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

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

O que veremos nesta aula?

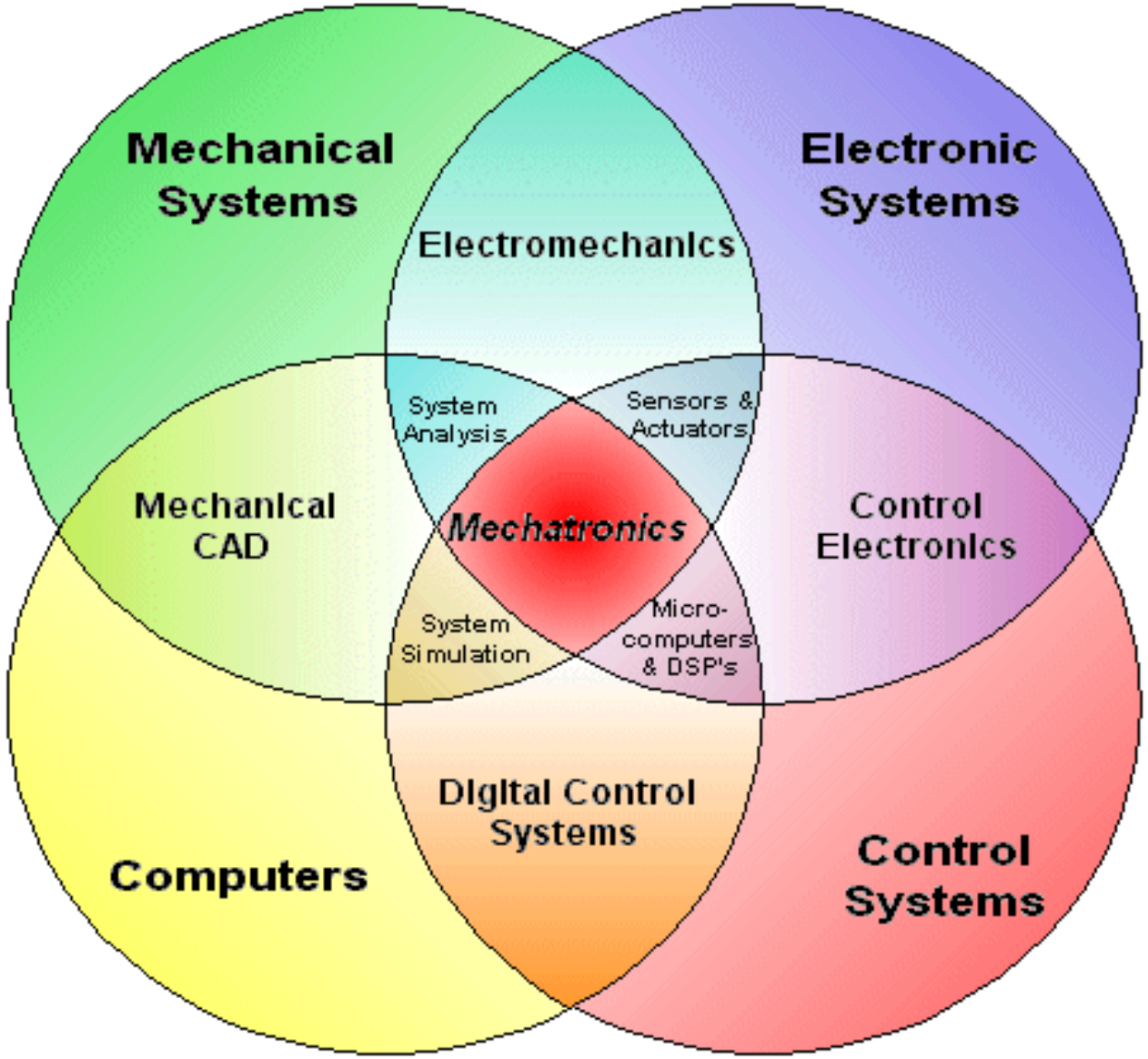
- Embasamento teórico;
- Arduino UNO;
- TinkerCAD;
- Comunicação Serial;
- Diodos, LEDs, resistores, protoboard, Arduino: entradas e saídas digitais;
- Potenciômetros, Arduino: entrada analógica, sensor de linha;
- PWM;
- PWM como comunicação, servo-motores;
- PACA
- Transistor, PWM com ponte-H, PACA: driver de motor;
- PACA: comunicação Bluetooth.

