

FIA/P GRADUAÇÃO

TECNOLOGIA EM ANÁLISE E DESENVOLVIMENTO DE SISTEMAS

Disruptive Architectures: AI and IoT

PROF. ANTONIO SELVATICI

SHORT BIO



É engenheiro eletrônico formado pelo ITA, com mestrado e doutorado pela Escola Politécnica (USP), e passagem pela Georgia Institute of Technology em Atlanta (EUA). Desde 2002, atua na indústria em projetos nas áreas de robótica, visão computacional e internet das coisas, aliando teoria e prática no desenvolvimento de soluções baseadas em Machine Learning, processamento paralelo e modelos probabilísticos. Desenvolveu projetos para Avibrás, Rede Globo, IPT, CESP e Systax.

PROF. ANTONIO SELVATICI
profantonio.selvatici@fiap.com.br

1. DISPOSITIVOS DE IoT

| ARDUINO

Introdução

- Arduino é uma plataforma de hardware para a rápida execução de projetos eletrônicos, possuindo um microcontrolador Atmel AVR com suporte de entrada/saída embutido
 - É um projeto *open source*, tanto no que tange ao hardware quanto ao software (ou seja, pode ser copiado)
 - Utiliza componentes de baixo custo
 - Emprega uma IDE de programação simplificada baseada em *Wiring*, que simplifica o processo de criação de projetos de C++
- Origem: criado por professores da Ivrea Interaction Design Institute para facilitar e baratear a criação de projetos pelos alunos
- Pode ser usado como um computador independente, ou estar conectado via USB no modo FTDI, sendo mapeado em uma porta serial.

ARDUINO UNO

Arduino Uno



I POR QUE USAR O ARDUINO

- A IoT forma uma complexa rede cujos elementos constituem em:
 - Dispositivos móveis de interação (smart phones, tablets, etc)
 - Servidores de aplicações como webservices Java, .Net, Node.js, etc.
 - Sistemas de bancos de dados
 - Dispositivos de interação com o ambiente, com sensores e atuadores
- *O Arduino se encontra na última categoria, permitindo a conexão de sensores e atuadores, além da comunicação com outros dispositivos*

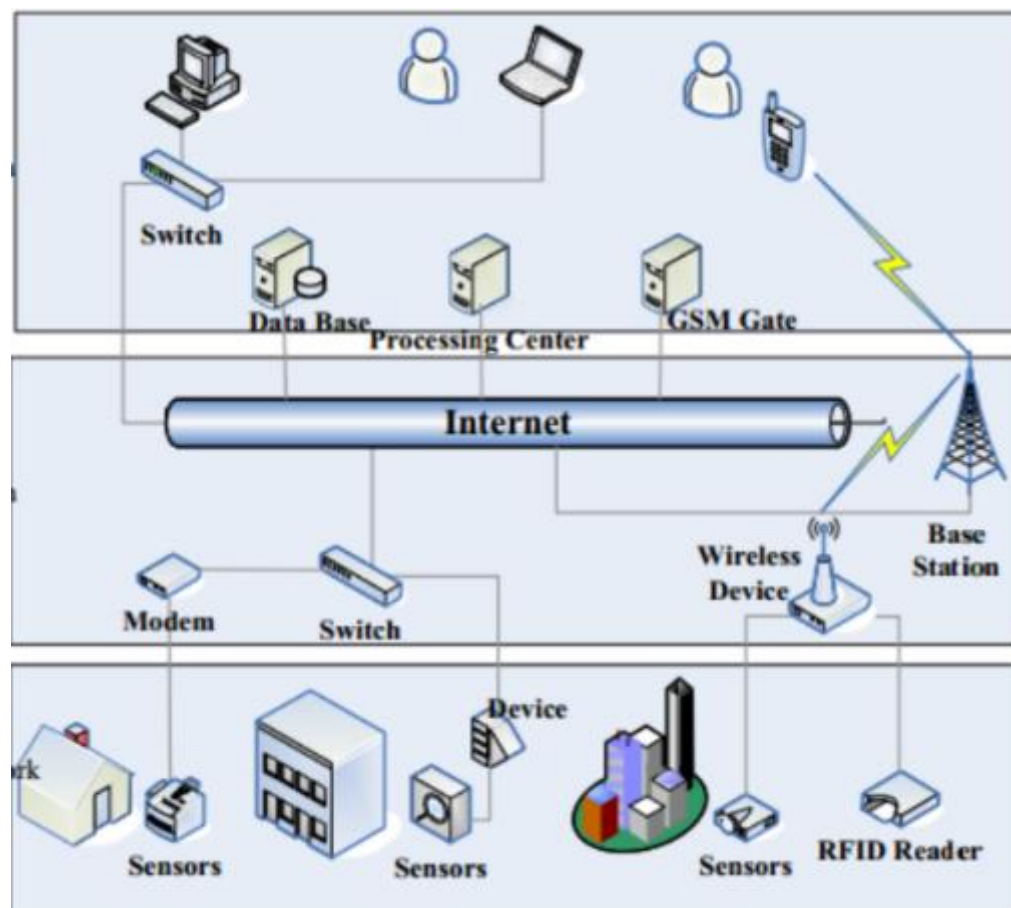
Arquitetura básica das aplicações de IoT

Como se relacionam os dispositivos, a internet e os usuários das aplicações [1]

Rede ou
Camada de
**Aplicações e
Serviços**

Rede ou
Camada de
Transmissão

Rede ou
Camada de
Sensores



Rede de Sensores

Coleta de dados, acionamento de dispositivos, comunicação local

- Rede de comunicação que interliga os diferentes objetos conectados
- É o “diferencial” da internet das coisas
 - onde atuam as tecnologias habilitadoras da IoT
- Comparada à “pele” da IoT, por onde ocorrem as trocas de informação com o mundo
 - Captura de dados por sensores
 - Execução de ações por atuadores
- Objetos sem conectividade própria são rastreados usando RFID ou outra forma de identificação
- Em geral, os objetos se comunicam em uma rede local (**WSN – *Wireless Sensor Networks***), que por sua vez se comunica com a internet através de gateways ou bridges
- Redes de comunicação de objetos muitas vezes usam tecnologias alternativas ao WiFi e 3G/4G, como Bluetooth, Zigbee, LoRaWan

Rede de Transmissão

Integra a rede de sensores à internet

- Sistema nervoso central da IoT, tendo o papel de transmitir os dados entre a rede de sensores e a rede de aplicação
- Corresponde à infraestrutura de comunicação que permite a interconexão de objetos, aplicações e seus usuários
- Integra os objetos inteligentes à internet, convertendo os protocolos de transporte próprios das redes de objetos ao TCP/IP

Rede de aplicação

Camada de provimento de serviços online e interação com o usuário, que se comunicam com os dispositivos de IoT através da rede de transmissão



- Formada pelos aplicativos de usuário final, bem como pelos serviços que permitem um melhor gerenciamento dos dispositivos e aplicações de IoT
- Expõe API's para acesso aos dados dos sensores e controle dos dispositivos
- Atualmente, a regra é usar *computação em nuvem* para prover os serviços de rede
 - **Gartner [2]:** Estilo de computação na qual recursos de TI escaláveis e elásticos são oferecidos como serviço usando tecnologias da internet.

