



**Paca (*Cuniculus*)** é um roedor herbívoro das Américas do Sul e Central.

# PMR 3202 – Introdução ao projeto de Sistemas Mecânicos

## Aula 04 - Arduino e PACA

(Plataforma de acionamento e comunicação baseada em arduino)

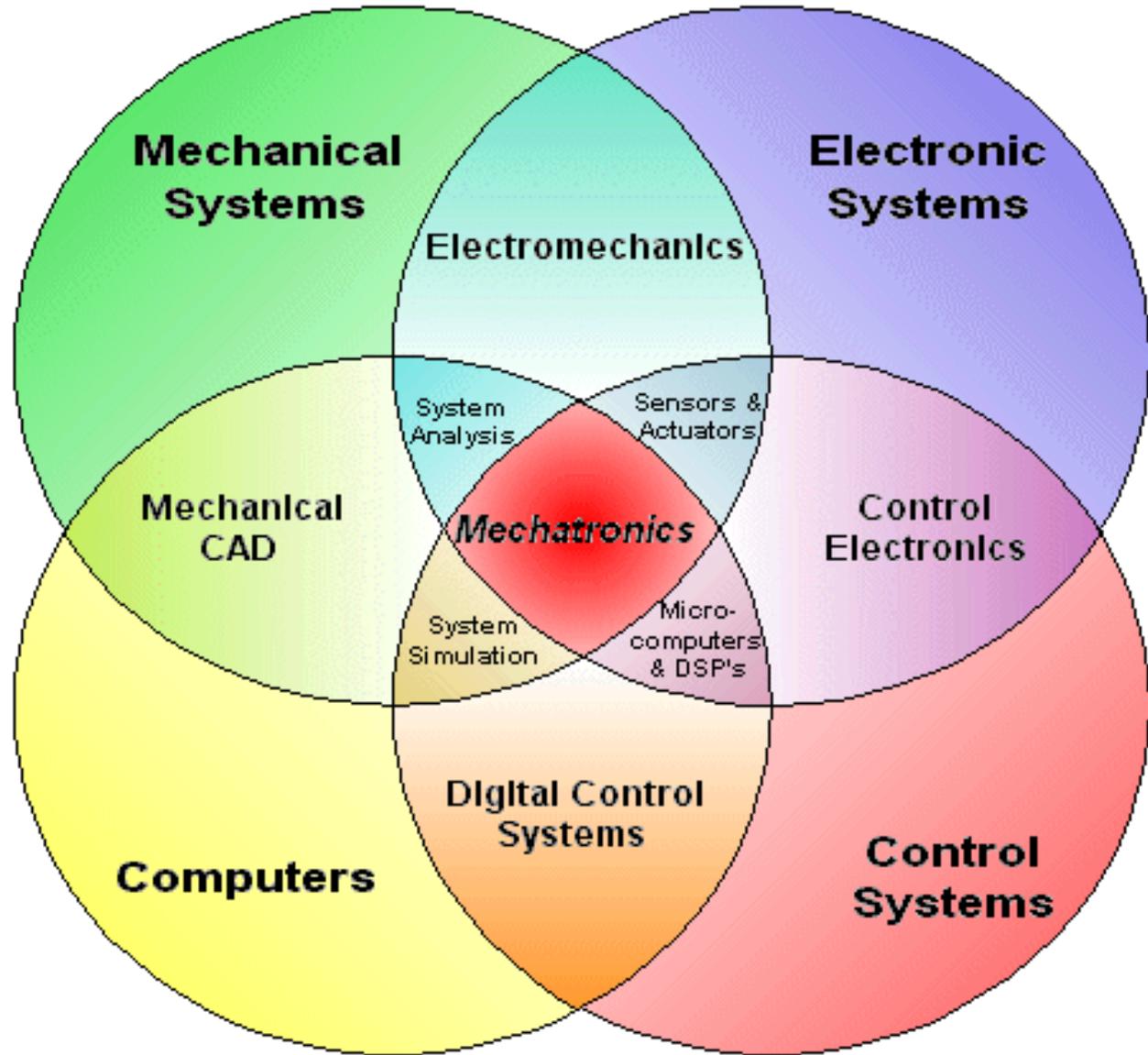
*Prof. Dr. Rafael Traldi Moura*



Um sistema é um conjunto de elementos inter-relacionados no qual o comportamento de cada elemento afeta o comportamento dos demais e do sistema como um todo.

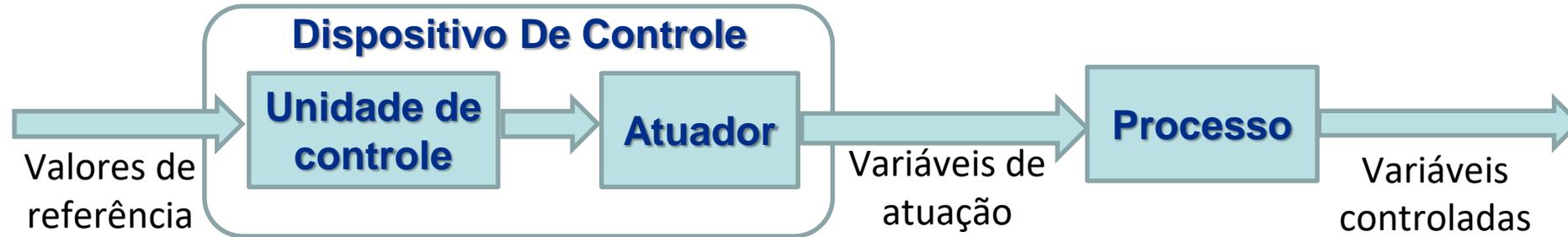
A mecatrônica é a intersecção entre as engenharias mecânica, eletrônica, de computação e de controle.

Entretanto, especialmente na fase de projeto, um sistema mecatrônico é mais do que a simples soma dos projetos mecânico, eletrônico de computação e de controle.

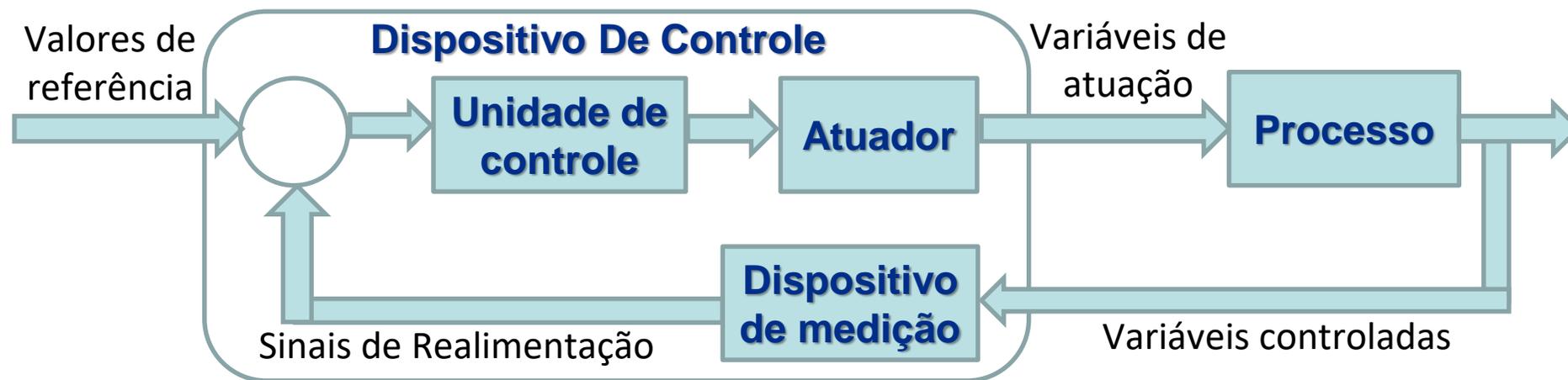




- Sistema mecatrônico com controle de malha aberta:

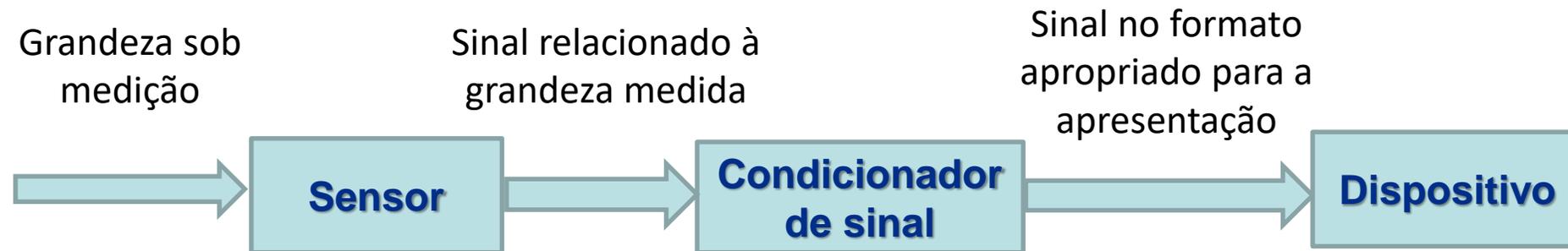


- Sistema mecatrônico com controle de malha fechada:





- O dispositivo de medição pode ser modelado por:

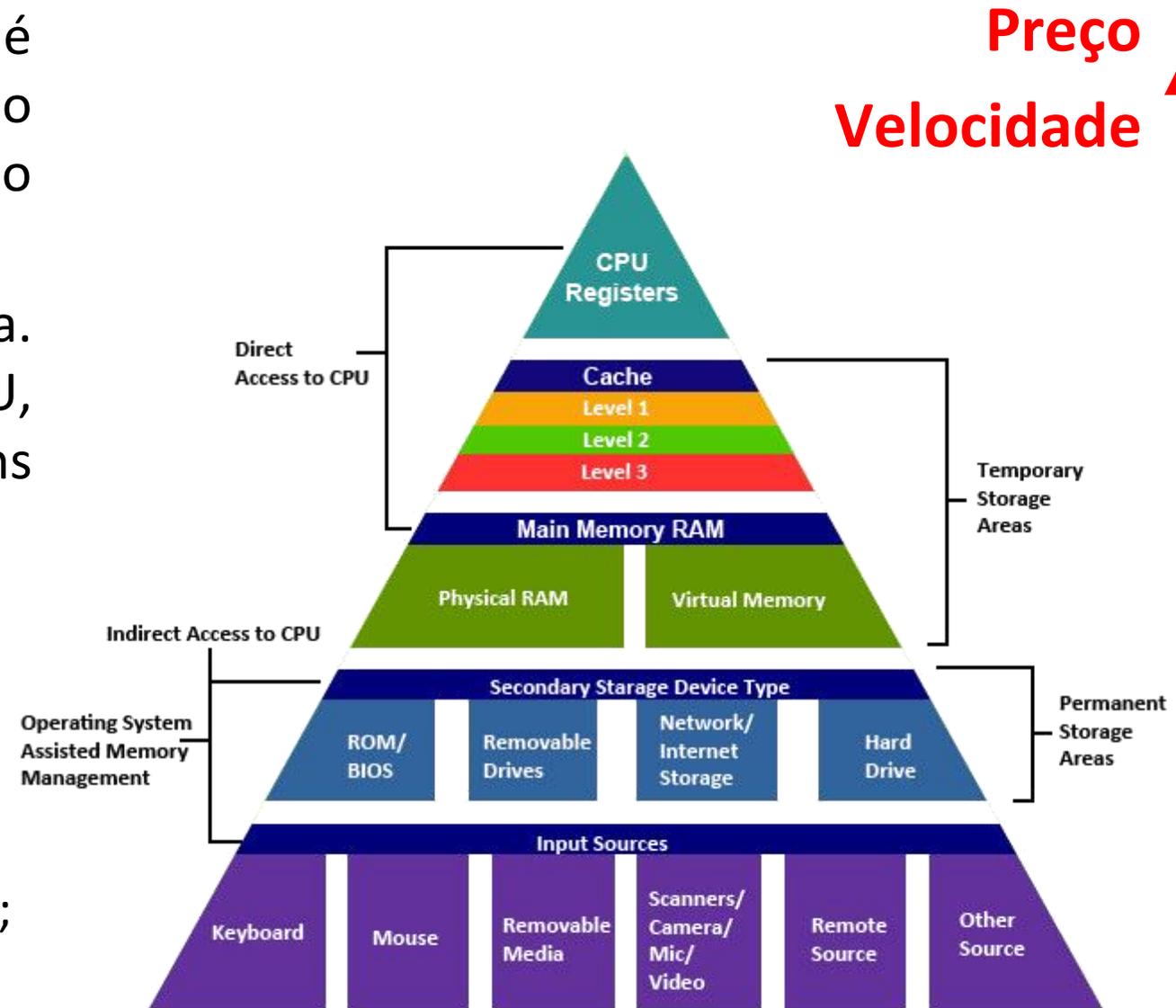




A memória é um componente cuja função é armazenar informações que serão manipuladas pelo Sistema quando necessário.

Existem vários tipos de memória. Geralmente, quanto mais *próxima* da ALU, mais rápida e mais cara é a memória. Alguns tipos são:

- RAM – Memória de acesso aleatório;
- ROM – Memória de apenas leitura;
- PROM – ROM programável;
- EPROM – PROM apagável;
- EEPROM – PROM eletricamente apagável;



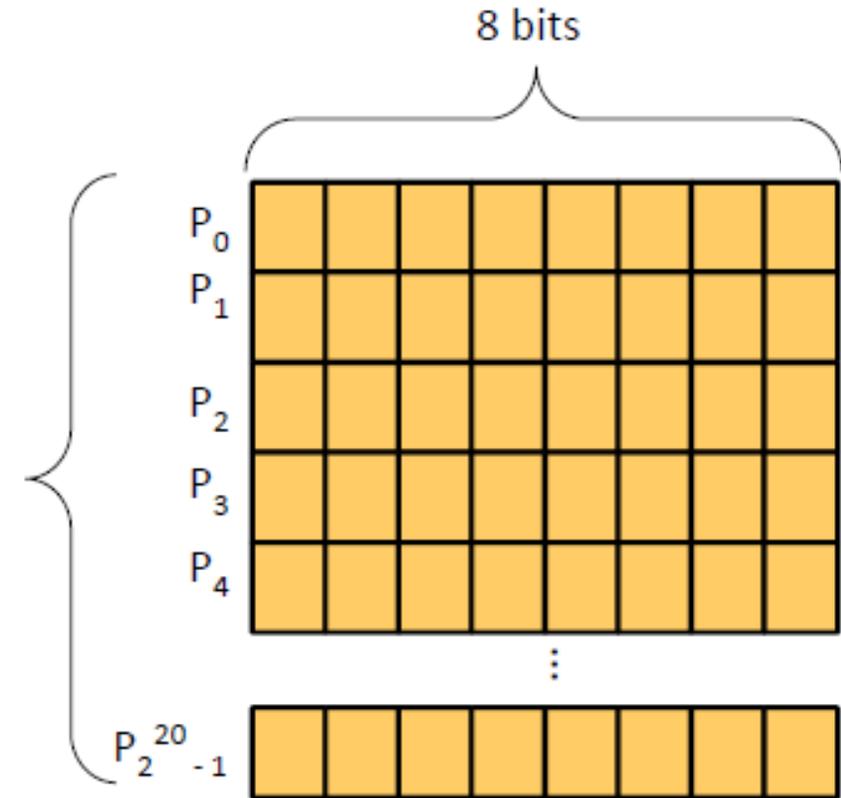
# Memória – Arranjo estrutural



- O arranjo físico do armazenamento de dados na memória está ilustrado na figura ao lado.
- Cada *espaço* para armazenamento recebe um valor de identificação, como o número da sua casa serve para identificar a posição da rua em que você mora. Esse valor de identificação é conhecido como **endereço**.
- Assim como o maior número que se pode representar com 2 algarismos é 99, repare que a largura da matriz ao lado, dada em bits, determina o maior valor a ser armazenado, sendo neste caso  $2^8-1=255$ .

