um circuito, reduzindo a voltagem e é muito utilizado em eletrônica, o resistor*

tem a finalidade de transformar energia elétrica em energia térmica por meio do efeito joule, com a capacidade de limitar a corrente elétrica. Os resistores possuem um código de cores isso por possuir um tamanho muito reduzido, é invariável imprimir nos resistores as suas respetivas resistências. Em um resistor as primeiras três faixas servem para indicar o valor nominal e a última faixa, a porcentagem na qual a resistência pode variar.

***LED**: Diodo emissor de luz que se acende quando a eletricidade passa através dela se transformando em energia luminosa possuindo uma variação de cor, em um LED existe o lado negativo conhecido como catodo e o positivo é o anodo.*

***Protoboard**: dependendo do tamanho pode ter 30 linhas, 10 colunas e dois pares de linha sendo um par positivo e o outro par é negativo.*

***Botão**: Chave que fecha um circuito enquanto está na posição pressionada.*

***Potenciômetro**: Tipo de resistor cuja resistência muda quando se vira uma chave. Se todos os três terminais são usados, ele atua como um divisor de tensão. O potenciômetro consiste em um elemento resistivo, chamado de* pista*, ou* trilha*, e de um cursor móvel, que se movimenta ao longo de um eixo, rotatório ou linear. De acordo com a posição desse cursor ao longo do eixo, a resistência obtida será diferente, dentro de certos limites característicos do componente em questão.*

***Capacitor**: Armazena e libera energia elétrica em um circuito.*

***Interruptor deslizante**: Interruptor com duas posições: aberta e fechada.*

***Bateria 9V**: Bateria comum, ótima para aplicações que empregam maior potência.*

***Capacitor polarizado**: capacitor direcional usado para armazenar e liberar energia elétrica em um circuito.*

***Diodo:** Permite o fluxo de eletricidade em uma única direção.*

***Diodo Zener**: assemelha-se a um Diodo normal, mas o Diodo Zener permite o fluxo de corrente invertido.*

***Indutor**: Resiste à mudança no fluxo de corrente.*

***Fotorresistor**: Sensor cuja resistência muda segundo o volume da luz detectado.*

***Fotodiodo**: Um dispositivo semicondutor que converte luz em corrente elétrica. O tempo de resposta de um fotodiodo tende a diminuir quando sua superfície aumenta.*

***Fototransístor**: um transistor bipolar encapado em uma capa transparente que permite que luz possa atingir a base coletora da junção. O fototransistor funciona de maneira similar a um fotodiodo, apresentando uma sensitividade muito maior à luz, pois os elétrons gerados pelos fótons na junção da base-coletora são aplicados na base do transistor, e sua corrente é então amplificada pela operação do transistor. O fototransistor apresenta um tempo de resposta maior do que o fotodiodo.*

***Sensor de infravermelho**: Detecta sinais de infravermelho emitidos por dispositivos como controles remotos.*

***Sensor de distância Ultrassônico**: Sensor que utiliza ondas de som para determinar a que distância está um objeto. O HC-SR04 é um sensor Ultrassônico capaz de medir distâncias de 2cm a 4m, sem nenhum contato e com uma excelente precisão de 3mm.*

***Sensor PIR**: Um sensor infravermelho passivo é um sensor eletrônico que mede a luz infravermelha que irradia dos objetos em seu campo de visão. Eles são frequentemente usados em detectores de movimento baseados em PIR. O sensor PIR é comumente usado em aplicações de iluminação automática e alarmes de segurança.*

***Sensor de temperatura TMP36**: É um sensor de temperatura de baixa voltagem e de precisão centígrada. Ela fornece uma voltagem de saída que é linearmente proporcional à temperatura em graus Celsius.*

***Sensor de gás**: Sensor de gás Winsen utilizado para detectar fugas de gás, como o monóxido de carbono.*

***Teclado 4X4**: Teclado de 16 botões, com dígitos de 0 a 9, letras de A a D e os símbolos * e #.*

***LED RGB**: Tipo de LED que combina vermelho, azul e verde para produzir qualquer cor.*

***Lâmpada**: Lâmpada incandescente de 12V/3W.*

***NeoPixel**: LED RGB único, que pode ser controlado com o uso de um microcontrolador.*

***Motor de vibração**: Motor que vibra quando acionado.*

***Motor CC**: Motor que converte energia elétrica em energia mecânica.*

***Motor CC com codificador**: Motor de engrenagem planetária actobotics de 3 a 12V*

***Micro servo**: Motor cuja posição pode ser controlada com o uso de um microcontrolador, como um ARDUINO.*

***Motor de engrenagem de uso não profissional**: Motor com engrenagem usado com frequência para acionar rodas de robôs.*

***Piezo**: Tipo de buzzer que emite ruído em diferentes frequências.*

***Infravermelho remoto**: controle remoto que emite sinais em infravermelho, decodificados com o uso de um sensor*

***Visor de sete segmentos**: LED único com sete segmentos usado para exibir um número ou uma letra.*

***LCD 16 x 2**: Tela de cristal líquido que exibe duas linhas de 16 caracteres.*

***Transistor NPN e PNP**: componentes usados para amplificar ou trocar sinais eletrônicos.*

***Transistor nMOS de pequenos sinais**: Transistor para pequenos sinais controlado por voltagem.*

***Transistores nMOS e pMOS**: transistores para grandes sinais controlado por voltagem.*

***TIP120**: transistor NPN Darlington usado para alternância em equipamentos eletrônicos.*

***Relé SPDT**: Relé SPDT de 5V para alternância entre dois circuitos.*

***Relé DPDT**: Relé de força DPDT de 5V em miniatura.*