PRIMEIRO PROGRAMA

Hello World!

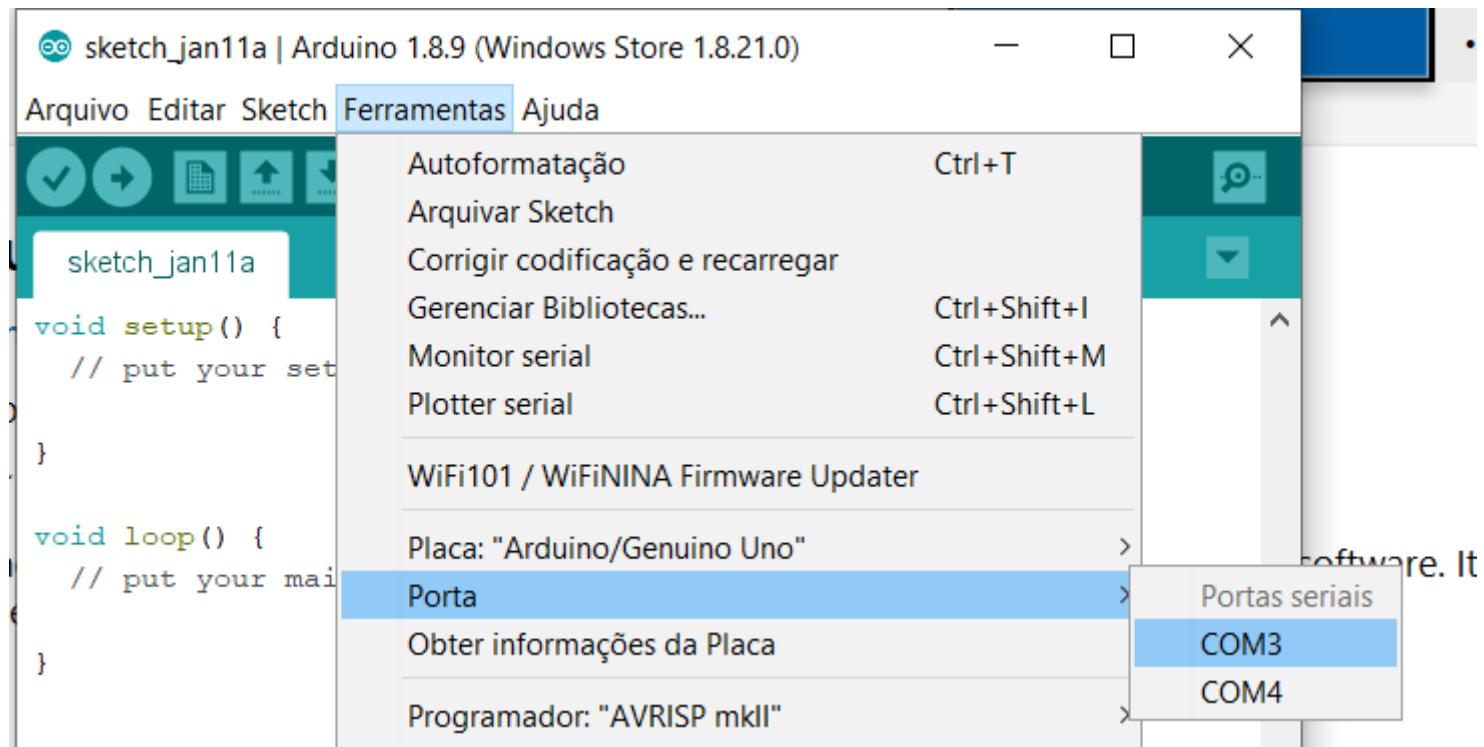
- Formar grupos de 2 ou 3 alunos
- Conectar o Arduino ao computador pelo cabo USB
- Abrir o programa “Arduino IDE”
- Alterar a porta serial e conexão (Tools -> Port)
- Digitar o programa abaixo na janela que se abriu (case sensitive!)

```
void setup() {  
    Serial.begin(9600);  
}  
void loop() {  
    Serial.println("Hello World!");  
    delay(3000);  
}
```

- Compilar e enviar o programa (símbolo de seta) 
- Abrir a janela do Monitor Serial (símbolo de lupa) 

SELEÇÃO DA PORTA

Escolhendo a porta serial onde o Arduino está conectado.
Geralmente a IDE reconhece a porta



COMPILANDO E SUBINDO O PROGRAMA

Compila e faz o upload
do programa

Monitor serial: monitora a saída
do Arduino para o PC

Apenas compila o
programa

```
sketch_feb16a | Arduino 1.5.9
Arquivo Editar Sketch Ferramentas Ajuda

int sensor = A1; // Pino analógico em que o sensor está conectado

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

void loop() {
  // Lendo o valor do sensor.
  int valorSensor = analogRead(sensor);

  // Exibindo o valor do sensor no serial monitor.
  Serial.println(valorSensor);

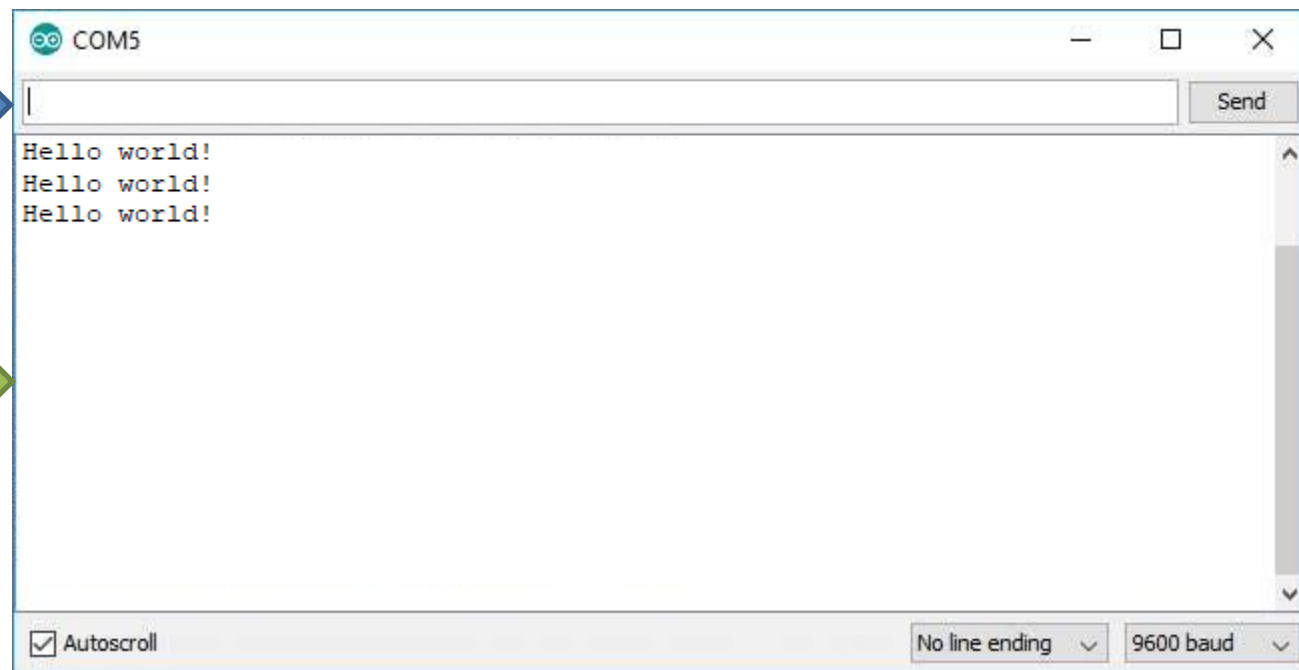
  delay(500);
}
```