EPROM de 8 Mbits  
(ou 1 MB)

1M Palavras



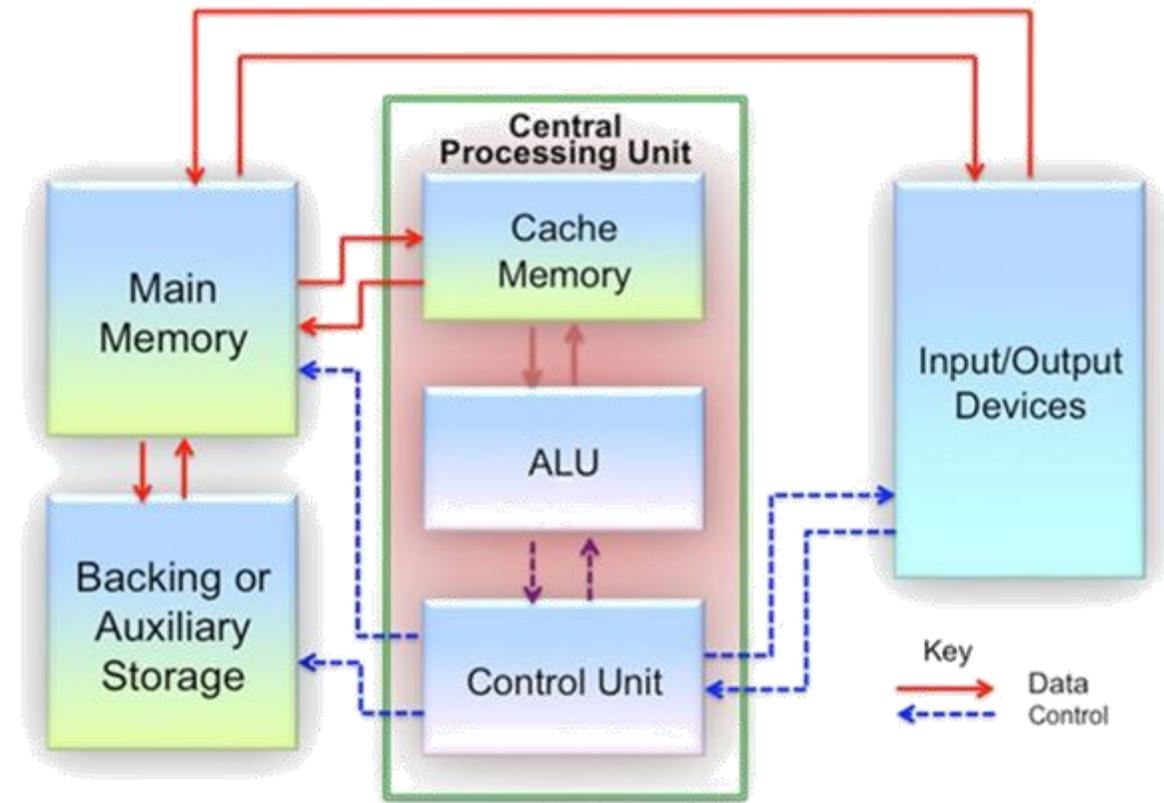


- É muito penoso trabalhar com os valores armazenados na memória em seu espelho físico, ou seja, na base binária. Desta forma, para facilitar, trabalha-se na base hexadecimal agrupando o número binário em grupos de 4. Desta forma, um número com 8 bits é representado por um hexadecimal de 2 algarismos.
- Existem diferentes tipos de variável, com cada tipo ocupando um espaço diferente na memória. Os mais usados no Arduino são:
  - Boolean (8 bits): verdadeiro ou falso;
  - Byte (8 bits): inteiro de 0 a 255;
  - Char (8 bits): inteiro de -128 a 127;
  - Int (16 bits): inteiro de -32768 a 32767;
  - Unsigned int (16 bits): inteiro de 0 a 65535;
  - Long (32 bits): inteiro de -2,147,483,648 to 2,147,483,647;
  - Float (32 bits): número real de -3.4028235E38 to 3.4028235E38. Não suportado nativamente pelo MPU do Arduino UNO. Evite se puder!

| DECIMAL | HEXADECIMAL | BINARIO |
|---------|-------------|---------|
| 0       | 0           | 0       |
| 1       | 1           | 1       |
| 2       | 2           | 10      |
| 3       | 3           | 11      |
| 4       | 4           | 100     |
| 5       | 5           | 101     |
| 6       | 6           | 110     |
| 7       | 7           | 111     |
| 8       | 8           | 1000    |
| 9       | 9           | 1001    |
| 10      | A           | 1010    |
| 11      | B           | 1011    |
| 12      | C           | 1100    |
| 13      | D           | 1101    |
| 14      | E           | 1110    |
| 15      | F           | 1111    |

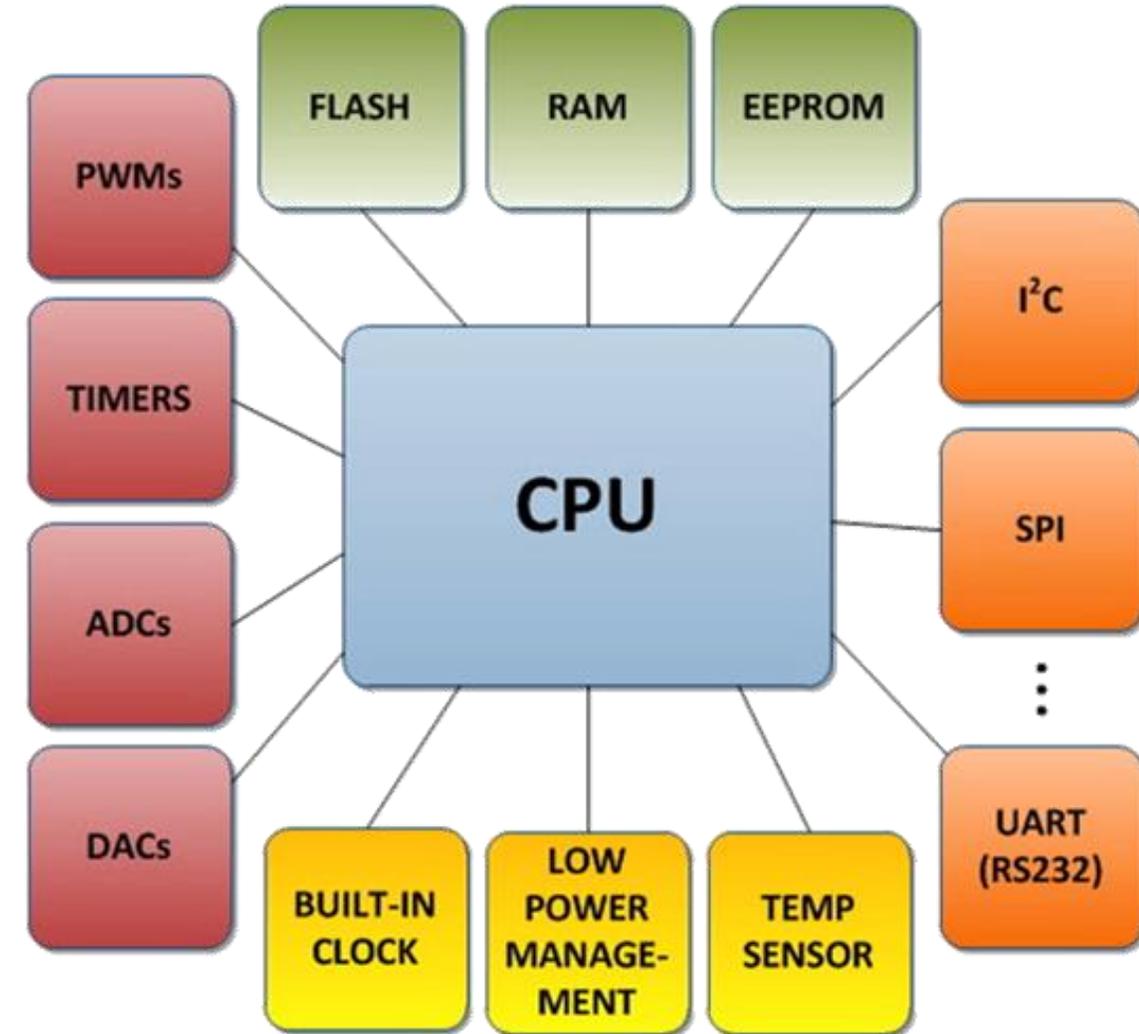


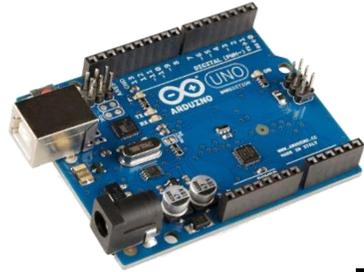
- Um microprocessador, MPU, é um CI (circuito integrado) composto por uma unidade de controle, uma ALU (Arithmetic Logic Unit) e memória (registradores e cache).
- A ALU faz operações aritméticas e lógicas (é igual? É maior? Etc...). Esta deve ser compatível com o tamanho da célula de memória, por exemplo, de 8, 16, 32, etc bits.





- Um microcontrolador, MCU, é um SoC (System-on-a-Chip), pois é um CI (circuito integrado) que contém uma CPU, memória, entradas e saídas, protocolos de comunicação, conversores analógico digital e digital analógico, timers e PWMs.





## Microcontrolador



Arduino uno – 42mA e 7V = 0.29Watt



## Microprocessador



2+?=4

Raspberry Pi 3 – 500mA a 1 A e 5V = 2,5 a 5Watt

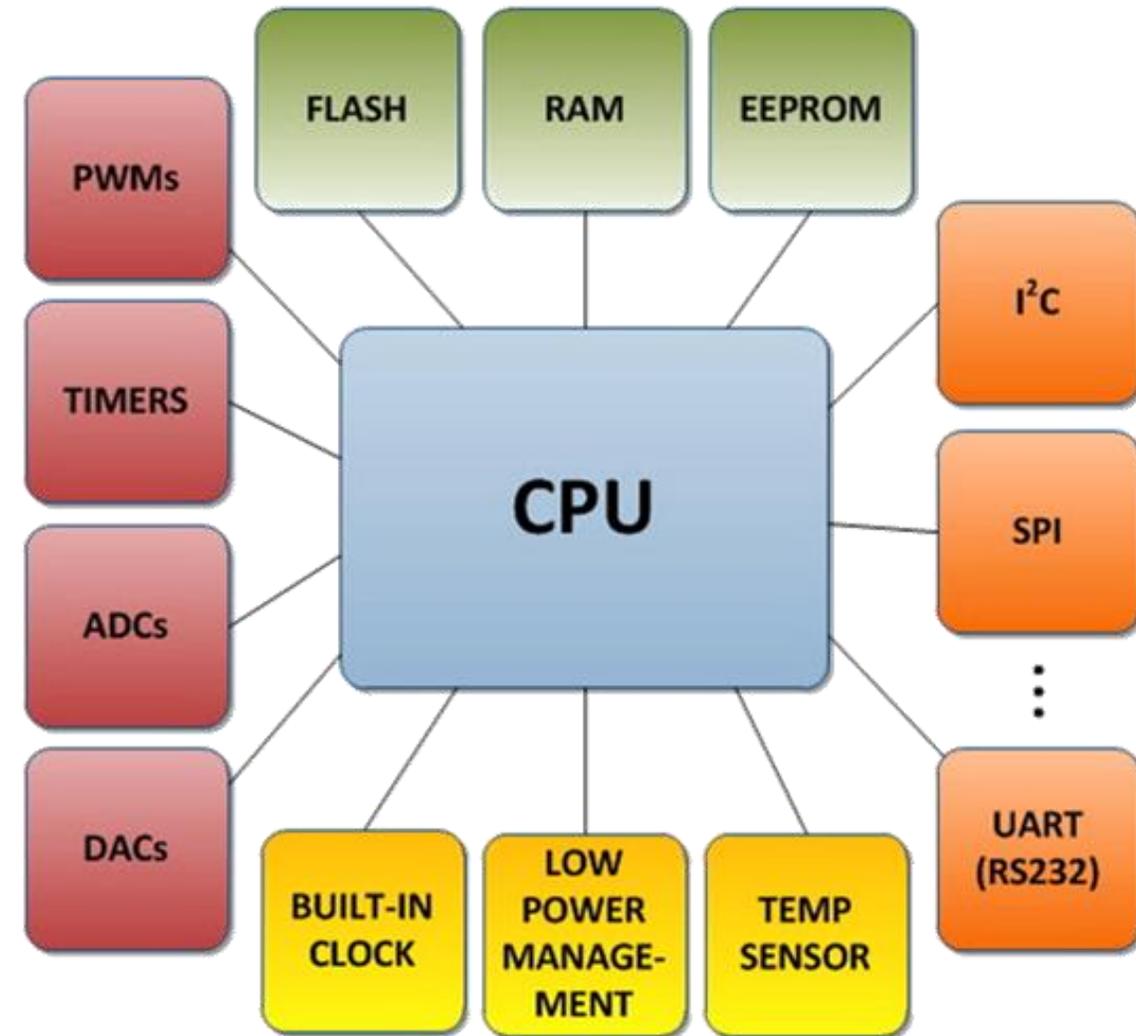


- O Arduino pode ser descrito como um sistema de desenvolvimento para microcontroladores;



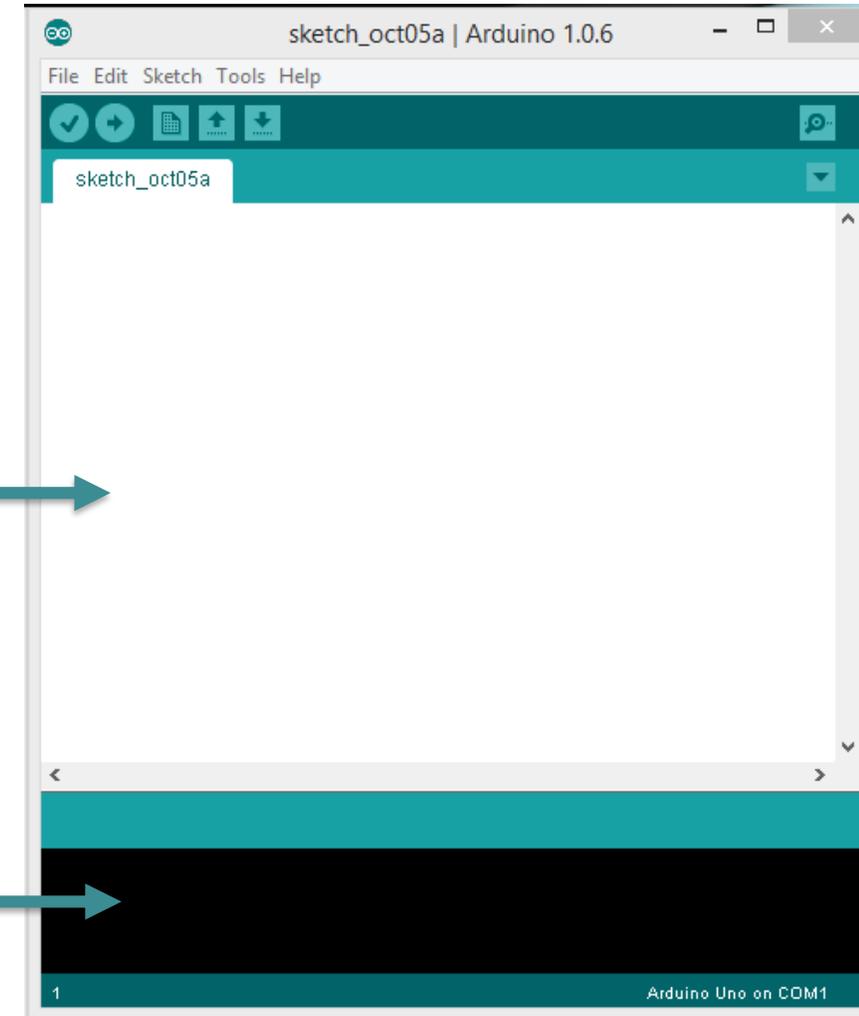
# Memórias do Arduino!

- Flash – memória que guarda o sketch. 32k bytes no Arduino uno;
- SRAM – memória na qual o sketch cria e manipula as variáveis durante execução. 2k bytes no Arduino uno;
- EEPROM – memória na qual pode-se armazenar informações a longo prazo. 1k bytes no Arduino uno;





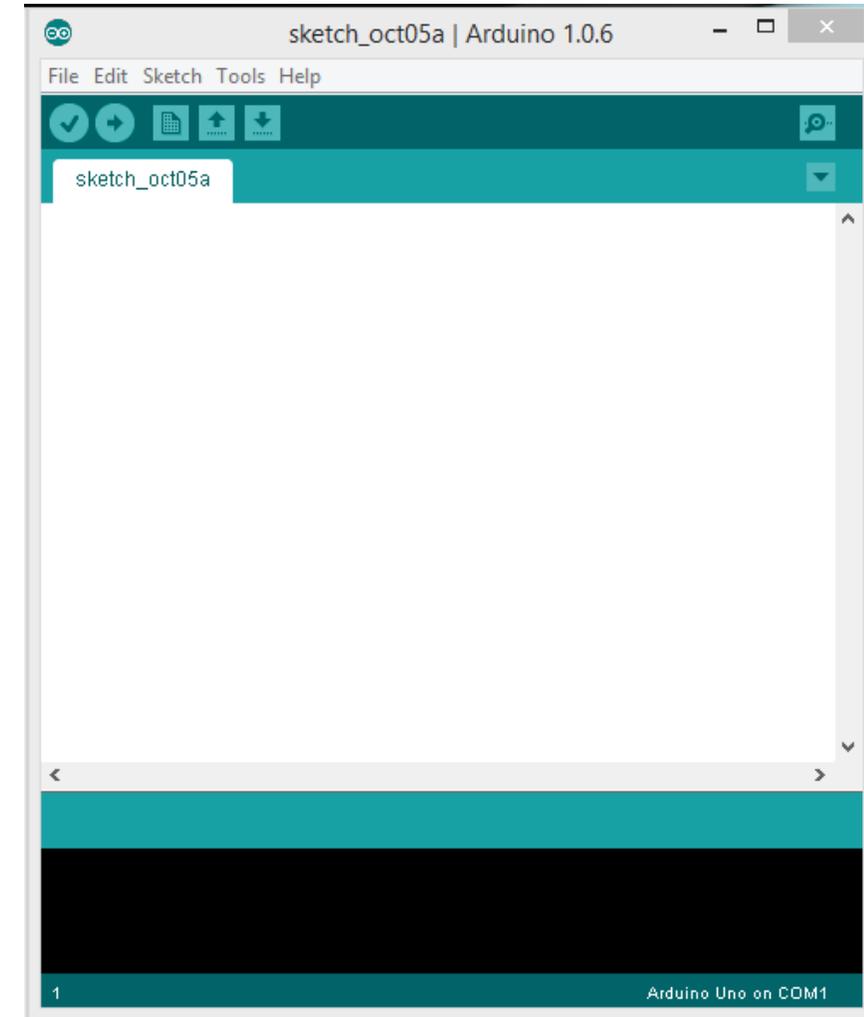
- Uma das vantagens que tornou o Arduino popular é sua IDE (Integrated Development Environment) e deve ser baixada do site: <http://arduino.cc/en/Main/Software> ;
- A IDE possui diversos elementos. O maior e mais visível é a área de **editor de texto**, parte branca, no qual o código é digitado.
- Logo abaixo, encontra-se o console, em preto. No console serão mostradas **mensagens de erro de compilação e durante a execução**.