MONITOR SERIAL

Campo de envio de mensagens para o Arduino



Área de recepção de mensagens vindas do Arduino



COMO FUNCIONA

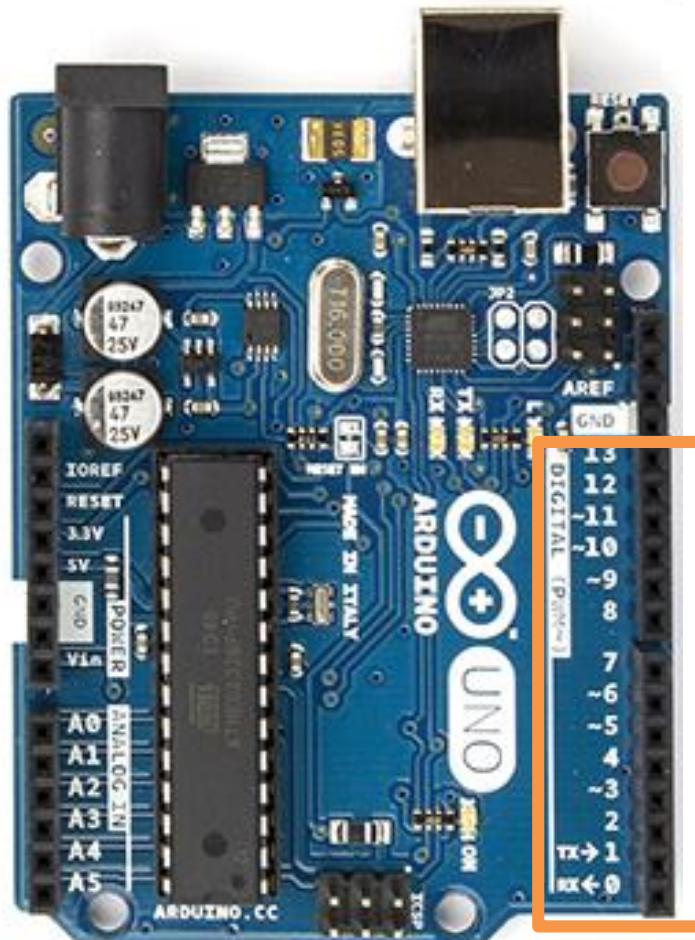
Hello World!

- A IDE do Arduino compila o programa escrito em C++, transformando-o em código em linguagem de máquina
- O código é enviado ao Arduino, que o salva em memória **não volátil** do tipo EEPROM.
 - Após desligar o Arduino, o programa continua gravado, e só será apagado quando sobrescrevermos outro programa
- Função `void setup();`
 - Chamada no início da execução do programa
 - Deve configurar os dispositivos a serem usados
- Função `void loop();`
 - Executada dentro do laço principal de execução
 - Executa enquanto a placa estiver ligada
- Uma vez que não conectamos nenhum monitor ou display ao Arduino, a saída de dados é realizada pela porta serial e recebida pelo computador
- Função `delay(milissegundos)`: pausa a execução pelo tempo especificado.

CONECTANDO COMPONENTES

Como perceber e interagir com o ambiente externo através de componentes eletrônicos

- Nosso objetivo ao usar o Arduino é trabalhar com sensores e atuadores, que correspondem a componentes eletrônicos que devem ser conectados ao Arduino
- Essa conexão é realizada através das **portas** ou **pinos** digitais, que são conexões que podem ler ou escrever um nível de tensão elétrica

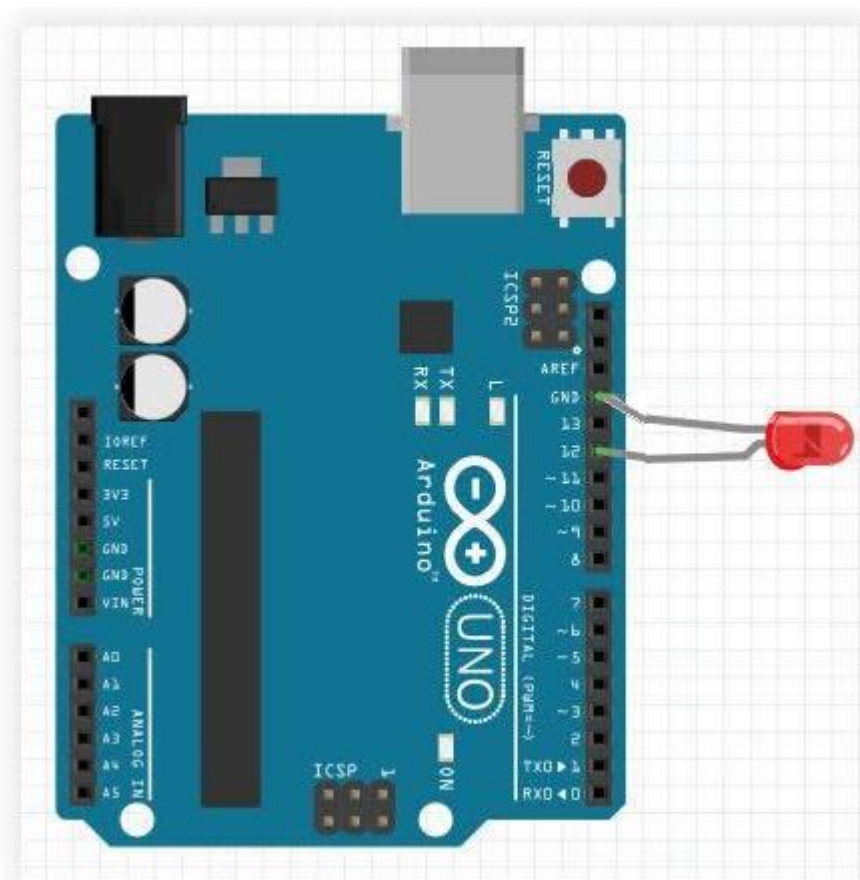


Portas
digitais

CONEXÃO POR CONTATO ELÉTRICO

Um componente deve possuir terminais ou “pernas” metálicas que devem encaixar firmemente nas portas do Arduino

- No exemplo ao lado, os terminais do LED estão conectados a duas portas do Arduino: 12 e GND
 - **Obs:** esta configuração é apenas ilustrativa, e pode danificar o LED a médio ou longo prazo.
- A porta 12, assim como as demais portas digitais, tem a capacidade de estar “ligada” ou “desligada”, dependendo da programação do Arduino
- Quando a porta está ligada, ela fornece uma tensão elétrica de cerca de 5V, e quando está desligada essa tensão fica próxima a 0V



I Tensão e corrente

- Para entender como funciona o Arduino e outros dispositivos eletrônicos, precisamos compreender os conceitos de tensão e corrente elétrica, por exemplo:
 - O Arduino Uno trabalha com lógica de 5V, ou seja, suas portas digitais fornecem 5V quando estão ligadas
 - Cada porta de saída pode fornecer uma corrente de até 40 mA.
 - O que isso significa?
- TL;DR: uma pilha comum fornece 1,5V de tensão, enquanto a tomada das casas fornecem 127V. Porém só existe corrente elétrica quando algum aparelho está conectado e ligado.