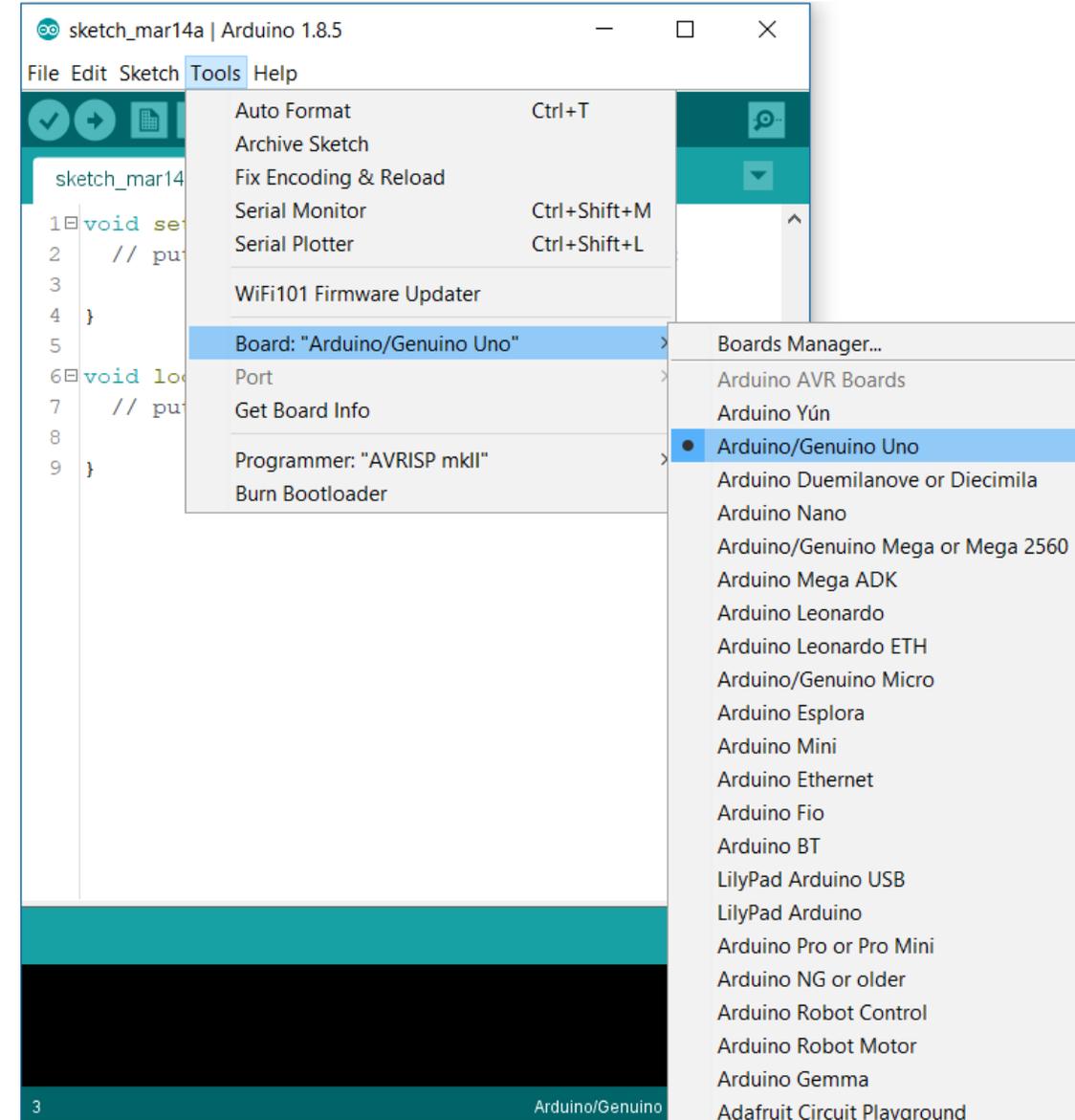
As funções dos botões de comando são:

-  verify;
-  upload
-  new;
-  open;
-  save;
-  serial monitor.



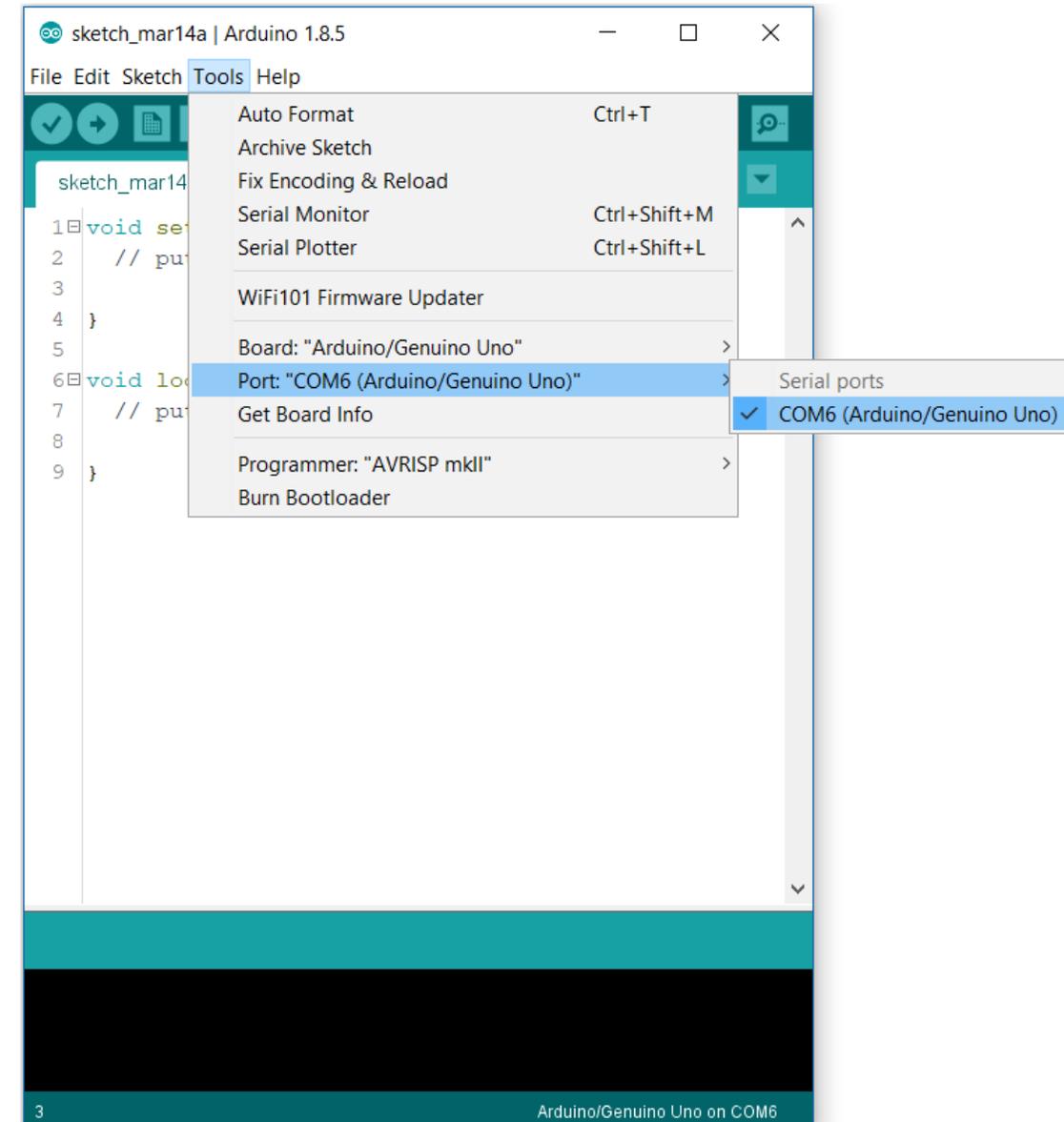


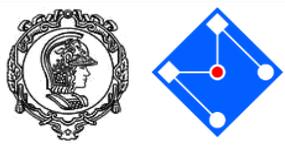
- Para que a IDE consiga transformar seu código em um conjunto de instruções chamada *firmware* corretamente, a mesma precisa saber qual microcontrolador está sendo utilizado;
- Para selecionar o microcontrolador, devemos marcar o modelo correto de Arduino.
- Clique em *Tools*, depois em *board* e selecione *Arduino/Genuino Uno*.





- Como você pode ter conectado mais de um Arduino ao seu computador, temos que informar a IDE qual a porta de comunicação COM na qual está o Arduino Uno;
- Clique em *Tools*, depois em *port* e selecione a COM que estiver indicado o *Arduino/Genuino Uno*.





- O código de um programa em Arduino é chamado de sketch. A estrutura básica de um sketch é dividida em 3 partes: o escopo de variáveis, a função **setup()** e a função **loop()**;
- Escopo de variáveis: É a parte na qual são declaradas as **variáveis globais** do sketch, acessíveis de qualquer função do seu código, assim como as bibliotecas utilizadas. Os tipos de dados comumente utilizados são: byte, int, float, double, boolean e char.



```
sketch_mar14a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_mar14a
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

3 Arduino/Genuino Uno on COM6



- Função **setup()**: é chamada uma vez apenas na execução do programa, normalmente no início da execução. É onde se especifica, por exemplo, se um pino é de entrada ou saída e são inicializadas algumas variáveis.
- **loop()**: a função será executada em loop até que o programa seja terminado.