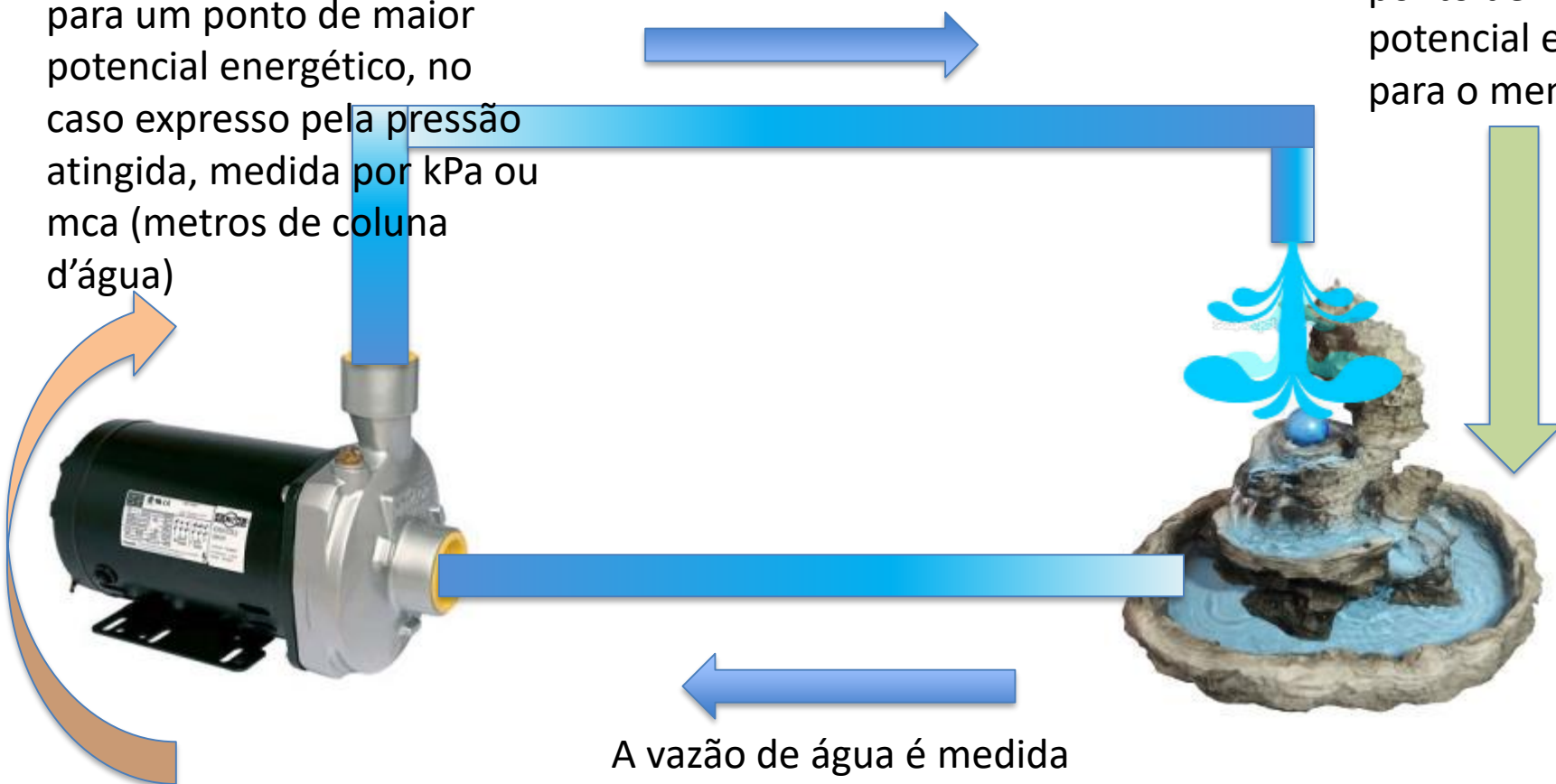
Circuitos elétricos

- A energia elétrica flui através do movimento das cargas elétricas livres, presentes nos materiais condutores elétricos, provocada por uma elevação do potencial elétrico em um ponto desse condutor.
 - Da mesma forma que uma pedra rola do alto da montanha para o vale, a carga elétrica livre tende a mover-se do ponto com maior potencial elétrico para o ponto de menor potencial
 - À diferença no potencial elétrico de um ponto a outro chamamos **diferença de potencial** ou **tensão elétrica**
 - À vazão de cargas elétricas que se movimentam dentro do condutor em um certo ponto chamamos **corrente elétrica**
- Para usarmos a energia elétrica, precisamos criar uma diferença de potencial, que por sua vez irá impor uma corrente elétrica a um material condutor
- A maneira que usamos a energia elétrica é na forma de um circuito elétrico, onde a corrente elétrica está sempre circulando devido a um fornecimento ininterrupto de tensão elétrica.
- Vamos comparar o circuito elétrico a uma fonte que está sempre jorrando água

Circuito elétrico: analogia com uma fonte

A bomba d'água usa energia externa para levar a água para um ponto de maior potencial energético, no caso expresso pela pressão atingida, medida por kPa ou mca (metros de coluna d'água)