```
sketch_mar14a | Arduino 1.8.5
File Edit Sketch Tools Help
sketch_mar14a
1 void setup() {
2   // put your setup code here, to run once:
3
4 }
5
6 void loop() {
7   // put your main code here, to run repeatedly:
8
9 }
```

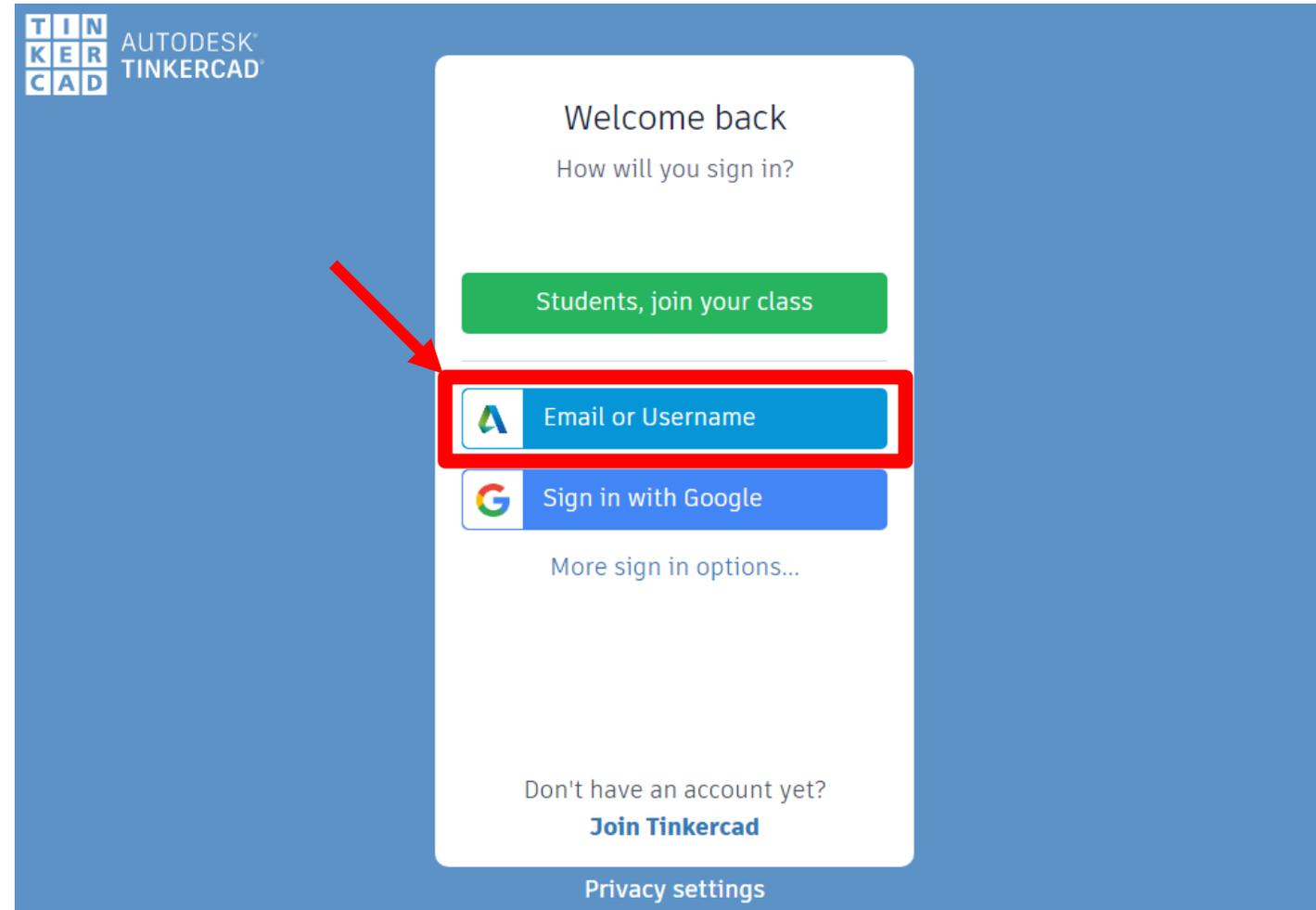
3 Arduino/Genuino Uno on COM6

Entre em <https://www.tinkercad.com/> e clique em **Sign in**, conforme figura ao lado.





Clique em **Email or Username**, conforme figura ao lado.





Caso já tenha feito uma conta, faça o login normalmente.

Caso não tenha criado uma conta antes, clique em **Create Account**, conforme figura ao lado. Então refaça os passos anteriores e faça o Login.

Sign in



Email or Username

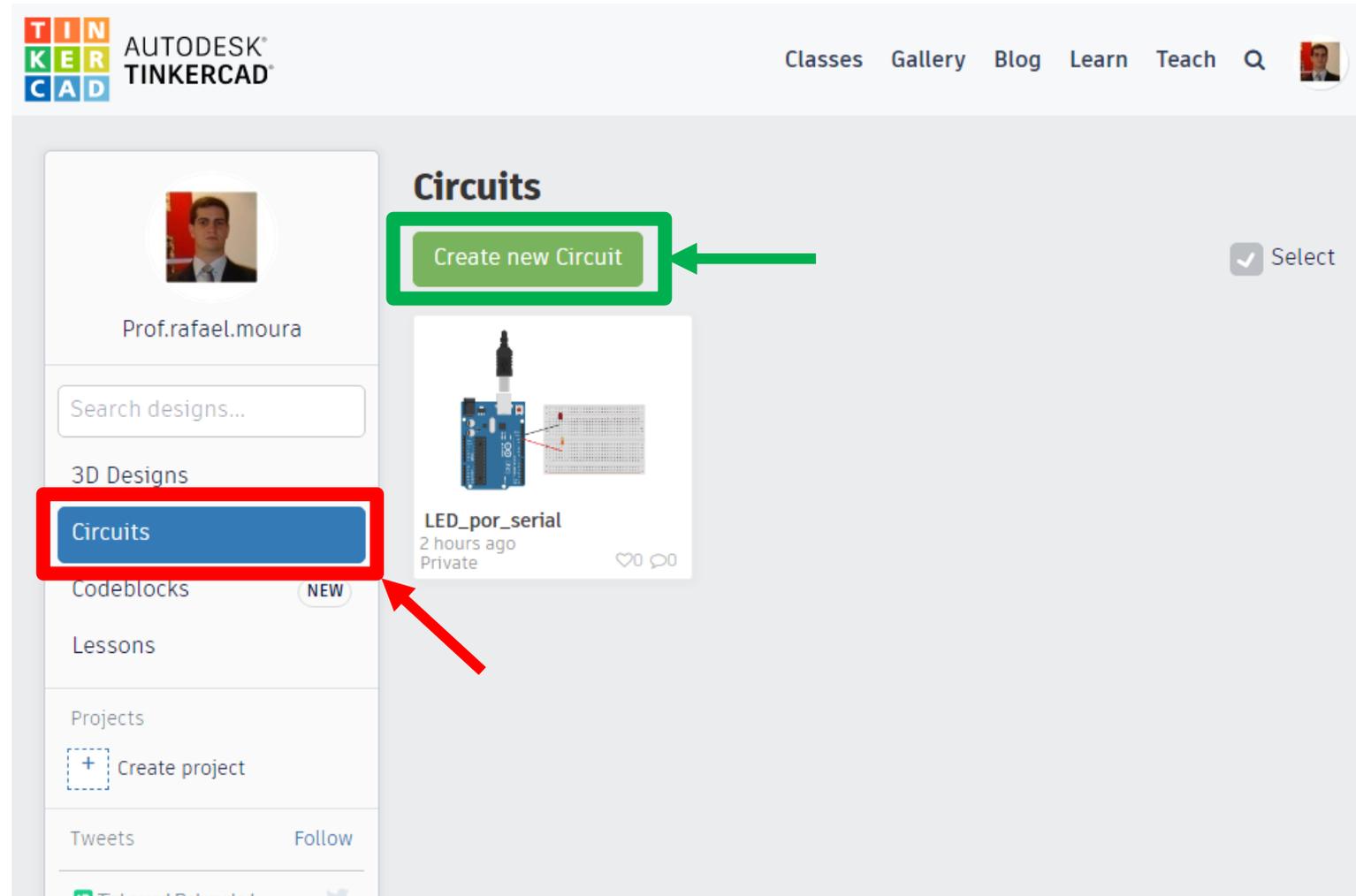
moura.gmsie@usp.br

NEXT

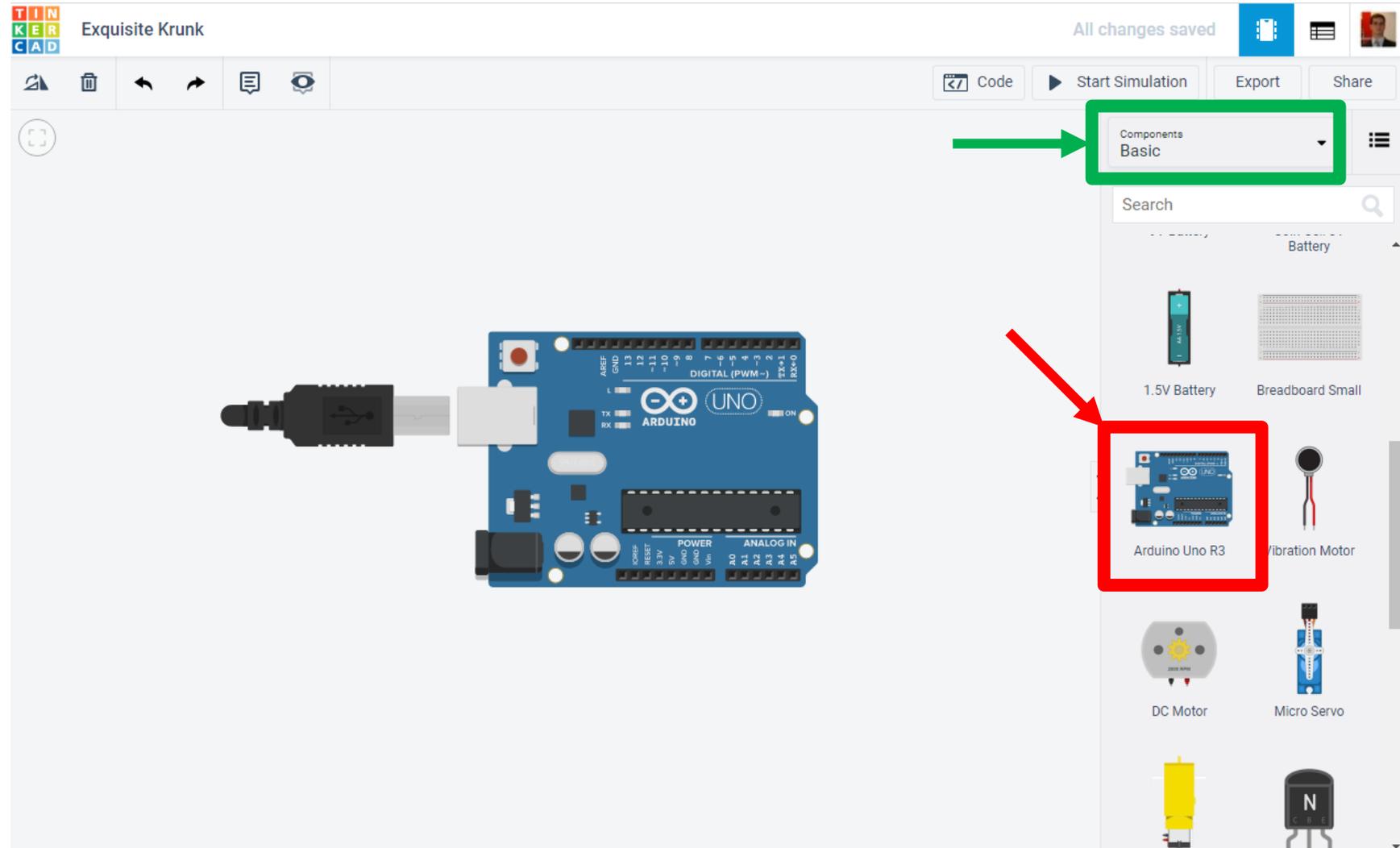
OR SIGN IN USING SOCIAL PROVIDERS

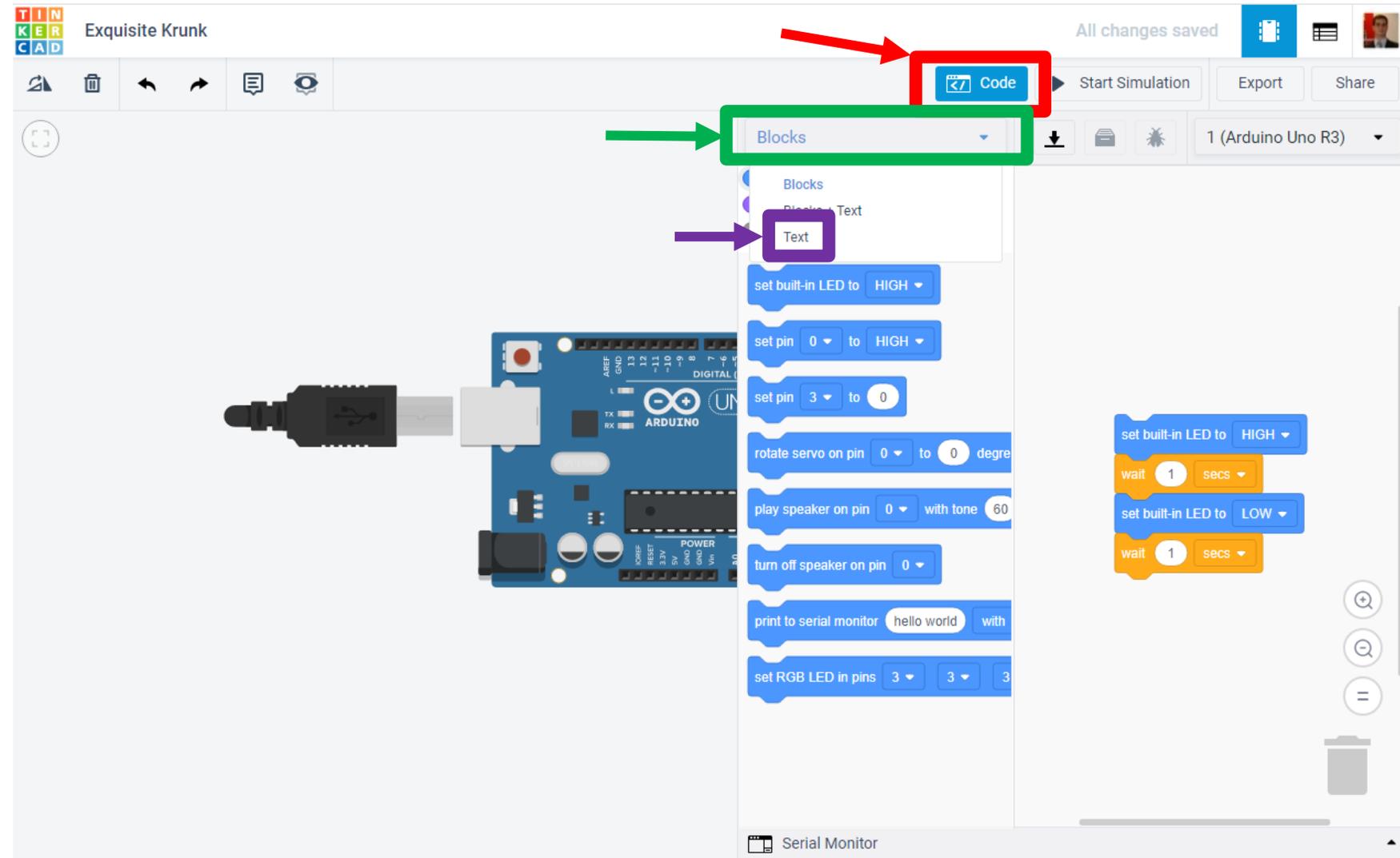
NEW TO AUTODESK? **CREATE ACCOUNT**

Uma vez logado, clicar em **Circuits** e em **Create new Circuit**.



Arraste um **Arduino Uno R3** dos itens de **Components Basics** para a área na esquerda.





Clique em **Code**, depois em **Blocks** e selecione **Text**.  
Surgirá um pop-up. Nele, selecione **Continue**.



# Exercício 01 – Comunicação Console – Sketch\_01

Digite o código e clique em **Start Simulation**.

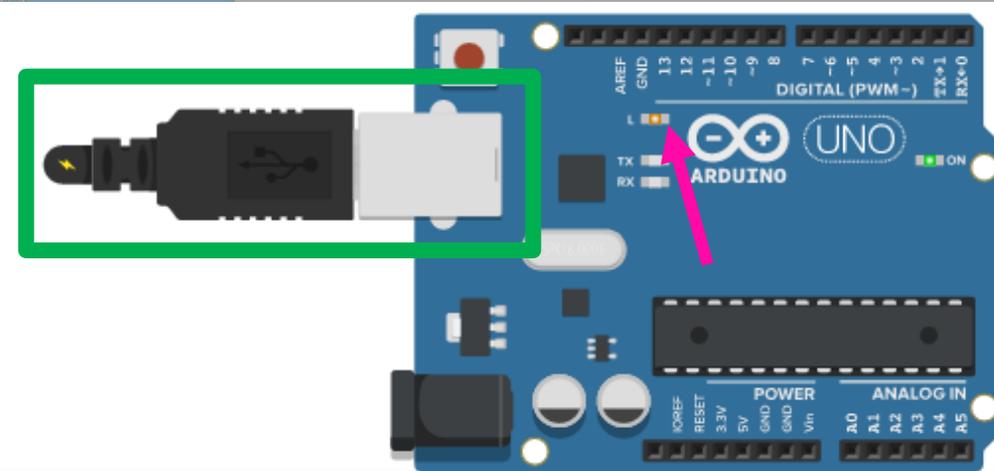
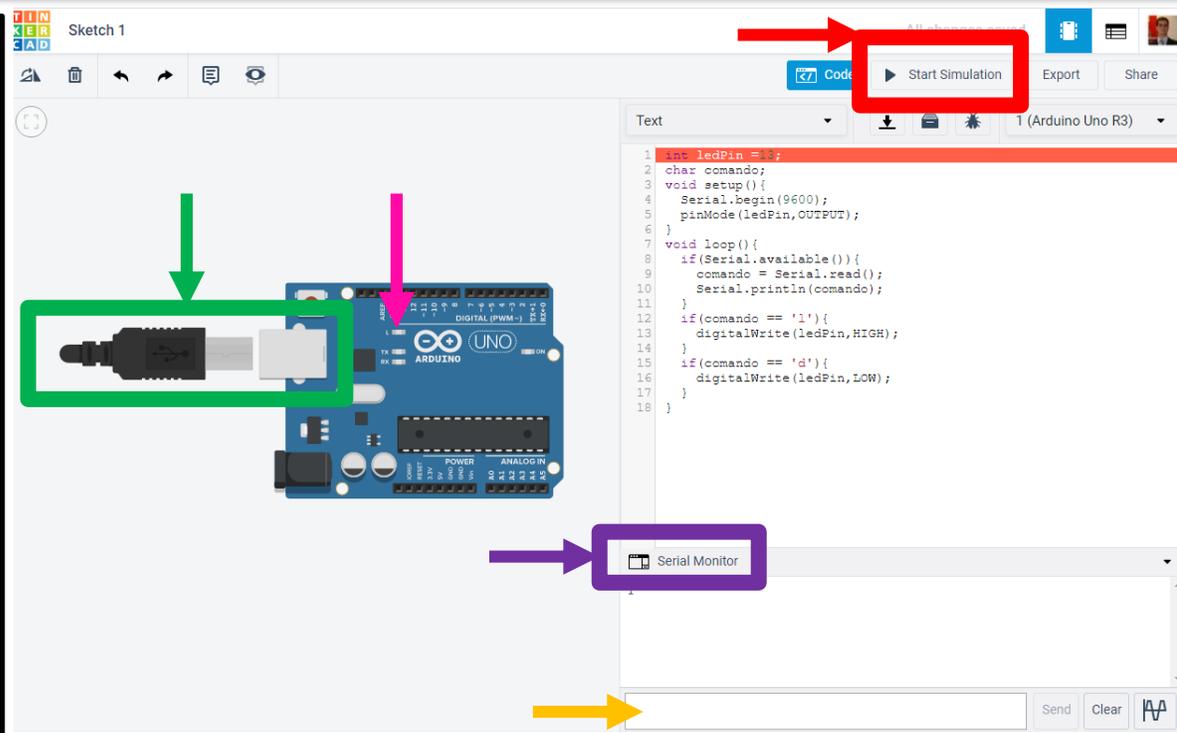
Note que a **USB** é conectada ao Arduino e o botão se torna **Stop Simulation**.

Clique então em **Serial Monitor**. Na parte mais abaixo, clique, digite **/** (de liga), de enter e veja o **LED** do pino 13 acender. Digite **d** (de desliga), de enter e veja o **LED** do pino 13 apagar.

```

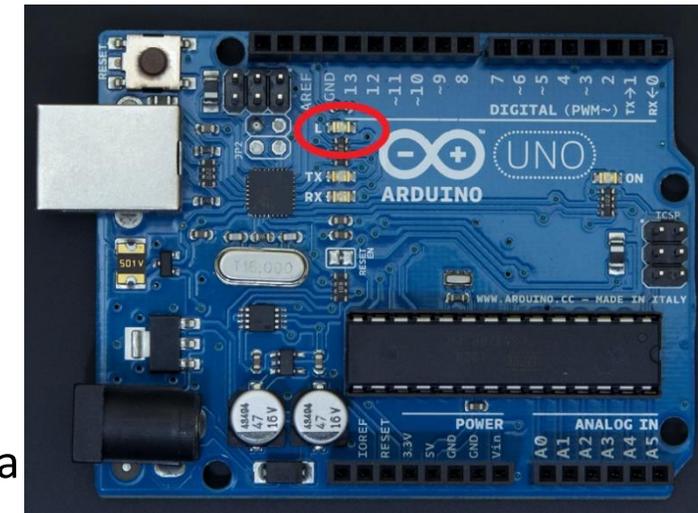
int ledPin =13;
char comando;
void setup() {
  Serial.begin(9600);
  pinMode(ledPin,OUTPUT);
}
void loop() {
  if(Serial.available()){
    comando = Serial.read();
    Serial.println(comando);
  }
  if(comando == '/'){
    digitalWrite(ledPin,HIGH);
  }
  if(comando == 'd'){
    digitalWrite(ledPin,LOW);
  }
}

```





- Para que o Arduino seja capaz de executar a comunicação serial, esta deve ser inicializada dentro de `setup()`, através de `Serial.begin(9600)`, com 9600 sendo a velocidade da conexão em bits por segundo (baud rate).
- `Serial.available()` Retorna a quantidades de bytes disponíveis para leitura no **buffer** de leitura. Essa função auxilia em loops onde a leitura dos dados só é realizada quando há dados disponível. A quantidade máxima de bytes no buffer é 64.
- `pinMode()` Configura a porta especificada para se comportar tanto como entrada ou saída.
- `Serial.read()` Lê o byte mais recente apontado no buffer de entrada da serial.
- `Serial.print()` Escreve na serial texto em formato ASCII.
- `Serial.println()` Como a função `Serial.print()`, a diferença é que esta função acrescenta ao fim da mensagem o caractere de retorno de carro e o caractere de nova linha.
- `digitalWrite()` Escreve um valor HIGH ou um LOW em um pino digital. Se o pino foi configurado como uma saída (output), sua voltagem será determinada como 5V (ou 3.3V nas placas de 3.3V) para HIGH, 0V (terra) para LOW.



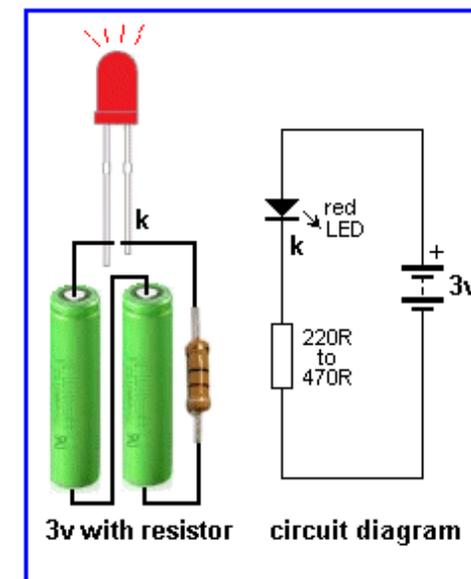
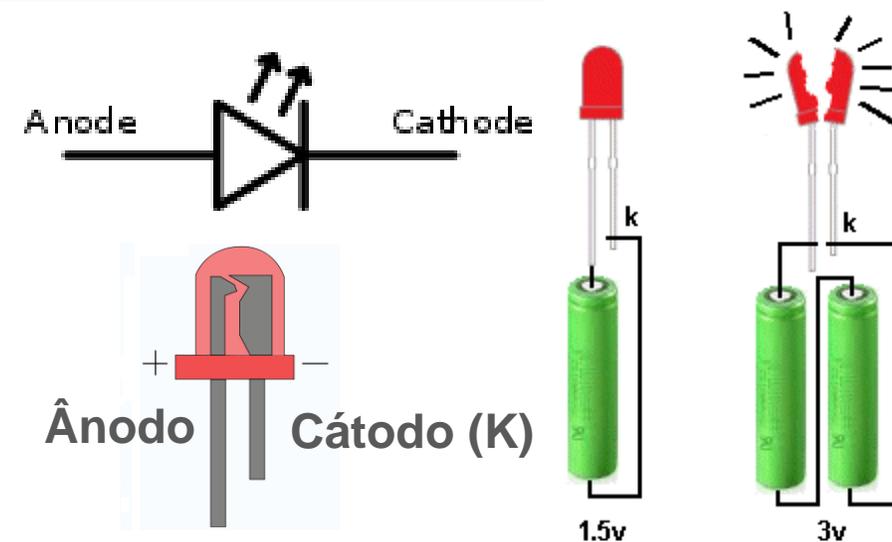


- O diodo é um componente eletrônico que permite corrente em uma direção, mas a impede de fluir na outra direção.
- No diodo, existe uma queda de potencial constante. O mesmo possui uma potência máxima, e, conseqüentemente, uma corrente máxima. O encapsulamento do diodo está diretamente relacionada com a capacidade de dissipação térmica do diodo e também a corrente máxima.
- Desta forma, sempre devemos colocar um resistor em série com o diodo para reduzir a corrente!



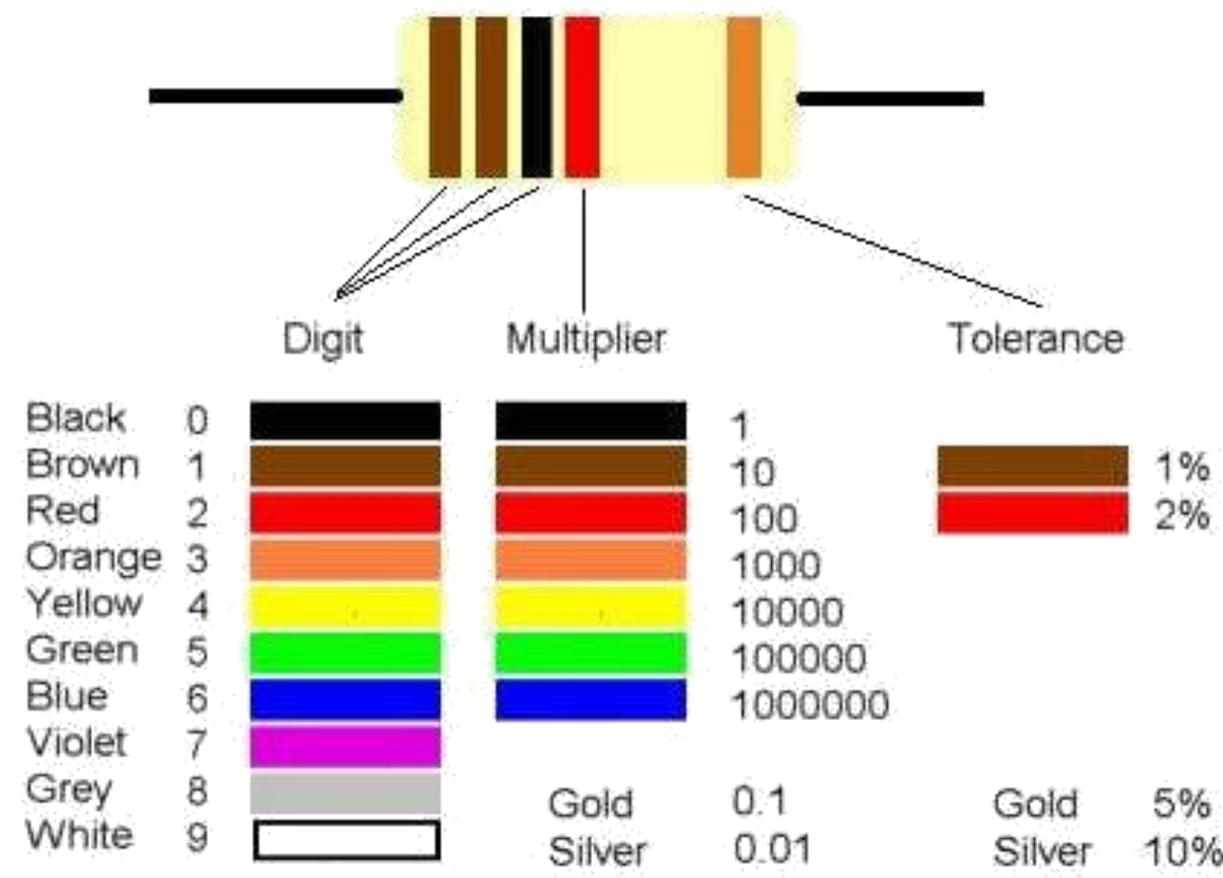


- LED significa Light Emitting Diode. Ou seja, ele é um diodo que emite uma luz quando há passagem de corrente. Assim como nos diodos, podemos considerar que há uma queda de voltagem no diodo constante, independente da corrente.
- O LED possui uma potência máxima suportada. Acima o mesmo queima. Como a queda de ddp é constante, ele possui uma corrente máxima. Desta forma, basta colocar um resistor em série para baixar a corrente!
- Se ligar sem resistor numa fonte de tensão, ele vai acender por algum tempo e depois queimar!



# Eletrônica Básica – Resistores

- Existem diversos tipos de resistores, sempre com o valor de sua resistência descrita por faixas ao redor do mesmo.
- Uma dessas faixas é responsável por informar a tolerância do valor nominal da resistência.
- Assim como no diodo, o encapsulamento está diretamente relacionado com a capacidade de dissipação térmica e, conseqüentemente, com a potência e conseqüente corrente máximas.



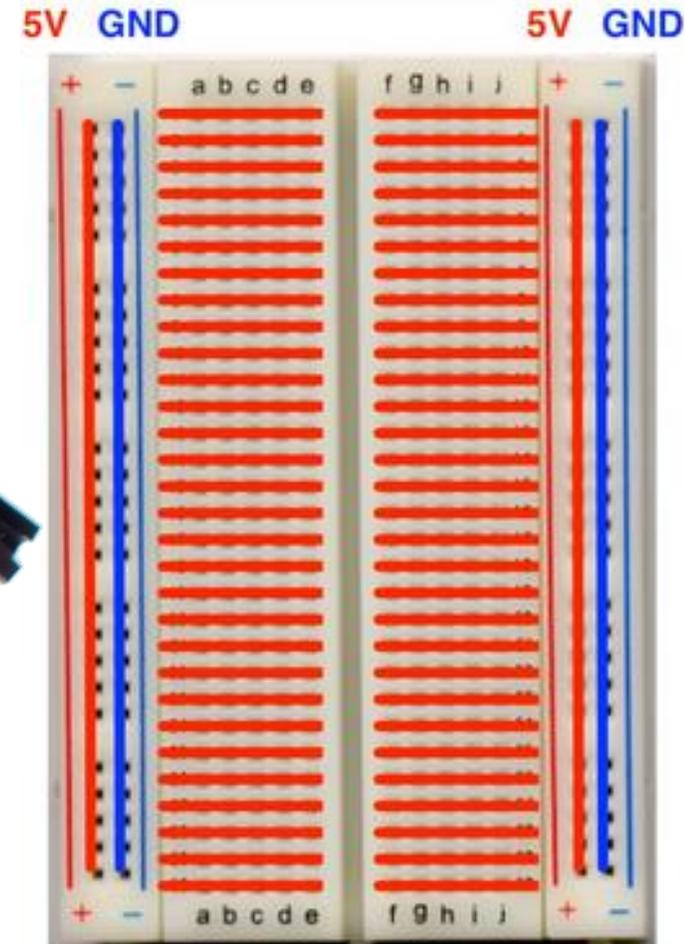


- Cada LED de cor diferente tem uma queda de potencial diferente.
- Se temos uma bateria de 9V e um LED vermelho com queda de potencial de 2V, teremos uma voltagem de 7V na resistência. Como sabemos a corrente máxima de 20mA, basta dividir a voltagem de 7V pela corrente de 20mA para obtermos uma resistência de 350  $\Omega$ ;
- Para Leds verde, amarelo e laranja temos uma corrente de  $i = 20$  a 25 mA e uma voltagem  $V = 2,1$  Volts. Para Leds azul e branco temos uma corrente de  $i = 15$  a 30 mA e uma voltagem  $V = 2,8$  a 4,2 Volts;
- Uma regra prática é sempre usar um resistor entre 470  $\Omega$  e 1k  $\Omega$ .



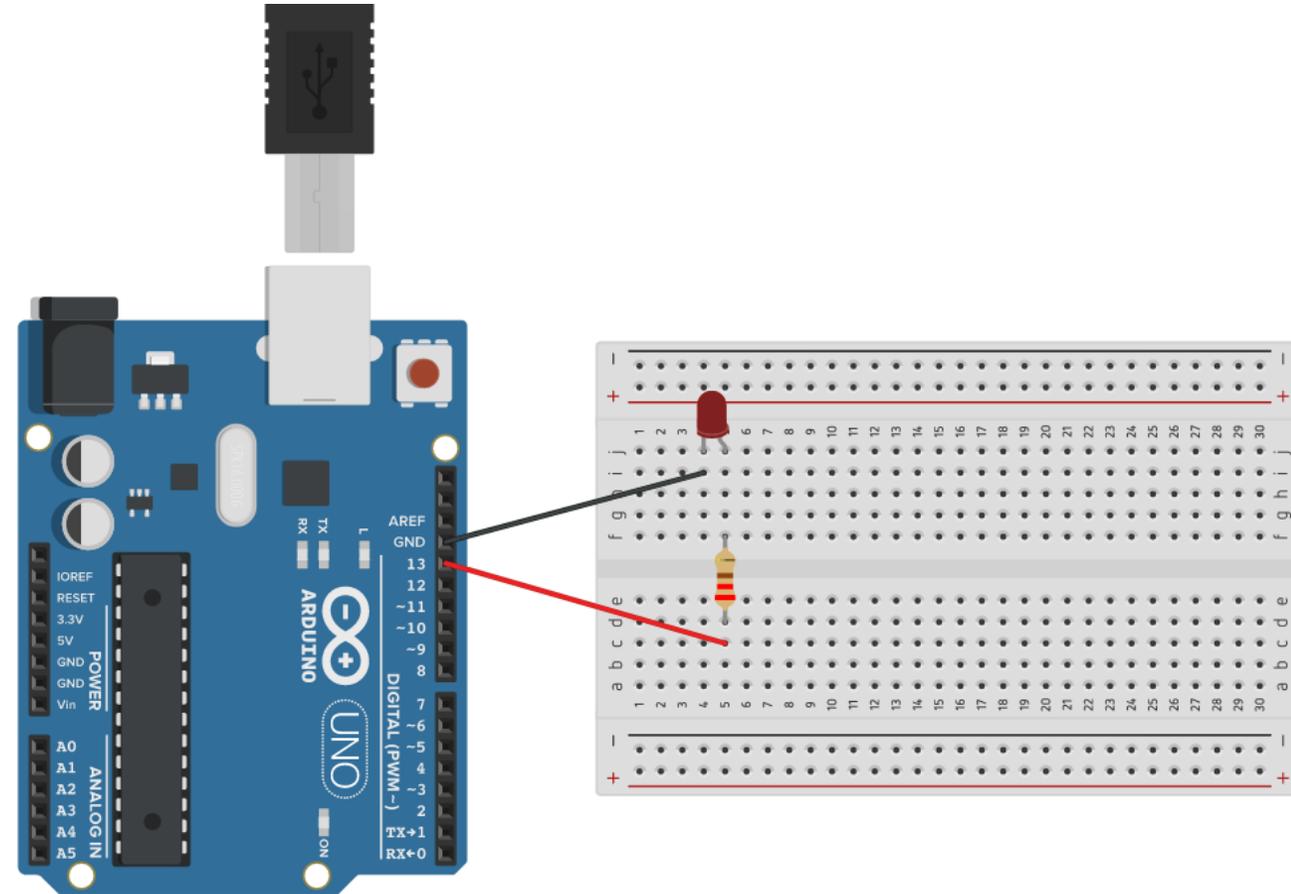


- Mas como podemos interligar os componentes de maneira que seja prático alterar e testar circuitos, ou seja, sem soldar os componentes?
- Para tal utilizamos tanto a protoboard como cabos jumper. A protoboard tem furos conectados entre si conforme a imagem ao lado.
- Já os cabos jumper servem para facilitar a utilização da protoboard, com pontas de conexão *normalizadas*, evitando mal contato.





- Monta o circuito ao lado com um LED vermelho em série com um resistor de 220 Ohm;
- Teste o mesmo programa carregado anteriormente no arduino.



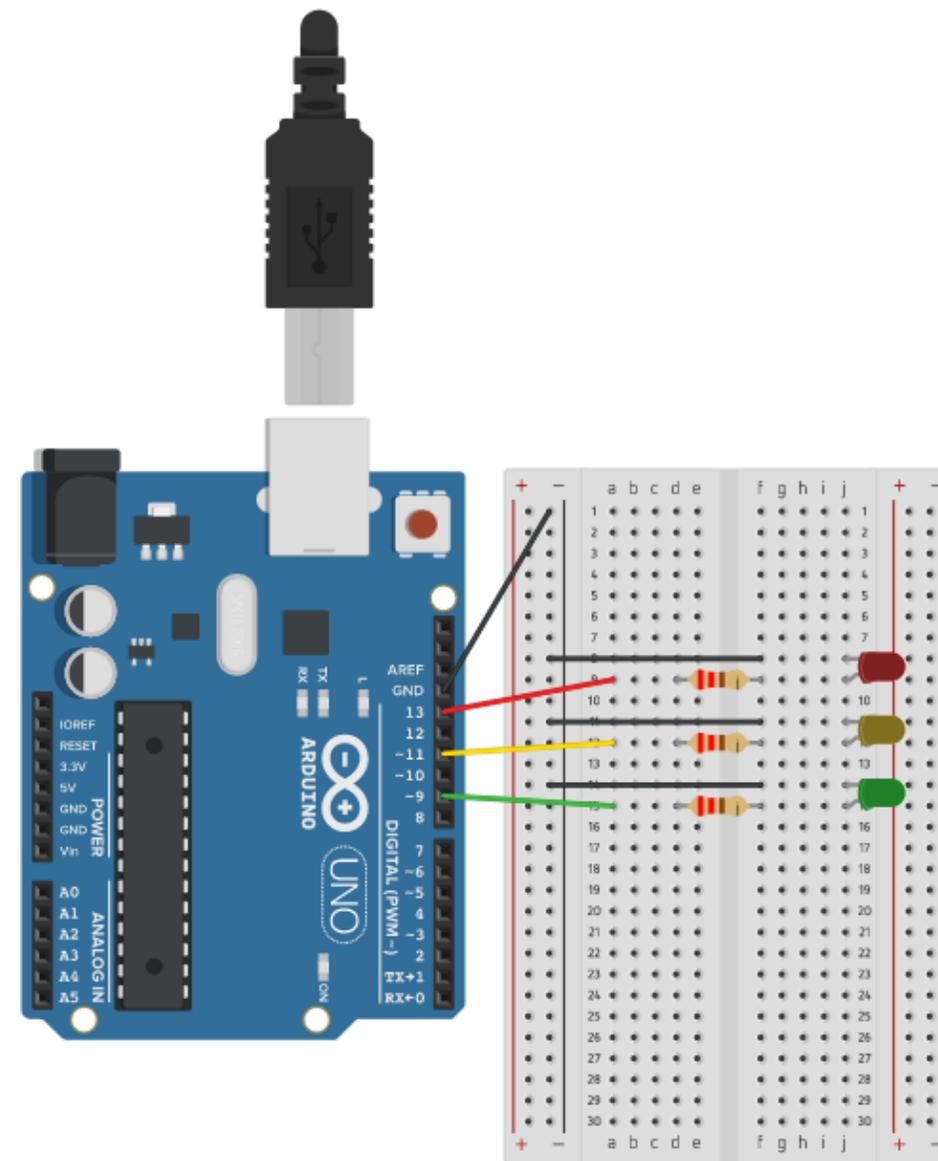


# Exercício 03 – Farol – Sketch\_02

Faça um arranjo com três LEDs, cada um conectado a uma porta digital diferente e os faça piscar, um de cada vez, cada um durante um determinado tempo.

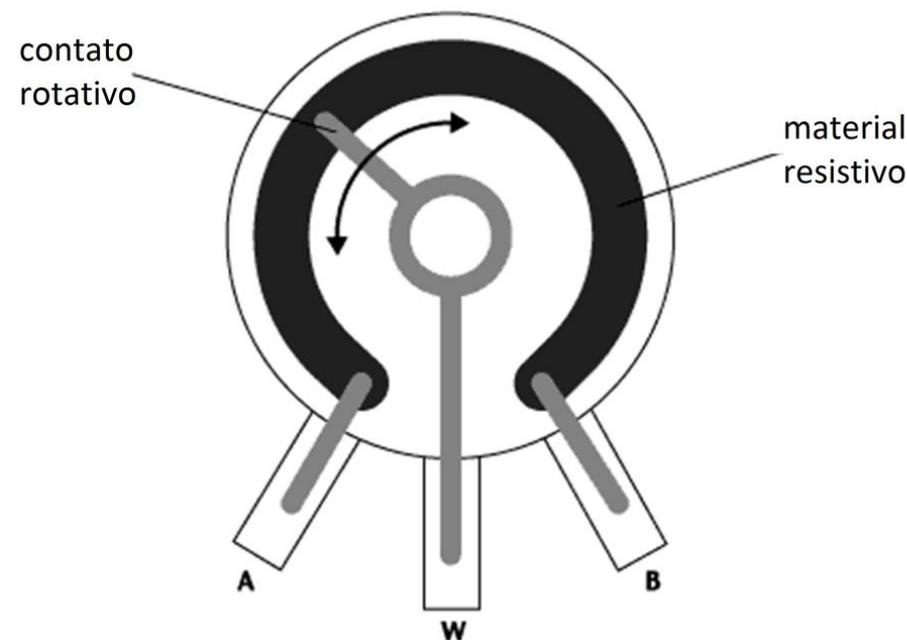
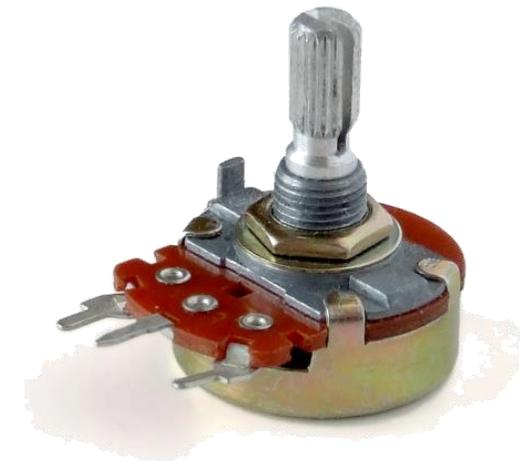
Utilize a função `delay()`. Essa função interrompe o programa para o período de tempo (em milissegundos) especificado como parâmetro.