A água flui naturalmente do ponto de maior potencial energético para o menor

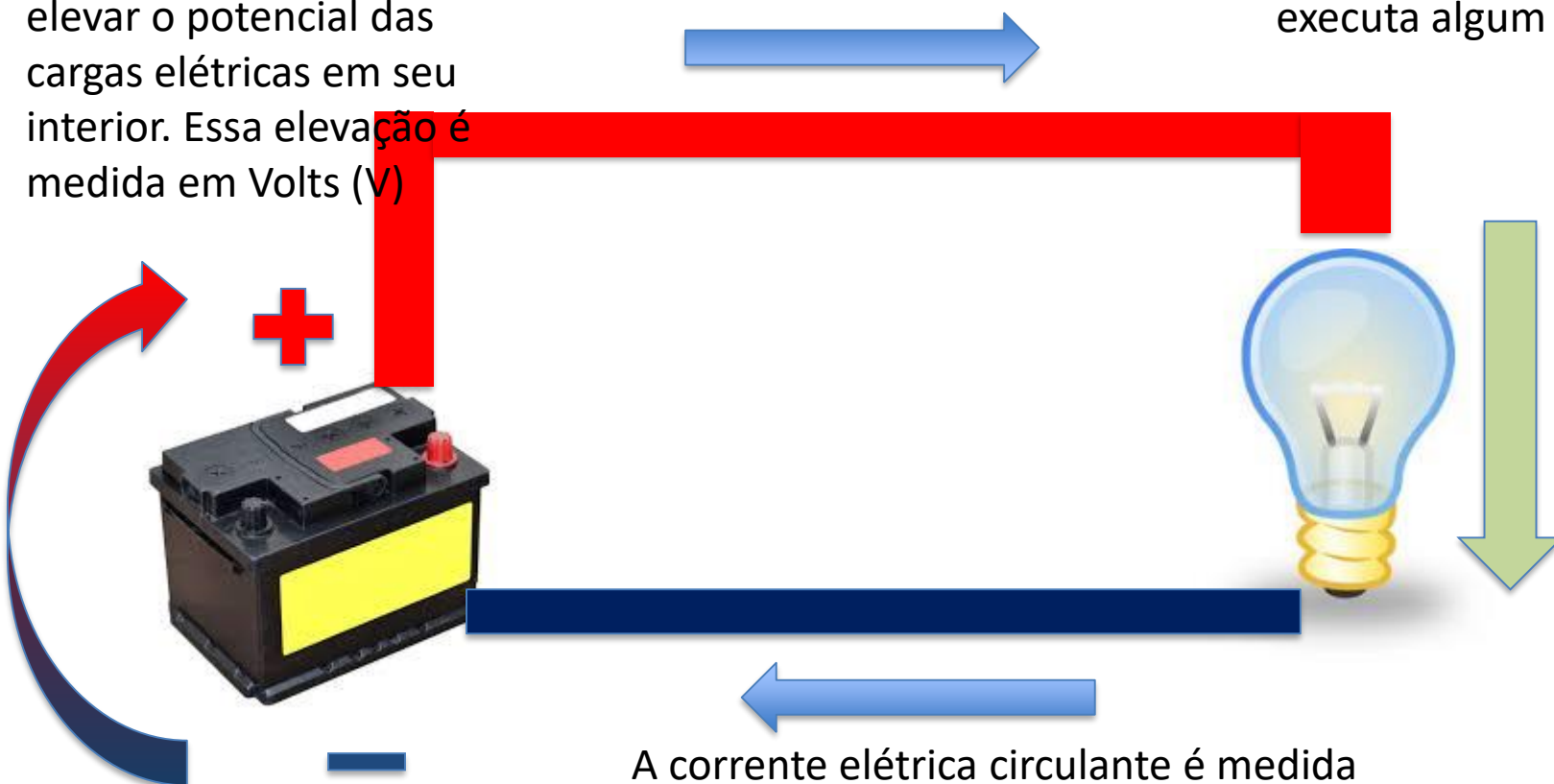


A vazão de água é medida em litros por segundo (l/s)

Circuito elétrico

Uma bateria elétrica usa algum tipo de energia para elevar o potencial das cargas elétricas em seu interior. Essa elevação é medida em Volts (V)

Uma **carga elétrica** usa a diferença de potencial fornecida para gerar uma corrente elétrica que executa algum trabalho

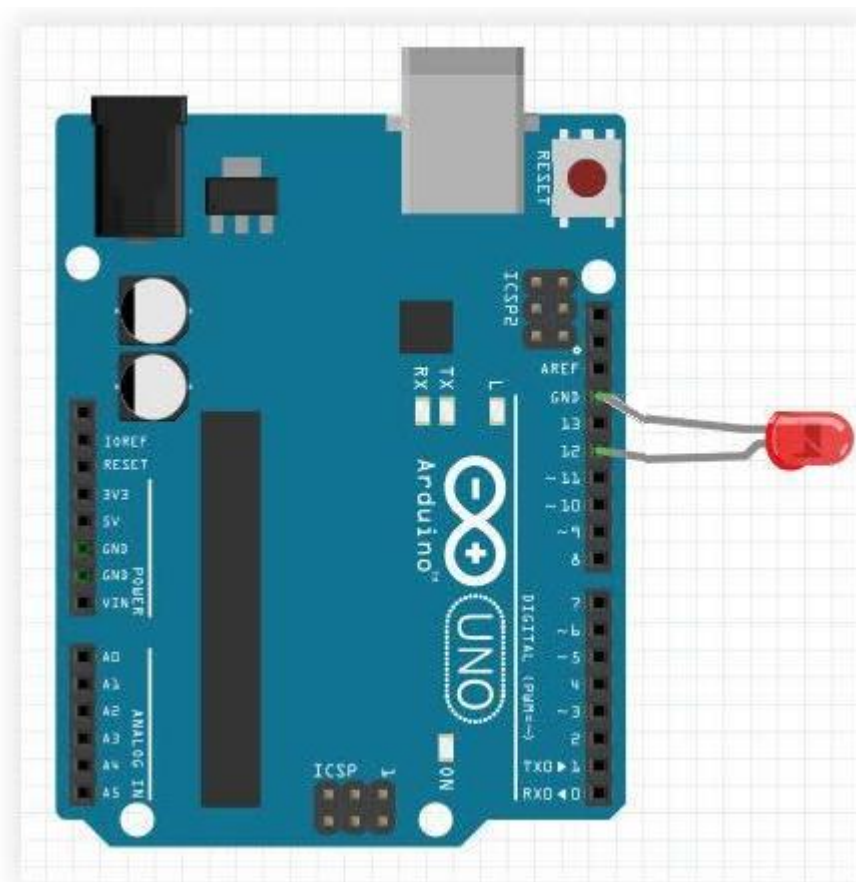


A corrente elétrica circulante é medida em ampères (A), ou em coulombs por segundo (C/s), onde C é a unidade de medida da carga elétrica

VOLTANDO AO ARDUINO

Agora que compreendemos o significado de tensão, corrente e circuito elétricos...

- ...entendemos por que um componente elétrico precisa possuir pelo menos dois terminais:
 - Um deles deve estar ligado ao maior potencial elétrico, e outro ao menor potencial
- No Arduino e demais placas de microcontroladores, há um ou mais pinos com o potencial elétrico constante, considerado como 0V. É frequentemente chamado de **terra** ou “**ground**” (GND).
- Quando a porta 12 está ligada, o LED recebe 5V por um terminal e 0V no outro, forçando a passagem de corrente, gerando luz.



I PROGRAMA QUE PISCA O LED

```
// LED no pino 12
int led = 12;
void setup() {
    // inicializa a porta do LED como saída digital.
    pinMode(led, OUTPUT);
}
void loop() {
    // liga a porta do LED
    digitalWrite(led, HIGH);
    delay(1000); // espera 1000 milissegundos
    // desliga a porta do LED
    digitalWrite(led, LOW);
    delay(1000); // espera 1000 milissegundos
}
```


PORTAS DIGITAIS COMO SAÍDA

Fazem o acionamento de dispositivos do tipo liga/desliga

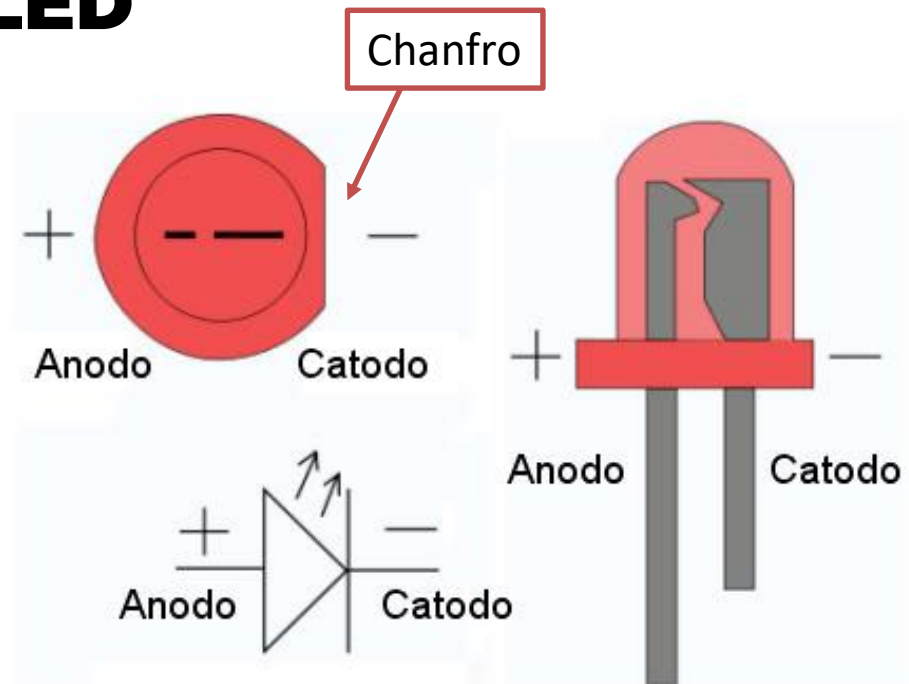
- As portas números 0 a 13 do Arduino permitem a leitura ou escrita de valores lógicos. Por isso são chamadas de GPIO (General-Purpose Input and Output)
 - Não usamos as portas 0 (RX0) e 1 (TX0) pois são destinadas para a comunicação serial – via cabo USB ou comunicação com shields, por exemplo
- São configuradas como saídas digitais na função **setup()** através da chamada:
 - `pinMode(numero, OUTPUT);`
- Podem acionar dispositivos que necessitem de uma informação digital (0 ou 1) para seu acionamento, como luz LED, relê para acionamento de equipamentos elétricos, drives de motores, etc.
- Os comandos para ligar ou desligar a porta são:
 - `digitalWrite(numero, HIGH)` : aciona a saída digital
 - `digitalWrite(numero, LOW)` : desliga a saída digital

I USANDO O LED COM O ARDUINO

- Na saída porta 13 há um LED conectado, que acende assim que a porta é acionada
- Se você tentou montar o circuito do slide 23 (apenas o LED conectado à porta 12 do Arduino), talvez não tenha tido sucesso.
 - Isso porque o LED é um componente que possui polaridade, ou seja, há uma perna específica para o terminal de maior potencial (**polo positivo**) e outro para o menor potencial (**polo negativo**)

■ INSTALAÇÃO DO LED

- Seus terminais são:
 - **Anodo:** correspondente ao polo positivo, geralmente ligamos à porta controladora
 - **Catodo:** correspondente ao polo negativo, geralmente ligamos ao GND (0V)



- Há três formas de identificar os terminais:
 - **Pelo tamanho da perna:** a perna maior corresponde ao anodo (polo +)
 - **Pelo chanfro:** a perna do lado do chanfro corresponde ao catodo (às vezes é difícil de identificar)
 - **Pela placa interna:** a perna ligada à placa interna de maior tamanho corresponde ao catodo (polo -)

I USO DO RESISTOR

- No Arduino Uno, o LED recebe cerca de 5V quando ligado. Porém, a tensão correta de funcionamento da maior parte dos LEDs que usamos em circuitos é de 2V a 3,5V.
- Para evitar que o LED queime a longo prazo, precisamos usar um resistor para limitar a tensão.
- A fórmula que usamos para calcular a resistência correta é:
$$R = \frac{V_{in} - V_{LED}}{I_{LED}}$$

I CÁLCULO DO RESISTOR

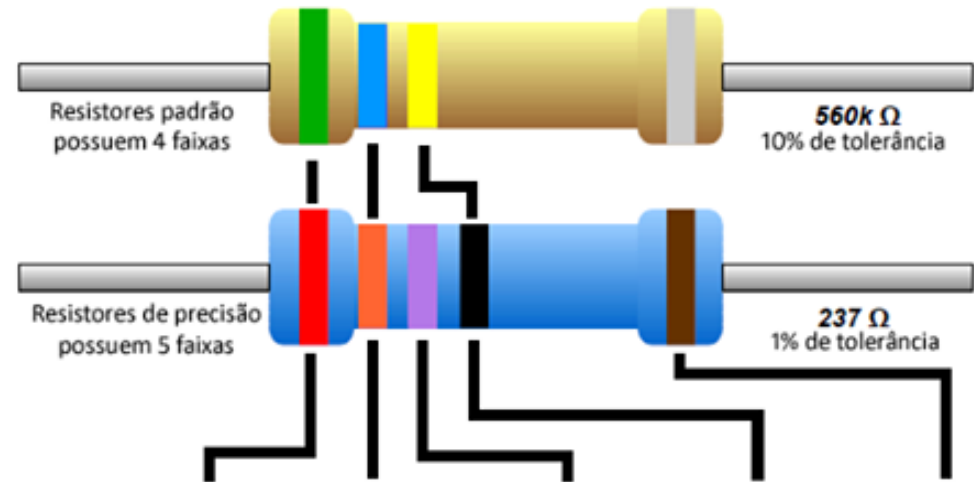
- Símbolos usados no cálculo:
 - V_{in} - tensão de entrada (5V par o Uno)
 - V_{LED} - tensão de funcionamento do LED
 - I_{LED} - corrente de funcionamento do LED, medida em A (ampères)
- Exemplos de LED:
 - LED pequeno (1,5V, 15 mA): $R = 233\Omega$
 - LED convencional (2V, 20 mA) $R = 150\Omega$
 - LED alto brilho (3,3V, 30 mA) $R = 56\Omega$

Lendo o valor de resistores pelo código de cores

- Verificar através da tabela de cores o algarismo correspondente à cor do primeiro anel, que será o primeiro dígito do valor da resistência.
- Verificar através da tabela de cores o algarismo correspondente à cor do segundo anel, que será o segundo dígito do valor da resistência.
- Determinar o valor para multiplicar o número formado pelos itens 1 e 2 pela cor do anel multiplicador (3ª ou 4ª faixa)
- Verificar a porcentagem de tolerância do valor nominal da resistência pela cor do último anel.
- Ex: cores marrom (1), preto (0), laranja (x1k) e dourado: 10k ohms de resistência com tolerância de 5%