```
int ledPin_1 = 13;
int ledPin_2 = 11;
int ledPin_3 = 9;
void setup() {
  pinMode(ledPin_1, OUTPUT);
  pinMode(ledPin_2, OUTPUT);
  pinMode(ledPin_3, OUTPUT);
}
void loop() {
  digitalWrite(ledPin_1, HIGH);
  delay(500);
  digitalWrite(ledPin_1, LOW);
  delay(500);
  digitalWrite(ledPin_2, HIGH);
  delay(250);
  digitalWrite(ledPin_2, LOW);
  delay(250);
  digitalWrite(ledPin_3, HIGH);
  delay(125);
  digitalWrite(ledPin_3, LOW);
  delay(125);
}
```



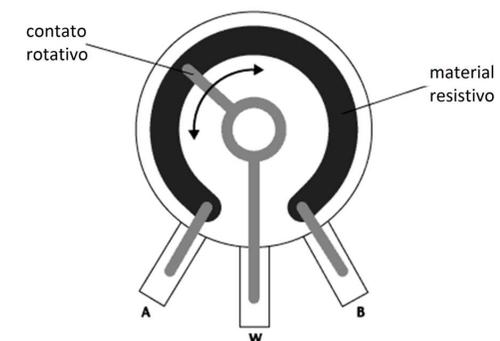


- Um potenciômetro é um simples botão giratório que fornece uma resistência variável e que pode ser lida pelo Arduino como um valor analógico.
- Possui internamente uma trilha resistiva (de níquel-cromo ou de carbono), sobre a qual desliza um cursor, que altera a resistência elétrica entre seu conector central e um dos dois laterais (normalmente são três conectores).
- Para lermos o valor a posição do potenciômetro no Arduino, usamos um conversor analógico-digital. No arduino, como o ADC é de 10 bits, temos 1024 níveis lógicos: de 0 a 1023.





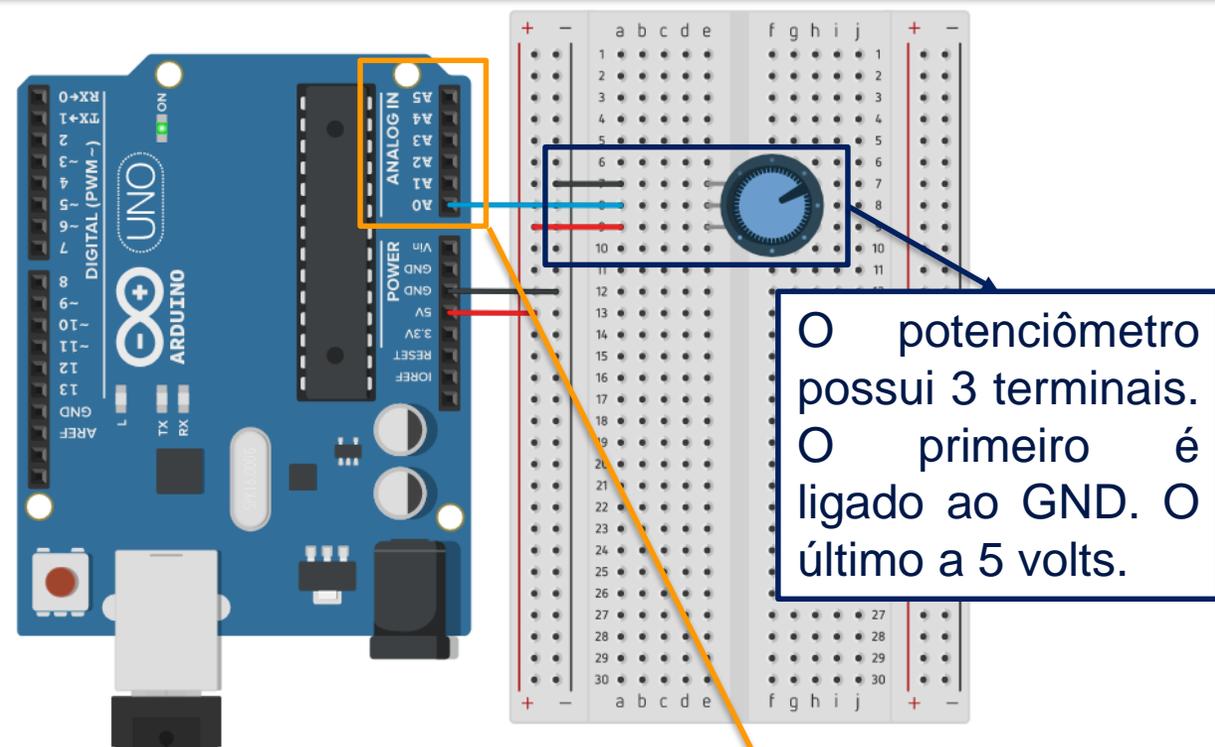
- Se girarmos o cursor do potenciômetro, alteramos a resistência em cada lado do contato elétrico que vai conectado ao terminal central do botão. Isso provoca a mudança na proximidade do terminal central aos 5 volts ou ao terra, o que implica numa mudança no valor analógico de entrada. Quando o cursor for levado até o final da escala, teremos por exemplo zero volts a ser fornecido ao pino de entrada do Arduino e, assim, ao lê-lo obteremos 0. Quando giramos o cursor até o outro extremo da escala, haverá 5 volts a ser fornecido ao pino do Arduino e, ao lê-lo, teremos 1023. Em qualquer posição intermediária do cursor, teremos um valor entre 0 e 1023, que será proporcional à tensão elétrica sendo aplicada ao pino do Arduino.





# Exercício 04 – potenciômetro – Sketch\_03

```
int sensorPin = A0;
int sensorValue = 0;
double Voltagem = 0.0;
void setup() {
  Serial.begin(9600);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  Voltagem = interpolacao((double)sensorValue,0.0,1023.0,0.0,5.0);
  Serial.print("Nivel Logico ADC: ");
  Serial.print(sensorValue);
  Serial.print(" Voltagem: ");
  Serial.print(Voltagem);
  Serial.println("V");
}
double interpolacao(double valor, double old_min, double old_max, double new_min, double new_max){
  double derivada;
  double delta_old;
  double valor_new;
  derivada = (new_max-new_min)/(old_max-old_min);
  delta_old = valor - old_min;
  valor_new = new_min + derivada*delta_old;
  return(valor_new);
}
```



O potenciômetro possui 3 terminais. O primeiro é ligado ao GND. O último a 5 volts.

O terminal central do potenciômetro deve ser ligado ao pino analógico, que pode assumir valores diversos, e não apenas 0 e 1 como o caso do pino digital. Internamente, existe um conversor analógico-digital, que converte os 5V em 1024 níveis, indo de 0 a 1023.

# Exercício 04 – potenciômetro – Sketch\_03



TIN  
KER  
CAD

Sketch\_03

All changes saved

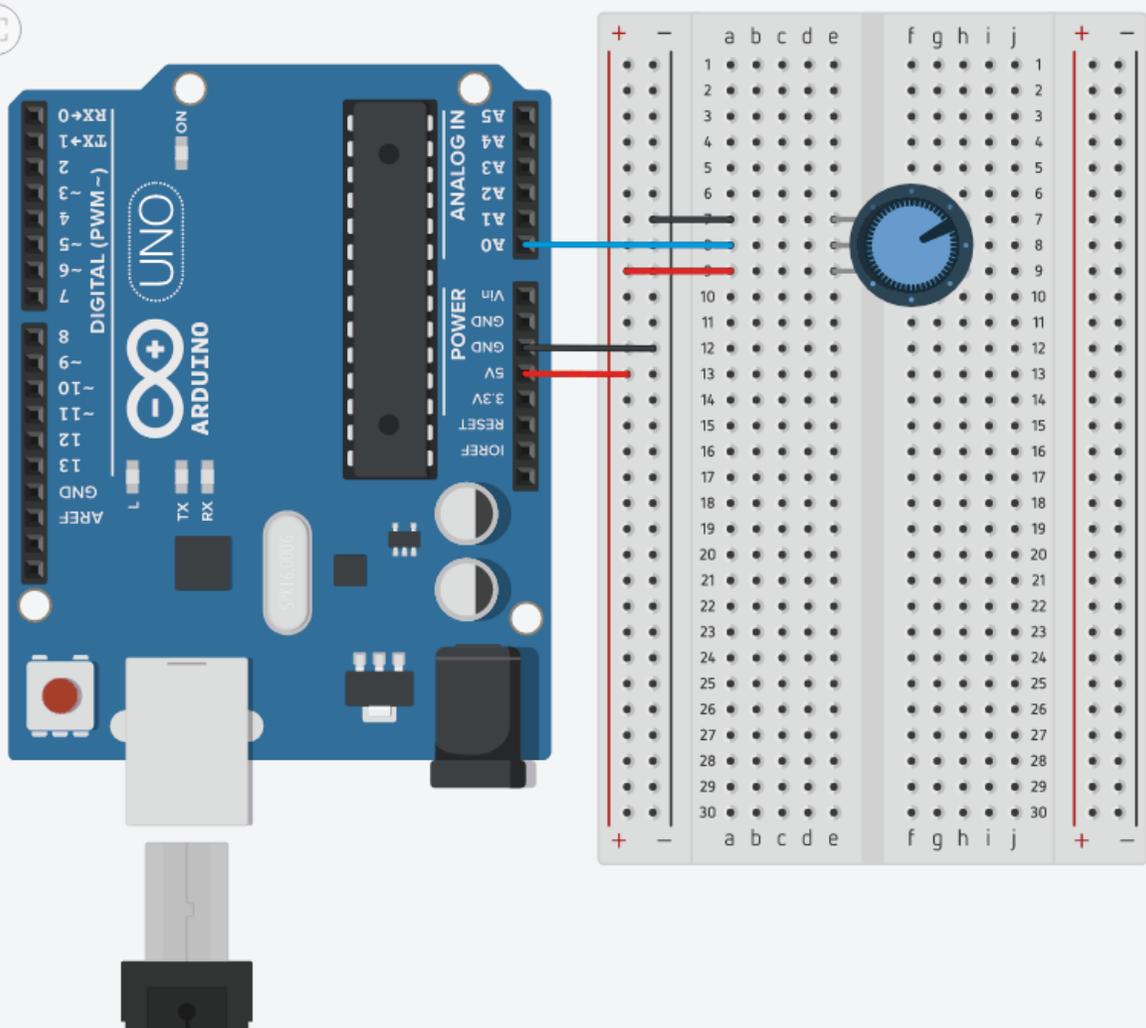


Code

Start Simulation

Export

Share



Text



1 (Arduino Uno R3)

```
1 int sensorPin = A0;
2 int sensorValue = 0;
3 double Voltagem = 0.0;
4 void setup() {
5     Serial.begin(9600);
6 }
7 void loop() {
8     sensorValue = analogRead(sensorPin);
9     Voltagem = interpolacao((double)sensorValue,0.0,1023.0,0.0,5.0);
10    Serial.print("Nivel Logico ADC: ");
11    Serial.print(sensorValue);
12    Serial.print(" Voltagem: ");
13    Serial.print(Voltagem);
14    Serial.println("V");
15 }
16 double interpolacao(double valor, double old_min, double old_max, double new_min, double new_max){
17     double derivada;
18     double delta_old;
19     double valor_new;
20     derivada = (new_max-new_min)/(old_max-old_min);
21     delta_old = valor - old_min;
22     valor_new = new_min + derivada*delta_old;
23     return(valor_new);
24 }
25
```

Serial Monitor

Send Clear



- A figura ao lado é o módulo sensor reflexivo infravermelho. Este é o sensor principal para desenvolver robôs seguidores de linha!
- O sensor conta com um transmissor e um receptor de infravermelho. O mesmo deve estar posicionado próximo ao chão e deve-se evitar que haja luz externa sendo captada pelo receptor.
- Assim como a grande maioria dos módulos, sensores ou atuadores que iremos estudar na mecatronica, este sensor possui uma entrada de terra (gnd) e uma de alimentação +Vcc, nesse caso 5V. O pino denominado D2 liga ou desliga o transmissor. A quantidade de luz captada pelo receptor é traduzida em voltagem de 0 a 5V, ou seja, deve ser colocado em um ADC do Arduino.