Código de Cores

A extremidade com mais faixas deve apontar para a esquerda

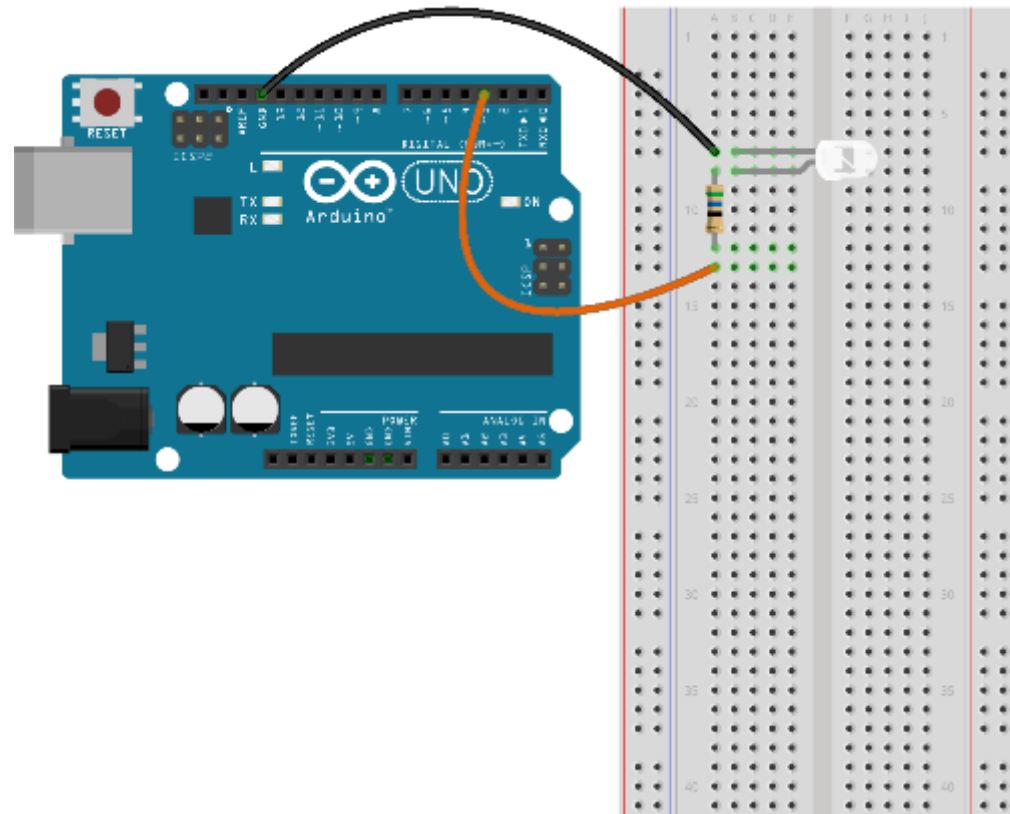


Cor	1ª Faixa	2ª Faixa	3ª Faixa	Multiplicador	Tolerância
Preto	0	0	0	x 1 Ω	
Marrom	1	1	1	x 10 Ω	+/- 1%
Vermelho	2	2	2	x 100 Ω	+/- 2%
Laranja	3	3	3	x 1K Ω	
Amarelo	4	4	4	x 10K Ω	
Verde	5	5	5	x 100K Ω	+/- .5%
Azul	6	6	6	x 1M Ω	+/- .25%
Violeta	7	7	7	x 10M Ω	+/- .1%
Cinza	8	8	8		+/- .05%
Branco	9	9	9		
Dourado				x .1 Ω	+/- 5%
Prateado				x .01 Ω	+/- 10%

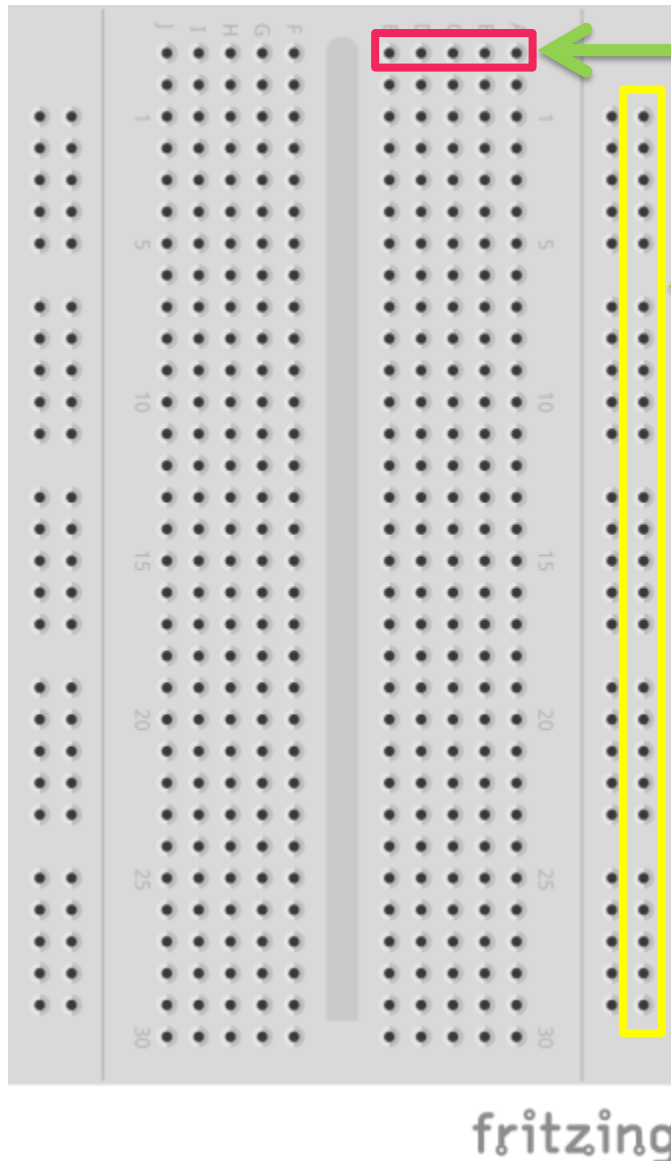
EXPERIMENTO 1: PISCAR O LED

- Materiais:
 - 1 Arduino Uno com cabo serial;
 - 1 Protoboard;
 - 1 LED alto brilho;
 - 1 Resistor de 56 ohm (ver cores na tabela)
 - Observação: o resistor não tem polaridade

- Montagem:



PROTOBOARD



- Faz a conexão elétrica entre componentes
- Cada fileira de 5 orifícios na coluna central forma uma trilha conectada, e isolada das demais trilhas
- Cada coluna de orifícios nas trilhas laterais forma uma trilha de alimentação, e está toda conectada.
 - Geralmente elas são ligadas à tensão de alimentação (5V) ou ao terra (GND ou 0V)
- Dois terminais de componentes plugados à mesma trilha estão eletricamente conectados
 - Nunca podemos conectar dois terminais de um mesmo dispositivo na mesma trilha, pois assim eles estarão em curto-circuito!