```
int sensorPin = A0;
int sensorValue = 0;
int emissorPin = 2;
void setup() {
  Serial.begin(9600);
  pinMode(emissorPin, OUTPUT);
  digitalWrite(emissorPin, HIGH);
}
void loop() {
  sensorValue = analogRead(sensorPin);
  Serial.print("Nivel Logico ADC: ");
  Serial.println(sensorValue);
}
```

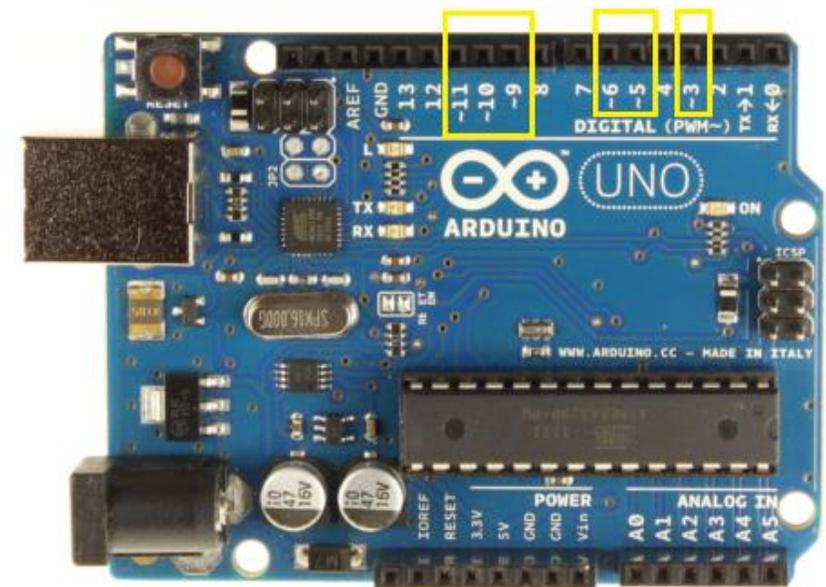
- Ligue os 4 fios do módulo de sensor de linha conforme a indicação no próprio sensor e no slide anterior.
- Compile e carregue no Arduino o código ao lado (Sketch\_04) e abra o serial plotter, no menu Tools. Aproveite a área preta na parte inferior da IDE do Arduino e use o sensor virando-o alternadamente para a área preta e branca. O que acontece com o gráfico?



- PWM é a modulação da largura de ondas quadradas.
- PWM, do inglês Pulse Width Modulation, é uma técnica utilizada por sistemas digitais para variação do valor médio da voltagem, enquanto só se aplica na saída voltagens nula ou máxima. A técnica consiste em manter a frequência de uma onda quadrada fixa e variar o tempo que o sinal fica em nível lógico alto. Esse tempo é chamado de duty cycle.

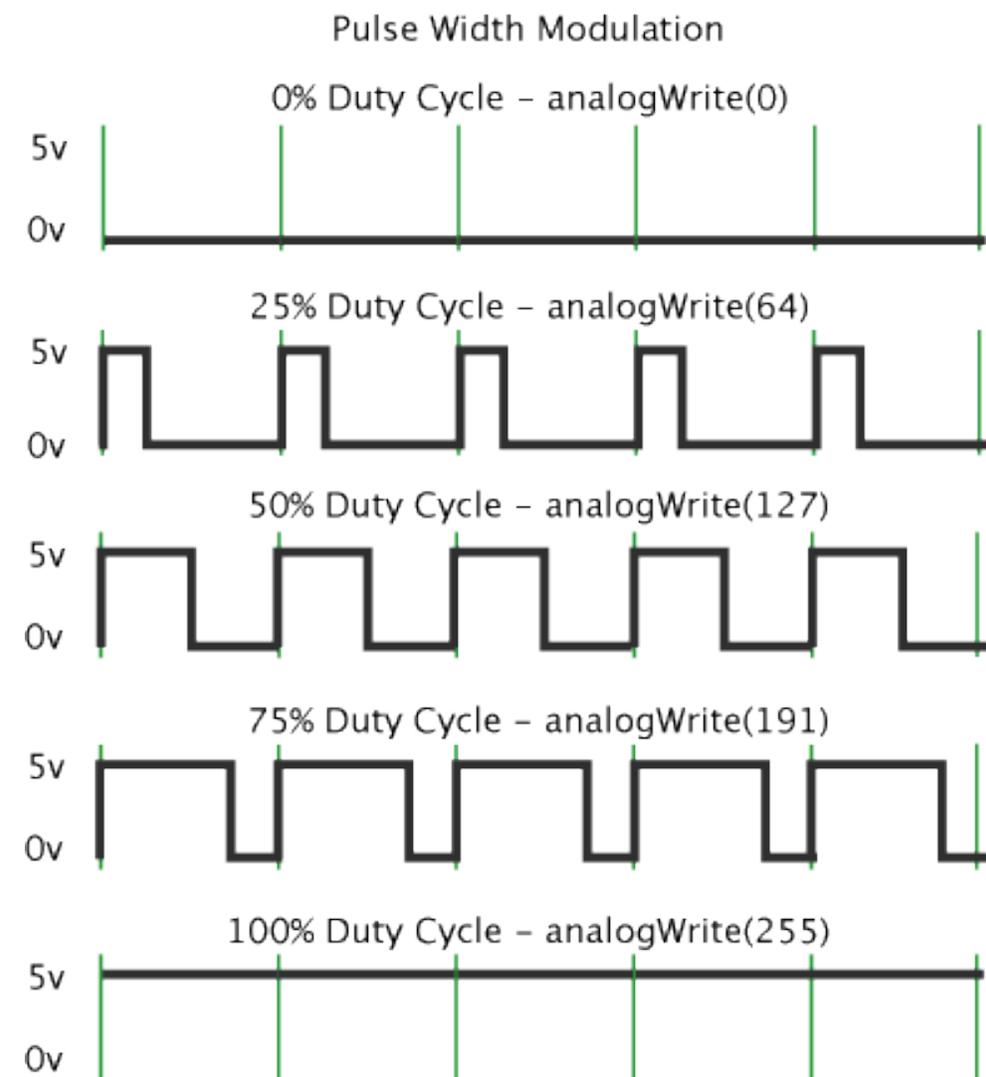
A frequência do PWM no Arduino é:

- 1KHz nos pinos 5 e 6
- 500 Hz nos pinos 3,9,10,11





- O gráfico ao lado ilustra a influência da variação do duty cycle no valor analógico médio.
- No Arduino, conforme explicado anteriormente, somente os pinos com ~ possuem PWM por Hardware, sendo controlados pela função `analogWrite(pino, valor)`, no qual o pino corresponde ao pino que será gerado o sinal PWM (3, 5, 6, 9, 10 ou 11) e o valor corresponde ao duty cycle, variando de 0 a 255. Se for 0, a saída permanece sempre em nível baixo (0V) e 255 a saída permanece sempre em nível alto (5V).



# Acionamentos – PWM – Sketch\_05



```
int led = 9;
int brilho = 0;
int delta = 5;
void setup() {
  pinMode(led, OUTPUT);
}
void loop() {
  analogWrite(led, brilho);
  brilho = brilho + delta;
  if(brilho<=0 || brilho>=255){
    delta = -delta;
  }
  delay(30);
}
```

The screenshot displays the Arduino IDE interface for a simulation titled "Sketch\_fade". The central workspace shows a 3D model of an Arduino Uno R3 board connected to a breadboard. A red LED is connected to digital pin 9, with its anode to the pin and its cathode to ground. A 220Ω resistor is connected between the LED's anode and the positive rail of the breadboard. A 10.0V power source is connected to the positive rail, and a 10.0ms time scale is indicated. The code editor on the right contains the following code:

```
1 int led = 9;
2 int brilho = 0;
3 int delta = 5;
4 void setup() {
5   pinMode(led, OUTPUT);
6 }
7 void loop() {
8   analogWrite(led, brilho);
9   brilho = brilho + delta;
10  if(brilho<=0 || brilho>=255){
11    delta = -delta;
12  }
13  delay(30);
14 }
15
```

The interface also shows a "Stop Simulation" button, a "Code" button, and a "Serial Monitor" window at the bottom.



- Conforme vimos, somente os pinos com ~ possuem a capacidade de pwm por hardware.
- Entretanto, podemos usar PWM nos outros pinos emulando via software. Para tanto, usaremos a biblioteca SoftPWM.
- Para instalar a biblioteca, clique em **Sketch** → **Include Library** → **Manage Libraries...**
- Digite SoftPWM no campo de procura, clique sobre a biblioteca e a instale.
- Compile, carregue e execute o Sketch\_05, detalhado ao lado e observe o LED ligado ao pino 13.

Library Manager

Type All Topic All softpwm

**SoftPWM** by Brett Hagman Version 1.0.0 **INSTALLED**  
A software library to produce a 50 percent duty cycle PWM signal on arbitrary pins.  
A Wiring Framework (and Arduino) Library, for Atmel AVR8 bit series microcontrollers, to produce PWM signals on any arbitrary pin. It was originally designed for controlling the brightness of LEDs, but could be adapted to control servos and other low frequency PWM controlled devices as well.  
It uses a single hardware timer (Timer 2) on an Atmel AVR 8 bit microcontroller to generate up to 20 PWM channels (your mileage may vary).  
Issues or questions: <https://github.com/bhagman/SoftPWM/issues>  
[More info](#)

```
#include <SoftPWM.h>
int led = 13;
int brightness = 0;
int fadeAmount = 5;
void setup(){
  SoftPWMBegin();
  SoftPWMSet(led, 0);
}
void loop(){
  SoftPWMSet(led, brightness);
  brightness = brightness + fadeAmount;
  if (brightness <= 0 || brightness >= 255) {
    fadeAmount = -fadeAmount;
  }
  delay(30);
}
```

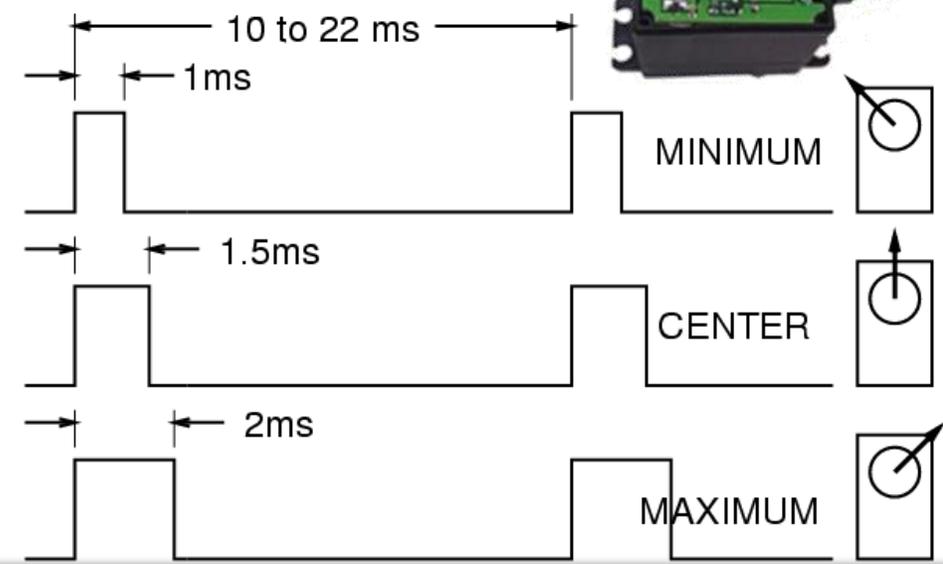
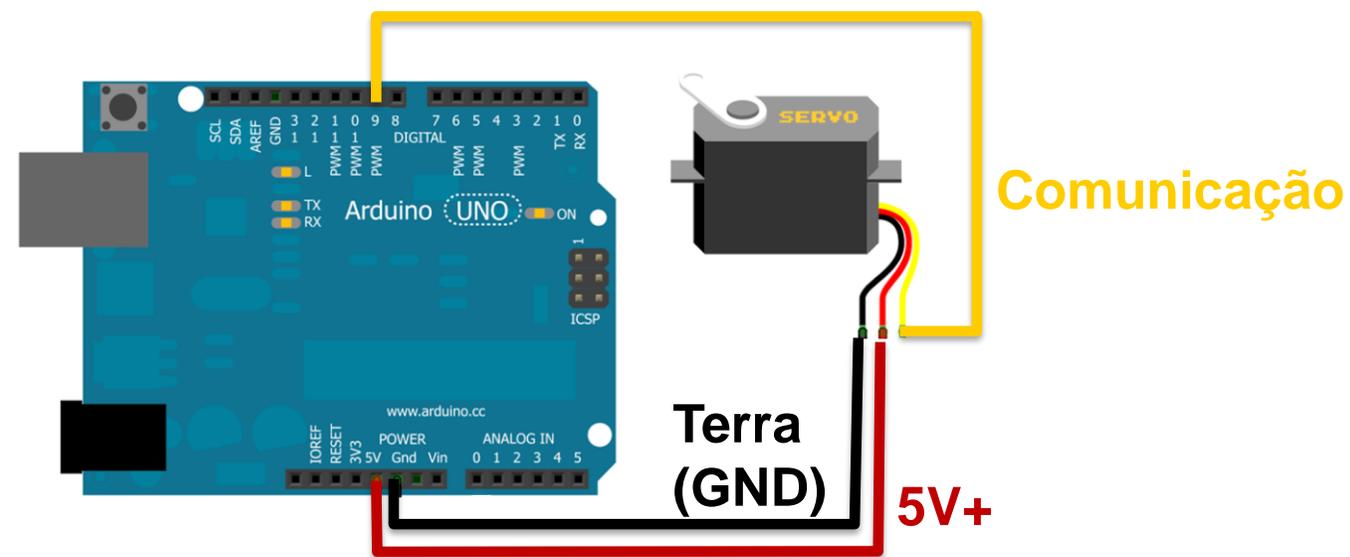
# Acionamentos – PWM para servo motores

O Servo motor é na verdade um sistema: possui uma placa de controle, um potenciômetro (que serve de encoder), um motor CC e uma redução.

O Servo motor possui amplitude de movimento de 180° ou 360° (dependendo do modelo) cuja posição é indicada pelo Arduino a partir de um sinal PWM conforme descrito abaixo;

Se estiver com o Arduino, ligue no seu servo motor conforme o diagrama abaixo. Após, clique em **File** → **Examples** → **Servo** → **Sweep**.

Caso contrário, vá para o próximo slide ver como simular no TinkerCAD.



# Acionamentos – PWM para servo motores – Sketch\_07



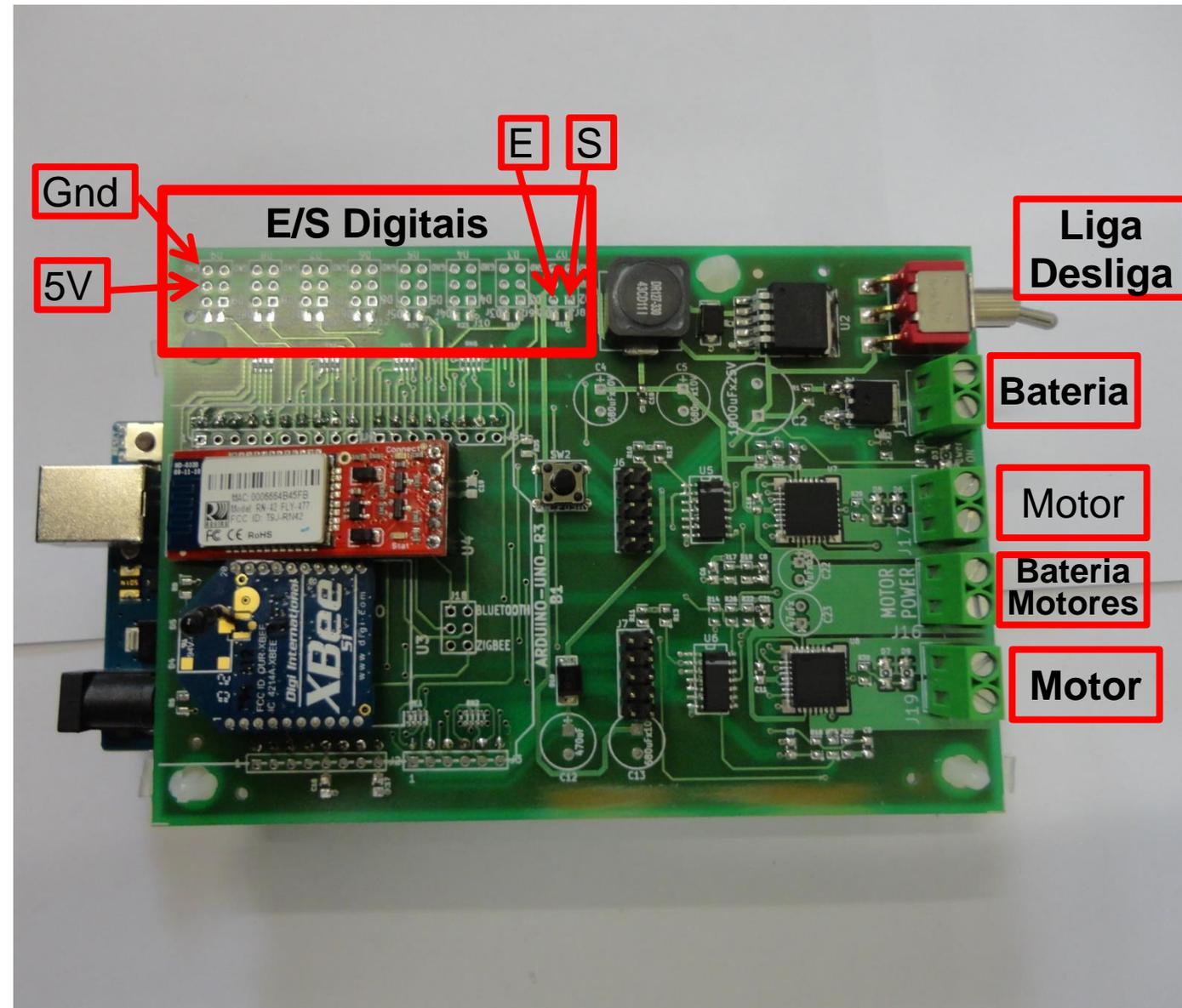
```
#include <Servo.h>
servo myservo;
int pos = 0;
void setup() {
  myservo.attach(8);
}
void loop() {
  for (pos = 0; pos <= 180; pos += 1) {
    myservo.write(pos);
    delay(15);
  }
  for (pos = 180; pos >= 0; pos -= 1) {
    myservo.write(pos);
    delay(15);
  }
}
```

The screenshot shows the Arduino IDE interface. The main window displays a simulation of an Arduino Uno R3 board connected to a servo motor (SM-523095) and an oscilloscope. The oscilloscope is connected to the servo motor's signal line and shows a square wave pulse. The code editor on the right contains the following code:

```
1 #include <Servo.h>
2 Servo myservo;
3 int pos = 0;
4 void setup() {
5   myservo.attach(8);
6 }
7 void loop() {
8   for (pos = 0; pos <= 180; pos += 1) {
9     myservo.write(pos);
10    delay(15);
11  }
12  for (pos = 180; pos >= 0; pos -= 1) {
13    myservo.write(pos);
14    delay(15);
15  }
16 }
```

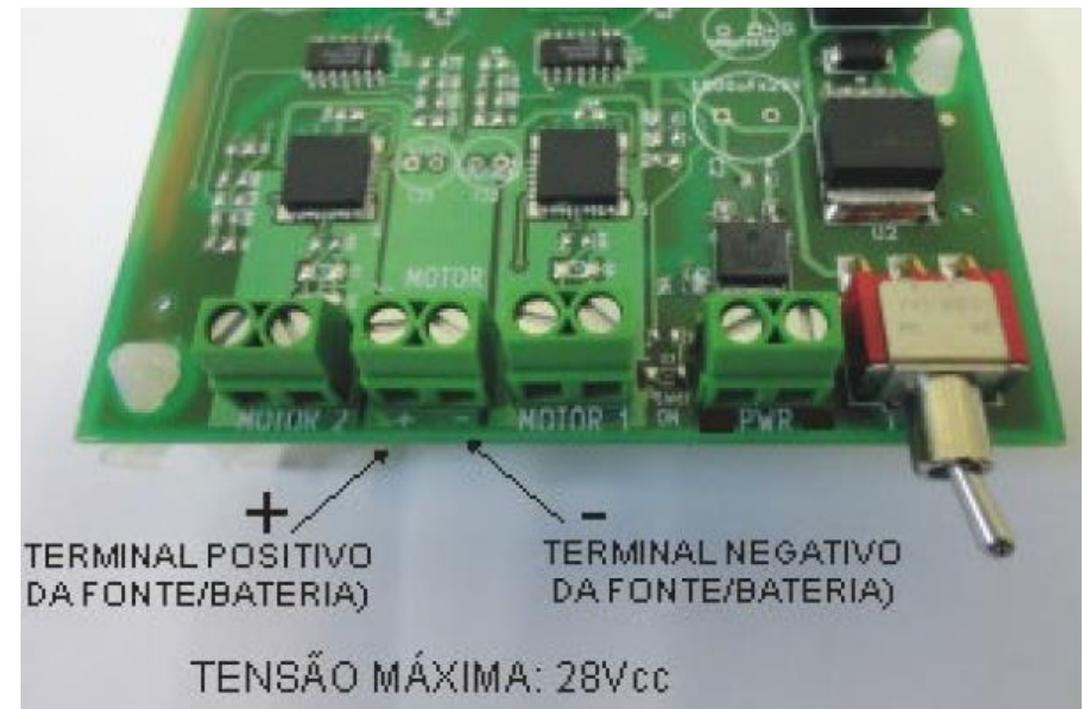
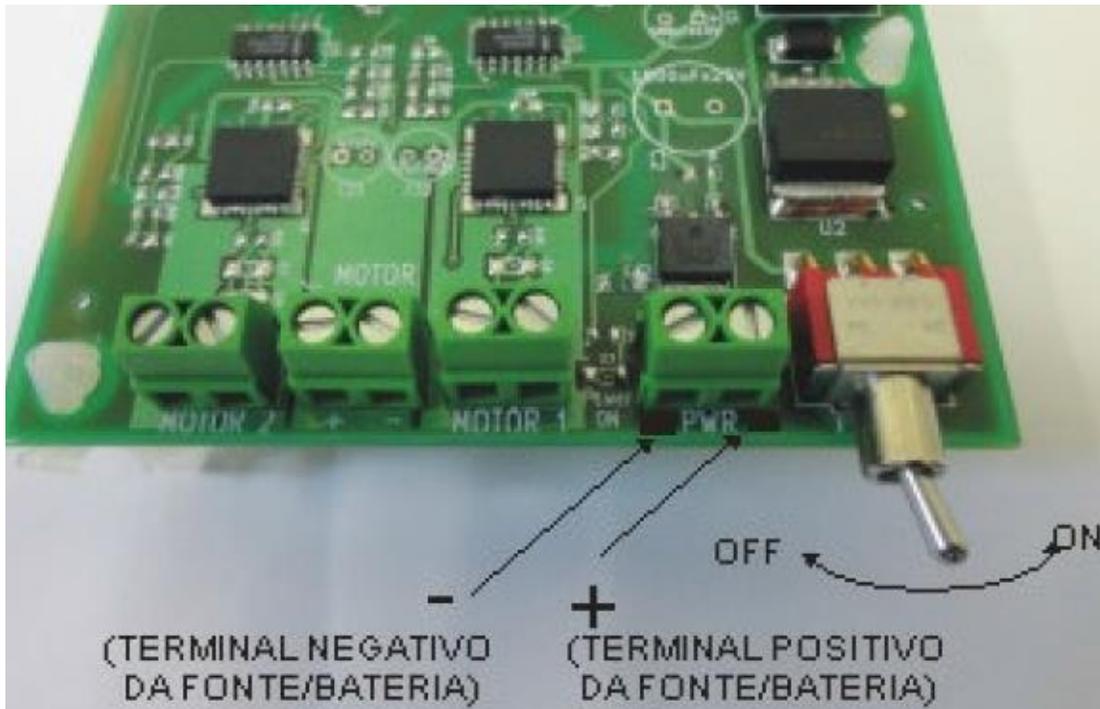


- A PACA contem 1 área de comunicações, com um módulo bluetooth, 1 módulo para entradas e saídas digitais, um modulo para regular a voltagem provinda da bateria e alimentar os componentes e por fim um módulo com driver para controlar motores DC;
- Um comentário é que o módulo de controle de motores deve ser energizado externamente, através da conexão denominada bateria motores na figura ao lado!





- Cuidado com as ligações entre bateria, PACA e módulo de driver de motor. Alguns textos de + e – estão trocados!
- SIGA ESTAS FIGURAS!





- Para o motor DC1, criarmos um sinal PWM no pino 9, para controlar a velocidade e um sinal digital LOW ou HIGH no pino 10 para controlar a direção.
- Para o motor DC2, criarmos um sinal PWM no pino 7, para controlar a velocidade e um sinal digital LOW ou HIGH no pino 8 para controlar a direção.
- Atente-se para a posição dos jumpers conforme figura ao lado!





```
int velocidade_PWM = 0;
int velocidadePerc = 0;
void setup(){
  Serial.begin(9600);
}
void loop() {
  char comando;
  if (Serial.available()) {
    comando = Serial.read();
    if (comando == 'a'){
      velocidadePerc = velocidadePerc + 5;}
    if (comando == 'd'){
      velocidadePerc = velocidadePerc - 5;}
    if (velocidadePerc<0){
      velocidadePerc = 0;}
    if (velocidadePerc>100){
      velocidadePerc = 100;}
  }
  velocidade_PWM = (int)(velocidadePerc*(255/100));
  Serial.print("velocidade ");
  Serial.print(velocidadePerc);
  Serial.println("%");
  analogWrite(9, velocidade_PWM);
}
```

- Caso já esteja com bateria carregada e ligada à PACA:

Faça as ligações do **motor 1** conforme explicado no slide anterior;

Compile e carregue o código ao lado.

Ao digitar “a” no serial monitor, o programa deve acelerar o motor em 5% e ao digitar “d” deve desacelerá-lo em 5%.

- Caso não esteja com o material em mãos, siga ao próximo slide para uma versão simplificada no TinkerCAD. Crie o circuito e use o código deste slide.

# Acionamentos com PACA – PWM, drivers e motores DC – Sketch\_08



TIN SKETCHER CAD Sketch\_Motor\_Vel

All changes saved

Simulator time: 00:00:00.117

Code Stop Simulation Export Share

Text

```
1 int velocidade_PWM = 0;
2 int velocidadePerc = 0;
3 char comando;
4 void setup(){
5   Serial.begin(9600);
6 }
7 void loop() {
8   if (Serial.available()) {
9     comando = Serial.read();
10    if (comando == 'a'){
11      velocidadePerc = velocidadePerc + 5;
12    }
13    if (comando == 'd'){
14      velocidadePerc = velocidadePerc - 5;
15    }
16    if (velocidadePerc<0){
17      velocidadePerc = 0;
18    }
19    if (velocidadePerc>100){
20      velocidadePerc = 100;
21    }
22    velocidade_PWM = (int)(velocidadePerc*(255/100));
23    Serial.print("velocidade ");
24    Serial.print(velocidadePerc);
25    Serial.println('%');
26    analogWrite(9, velocidade_PWM);
27 }
```

Serial Monitor

```
velocidade 20%velocidade 0%
velocidade 0%
velocid
```

Send Clear



```
int velocidadePerc = 0;
char comando;
void setup(){
  pinMode(9,OUTPUT);
  pinMode(10,OUTPUT);
  Serial.begin(9600);
} void loop() {
  char comando;
  if (Serial.available()) {
    comando = Serial.read();
    if (comando == 'a'){
      velocidadePerc = velocidadePerc + 5;}
    if (comando == 'd'){
      velocidadePerc = velocidadePerc - 5;}
    if (comando == 'b'){
      digitalWrite(10,HIGH);}
    if (comando == 'c'){
      digitalWrite(10,LOW);}
    if (velocidadePerc<0){
      velocidadePerc = 0;}
    if (velocidadePerc>100){
      velocidadePerc = 100;}
  }
  velocidade_PWM = (int)(velocidadePerc*(255/100));
  Serial.print(" velocidade ");
  Serial.print(velocidadePerc);
  Serial.println("%");
  analogWrite(9, velocidade_PWM ); }
```

- Caso já esteja com bateria carregada e ligada à PACA:
  - Faça as ligações do **motor 1** conforme explicado no slide anterior;
  - Compile e carregue o código ao lado.
  - Ao digitar “a” no serial monitor, o programa deve acelerar o motor em 5% e ao digitar “d” deve desacelerá-lo em 5%.
  - Ao digitar “b” no serial monitor, o motor deve girar em um sentido e ao digitar “c”, o motor deve girar no sentido oposto.
- Caso não esteja com o material em mãos, siga ao próximo slide para uma versão simplificada no TinkerCAD. Crie o circuito e use o código deste slide.

# Acionamentos com PACA – PWM, direção, drivers e motores DC – Sketch\_09



Exquisite Crift-Bigery

Simulator time: 00:00:00.119

All changes saved

Code Stop Simulation Export Share

```
1 int velocidade_PWM = 0;
2 int velocidadePerc = 0;
3 char comando;
4 void setup(){
5   Serial.begin(9600);
6   pinMode(9,OUTPUT);
7   pinMode(10,OUTPUT);
8   digitalWrite(10,LOW);
9 }
10 void loop() {
11   if (Serial.available()) {
12     comando = Serial.read();
13     if (comando == 'a'){
14       velocidadePerc = velocidadePerc + 5;
15     }
16     if (comando == 'd'){
17       velocidadePerc = velocidadePerc - 5;
18     }
19     if (velocidadePerc<0){
20       velocidadePerc = 0;
21     }
22     if (velocidadePerc>100){
23       velocidadePerc = 100;
24     }
25     if (comando == 'b'){
26       digitalWrite(10,LOW);
27     }
28     if (comando == 'c'){
29       digitalWrite(10,HIGH);
30     }
31   }
32   velocidade_PWM = (int)(velocidadePerc*(255/100));
33   Serial.print("velocidade ");
34   Serial.print(velocidadePerc);
35   Serial.println("%");
36   analogWrite(9,velocidade_PWM);
37 }
```

Serial Monitor

velocidadvelocidade 0%  
velocidade 0%  
velocidade 0%  
velocidade 0%  
velocidade 0%  
velocidade 0%  
velocidade 0%  
velocidade

Send Clear



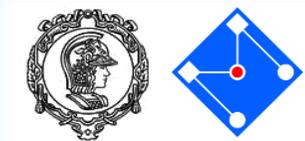
```
#include <SoftPWM.h>
int velocidadePerc = 0;
void setup(){
  pinMode(10,OUTPUT);
  Serial.begin(9600);
  SoftPWMBegin();
  SoftPWMSet(9, 0);
} void loop() {
  char comando;
  if (Serial.available()) {
    comando = Serial.read();
    if (comando == 'a'){
      velocidadePerc = velocidadePerc + 5;}
    if (comando == 'd'){
      velocidadePerc = velocidadePerc - 5;}
    if (comando == 'b'){
      digitalWrite(10,HIGH);}
    if (comando == 'c'){
      digitalWrite(10,LOW);}
    if (velocidadePerc<0){
      velocidadePerc = 0;}
    if (velocidadePerc>100){
      velocidadePerc = 100;}
    Serial.print(" velocidade ");
    Serial.print(velocidadePerc);
    Serial.println("%");
    SoftPWMSetPercent(9, velocidadePerc); } }
```

- Caso já esteja com bateria carregada e ligada à PACA: Faça as ligações do **motor 2** conforme explicado no slide anterior;  
Compile e carregue o código ao lado.  
Ao digitar “a” no serial monitor, o programa deve acelerar o motor em 5% e ao digitar “d” deve desacelerá-lo em 5%.  
Ao digitar “b” no serial monitor, o motor deve girar em um sentido e ao digitar “c”, o motor deve girar no sentido oposto.
- Caso não esteja com o material em mãos, siga ao próximo slide para uma versão simplificada no TinkerCAD. Crie o circuito e use o código deste slide.



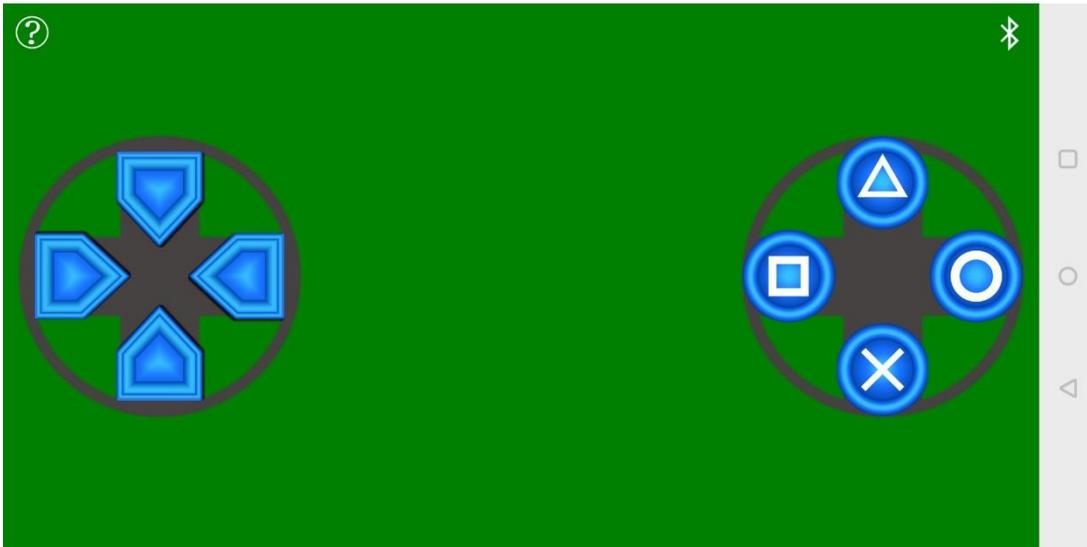
```
#include <SoftwareSerial.h>
SoftwareSerial bt(11,12); //RX,TX
char comando;
void setup() {
  pinMode(13,OUTPUT);
  Serial.begin(9600);
  bt.begin(9600);
  Serial.println("Começando...");
}
void loop() {
  if(bt.available()){
    comando = bt.read();
    Serial.println(comando);
  }
  if(Serial.available()){
    bt.write(Serial.read());
  }
  if(comando == 'F'){
    digitalWrite(13,HIGH);
  }
  if(comando == 'H'){
    digitalWrite(13,LOW);
  }
}
```

- Como o arduino uno não consegue. Ao estar sendo alimentado pelo cabo USB, dar energia suficiente para o módulo bluetooth, tenha certeza que a bateria está alimentando a PACA;
- A primeira coisa a se fazer para a comunicação bluetooth é dar upload do código ao lado para o arduino uno.



```
COM3 (Arduino/Genuino Uno)
AT+NAMErafael Send
começando...
*****
* Command      Description
* -----
* AT           Check if the command terminal work normally
* AT+RESET     Software reboot
* AT+VERSION   Get firmware, bluetooth, HCI and LMP version
* AT+HELP      List all the commands
* AT+NAME      Get/Set local device name
* AT+PIN       Get/Set pin code for pairing
* AT+PASS      Get/Set pin code for pairing
* AT+BAUD      Get/Set baud rate
* AT+LADDR     Get local bluetooth address
* AT+ADDR      Get local bluetooth address
* AT+DEFAULT   Restore factory default
* AT+RENEW     Restore factory default
* AT+STATE     Get current state
* AT+PWRM      Get/Set power on mode(low power)
* AT+POWE      Get/Set RF transmit power
* AT+SLEEP     Sleep mode
* AT+ROLE      Get/Set current role.
* AT+PARI      Get/Set UART parity bit.
* AT+STOP      Get/Set UART stop bit.
* AT+START     System start working.
* AT+IMME      System wait for command when power on.
* AT+IBEA      Switch iBeacon mode.
* AT+IBE0      Set iBeacon UUID 0.
* AT+IBE1      Set iBeacon UUID 1.
* AT+IBE2      Set iBeacon UUID 2.
* AT+IBE3      Set iBeacon UUID 3.
* AT+MARJ      Set iBeacon MARJ .
* AT+MINO      Set iBeacon MINO .
* AT+MEA       Set iBeacon MEA .
* AT+NOTI      Notify connection event .
* AT+UUUI      Get/Set system SERVER_UUID .
* AT+CHAR      Get/Set system CHAR_UUID .
* -----
* Note: (M) = The command support slave mode only.
* For more information, please visit http://www.cyobd.com
* Copyright@2013 www.cyobd.com. All rights reserved.
*****
Autoscroll Both NL & CR 9600 baud Clear output
```

- Abra o Serial Monitor, na parte inferior direita, coloque *9600 baud* e troque para *both NL & CR* (ambos nova linha e carriage return);
- Digite na parte de cima: *AT+HELP* e de enter. Aqui serão listados vários comandos de configuração do seu módulo Bluetooth;
- Então digite *AT+NAMEseunome* (sim, é tudo junto) e de enter.
- Você receberá uma confirmação do novo nome do bluetooth.



- Em um celular Android, baixe o app BLEJoystick;
- Com o bluetooth ativado, clique no icone na parte superior direita da tela, clique em Scan e selecione seu dispositivo. Neste momento, o led vermelho do bluetooth ira ficar aceso constantemente;
- Aperte quadrado e bola alternadamente e verifique o que acontece com o led conectado ao pino 13 do arduino uno.