EXPERIMENTO 1: PROGRAMAÇÃO

- Usar o programa do slide 24, adaptando a porta do LED
- Modificar o programa para que, a cada vez que o LED liga ou desliga, ele mande uma mensagem para a porta serial
- Mudar a temporização para ligar ou desligar a cada 2 segundos

EXPERIMENTO 2: Temporização

- Mantenha o mesmo circuito
- Modifique o programa do slide 24 para que a temporização do LED obedeça a duas variáveis, declaradas no início do programa, como no exemplo abaixo:

```
void loop() {  
    digitalWrite(led, HIGH);  
    delay(tempo_on); // declarar como int  
    digitalWrite(led, LOW);  
    delay(tempo_off); // declarar como int  
}
```

- Teste o programa para as seguintes combinações de temporizações:
 - tempo_on=80 e tempo_off=20
 - tempo_on=50 e tempo_off=50
 - tempo_on=20 e tempo_off=80
 - tempo_on=8 e tempo_off=2
 - tempo_on=5 e tempo_off=5
 - tempo_on=1 e tempo_off=9

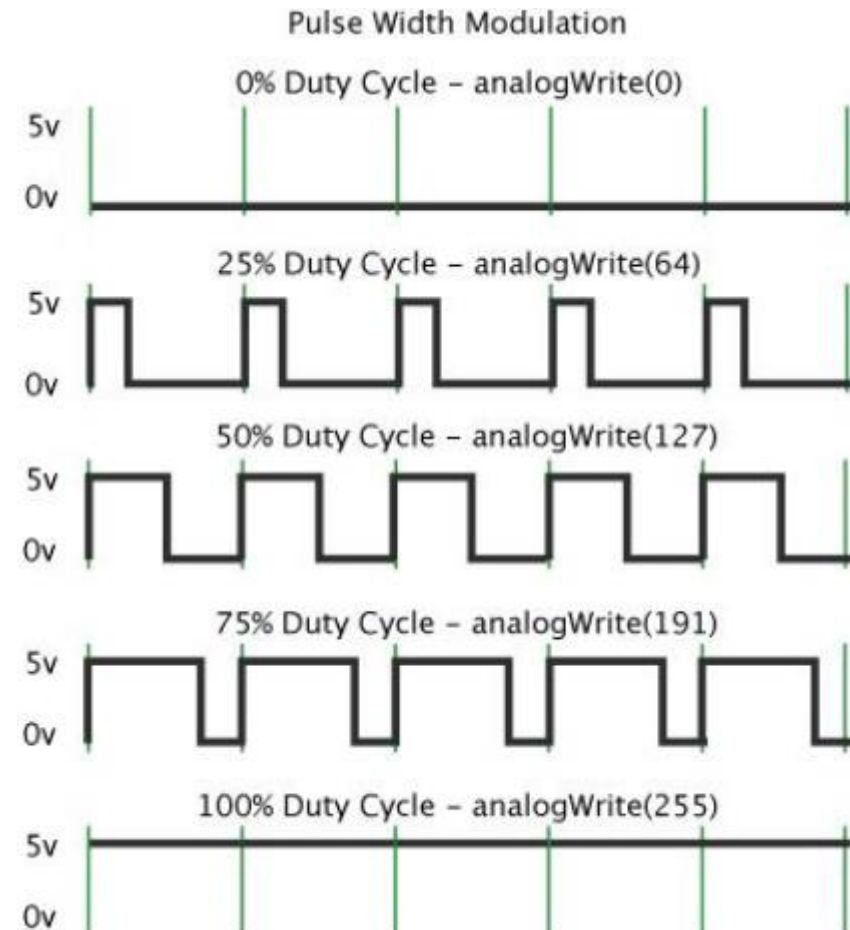
SAÍDA PWM

- No último experimento, permitimos diferentes temporizações para o LED aceso e apagado, mudando a nossa percepção sobre o seu brilho:
 - Se o tempo ligado for maior do que o tempo desligado, o brilho fica mais forte
 - Se o tempo ligado for menor do que o tempo desligado, o brilho fica mais fraco
- A essa forma de controle dos dispositivos chamamos de PWM – Pulse-Width Modulation, e pode ser usada para controlar o brilho do LED, a velocidade de motores elétricos, o posicionamento de servo-motores, além de outros dispositivos
 - A frequência do PWM equivale ao número de vezes em que a saída alterna entre ligada e desligada a cada segundo. Assim, é calculada como: $\frac{1}{t_{on}+t_{off}}$
- O Arduino e outros microcontroladores possuem saídas digitais que podem ser programadas como saídas PWM.
 - No caso do Arduino, as portas digitais marcadas com um ~ (til) possuem a capacidade de servirem de saída do tipo PWM

SAÍDAS ANALÓGICAS (PWM)

Fazem o acionamento de dispositivos que possam ser controlados por PWM

- Uma saída PWM automaticamente alterna períodos em que ela esta ligada (HIGH) e em que está desligada (LOW). A proporção do tempo em que a saída está ligada em relação ao tempo total do ciclo é chamad de Duty Cycle, e varia de 0 a 100%
- Exemplos: motor de passo, iluminação LED ou acionador dimerizável de lâmpada, LED RGB (seleciona a cor do LED)
- O valor **inteiro** a ser escrito deve estar entre 0 e 255, e deve ser calculado a partir do Duty Cycle: $v = \frac{\text{DutyCycle}}{100} \times 255$
- Para escrever um valor de 0 a 255 na saída PWM, usamos a forma:
`analogWrite(numero, valor);`



■ EXPERIMENTO 3: Saída PWM

- Calcule as frequências PWM empregadas no experimento 2, bem como os valores de Duty Cycle empregados
- Mantendo o mesmo circuito, modifique o programa do slide 24 para que o brilho do LED seja controlado pela saída PWM do Arduino.

Exemplo:

```
void loop() {  
    analogWrite(led, valor);  
}
```

- Calcule e aplique o valor da saída PWM para os seguintes níveis de brilho do LED:
 - Brilho de 10%
 - Brilho de 50%
 - Brilho de 85%

I REFERÊNCIAS



1. Min-Woo Ryu et al. **Survey on Internet of Things: Towards Case Study**. The Smart Computing Review, v. 2(3), 2012.
2. Gartner. **Gartner IT Glossary**. url: <http://www.gartner.com/it-glossary/cloud-computing>
Acesso em 17/01/2016
3. Arduino Team. **Arduino Reference: Digital Write**. url: <https://www.arduino.cc/reference/en/language/functions/digital-io/digitalwrite/>. Acesso em 10/01/2020.



Copyright © 2020 Prof. Antonio Henrique Pinto Selvatici

Todos direitos reservados. Reprodução ou divulgação total ou parcial deste documento é expressamente proibido sem o consentimento formal, por escrito, do Professor (autor).