Mecatrônica

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.

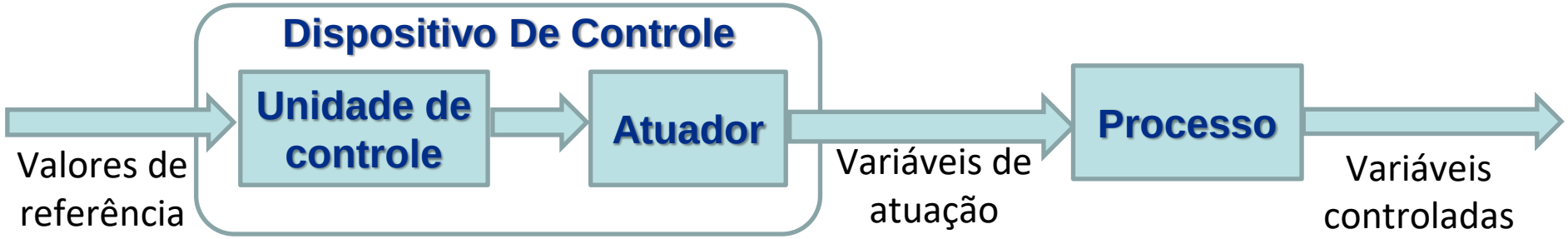
A mecatrônica é a intersecção entre as engenharias mecânica, eletrônica, de computação e de controle.

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.

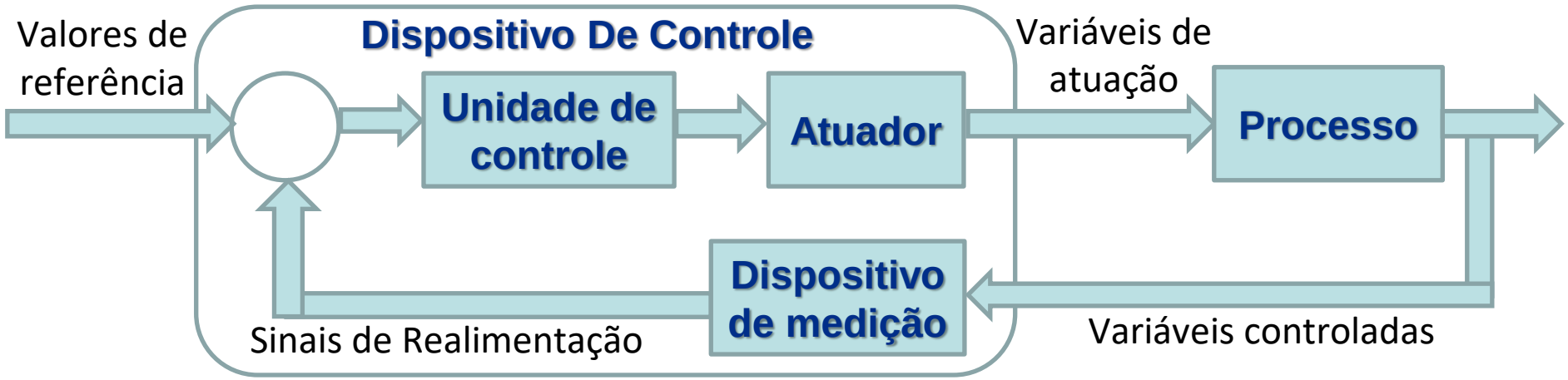


Sistema mecatrônico

- Sistema mecatrônico com controle de malha aberta:

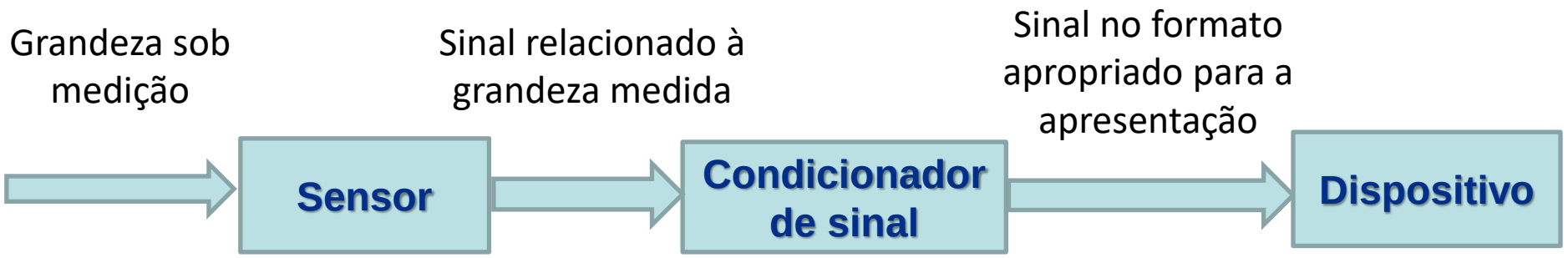


- Sistema mecatrônico com controle de malha fechada:



Sistema mecatrônico

- O dispositivo de medição pode ser modelado por:

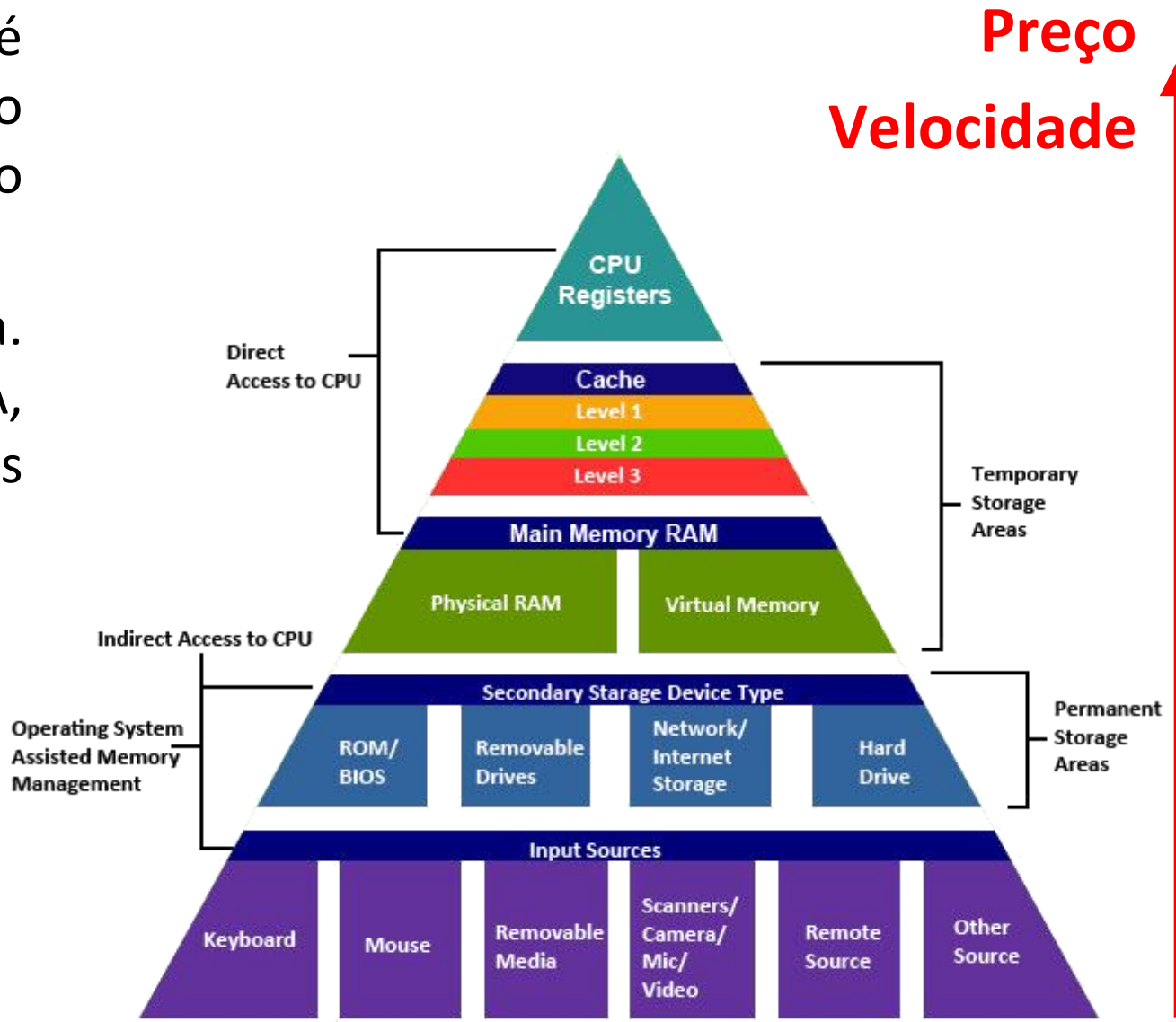


Memória

A memória é um componente cuja função é armazenar informações que serão manupuladas pelo sistema quando necessário.

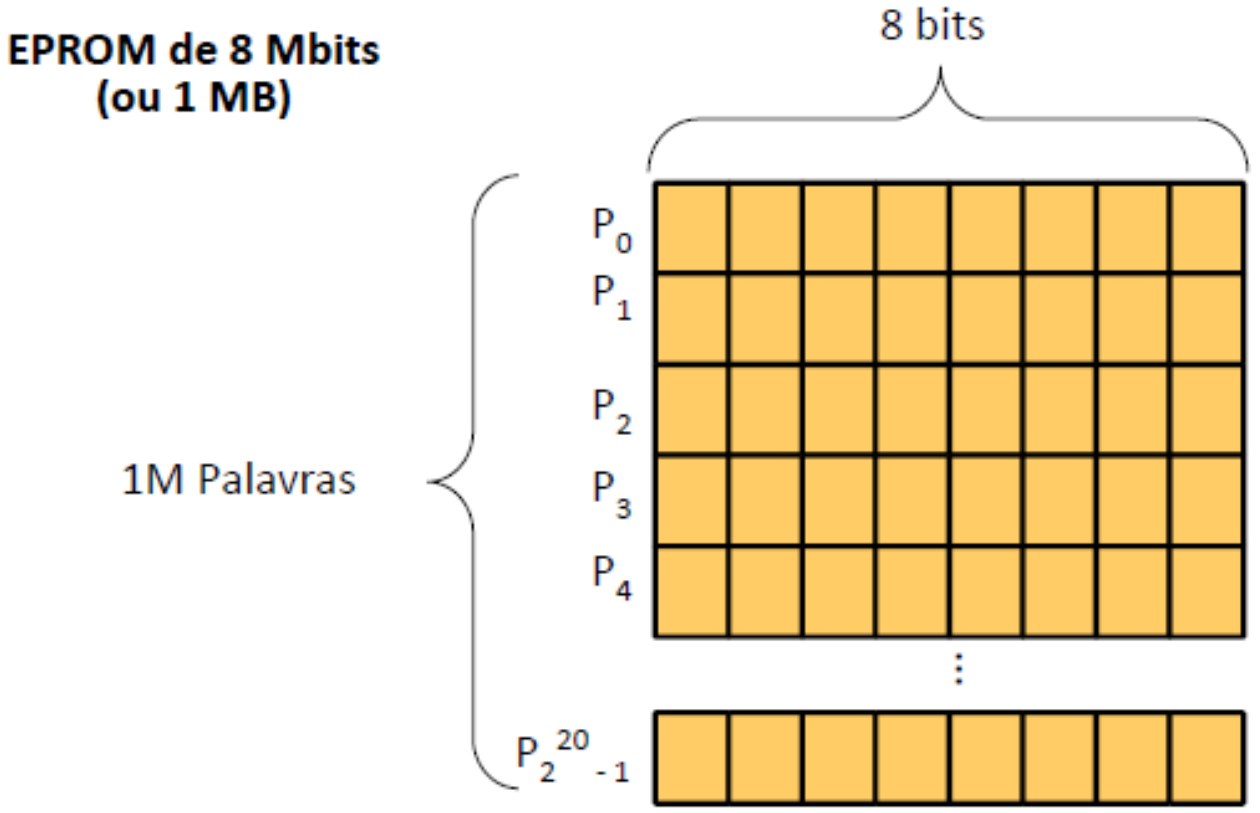
Existem vários tipos de mémoria. Geralmente, quanto mais *próxima* da ULA, mais rápida e mais cara é a memória. Alguns tipos são:

- 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 inteiro a ser armazenado, sendo neste caso 2^8-1=255.



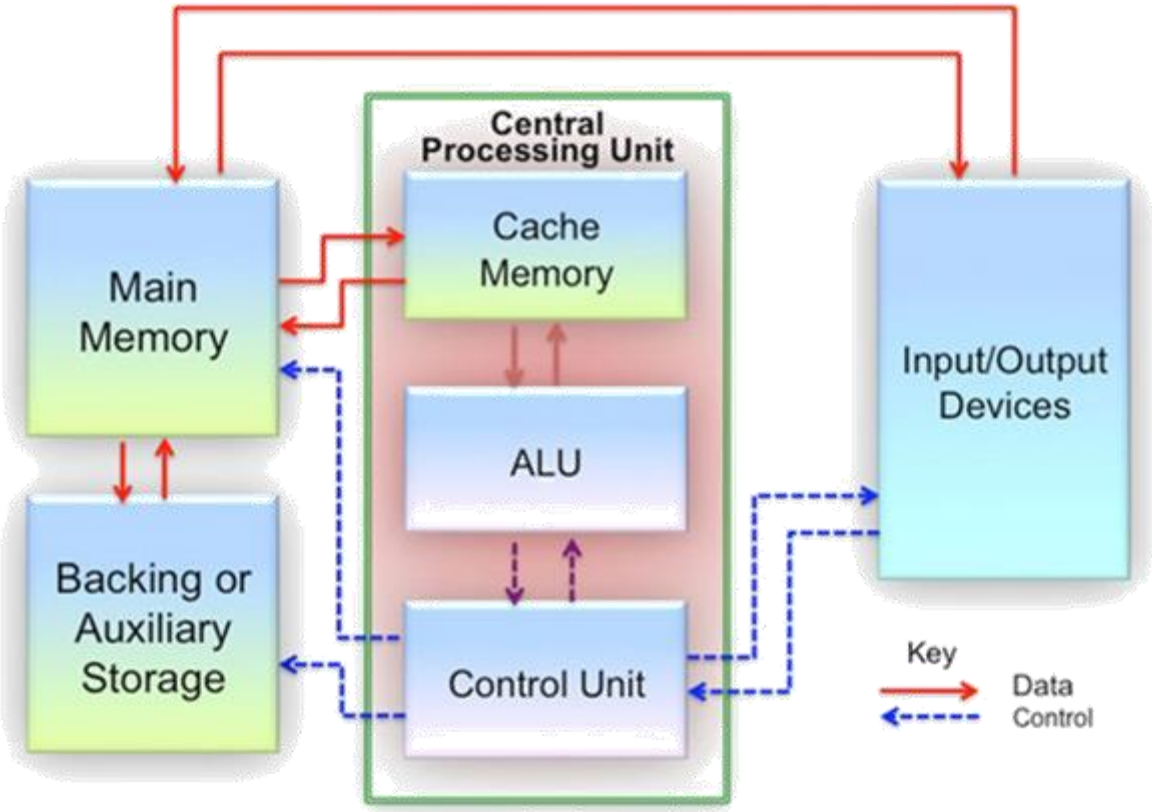
Memória – linguagem C e tipos de variável

- 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

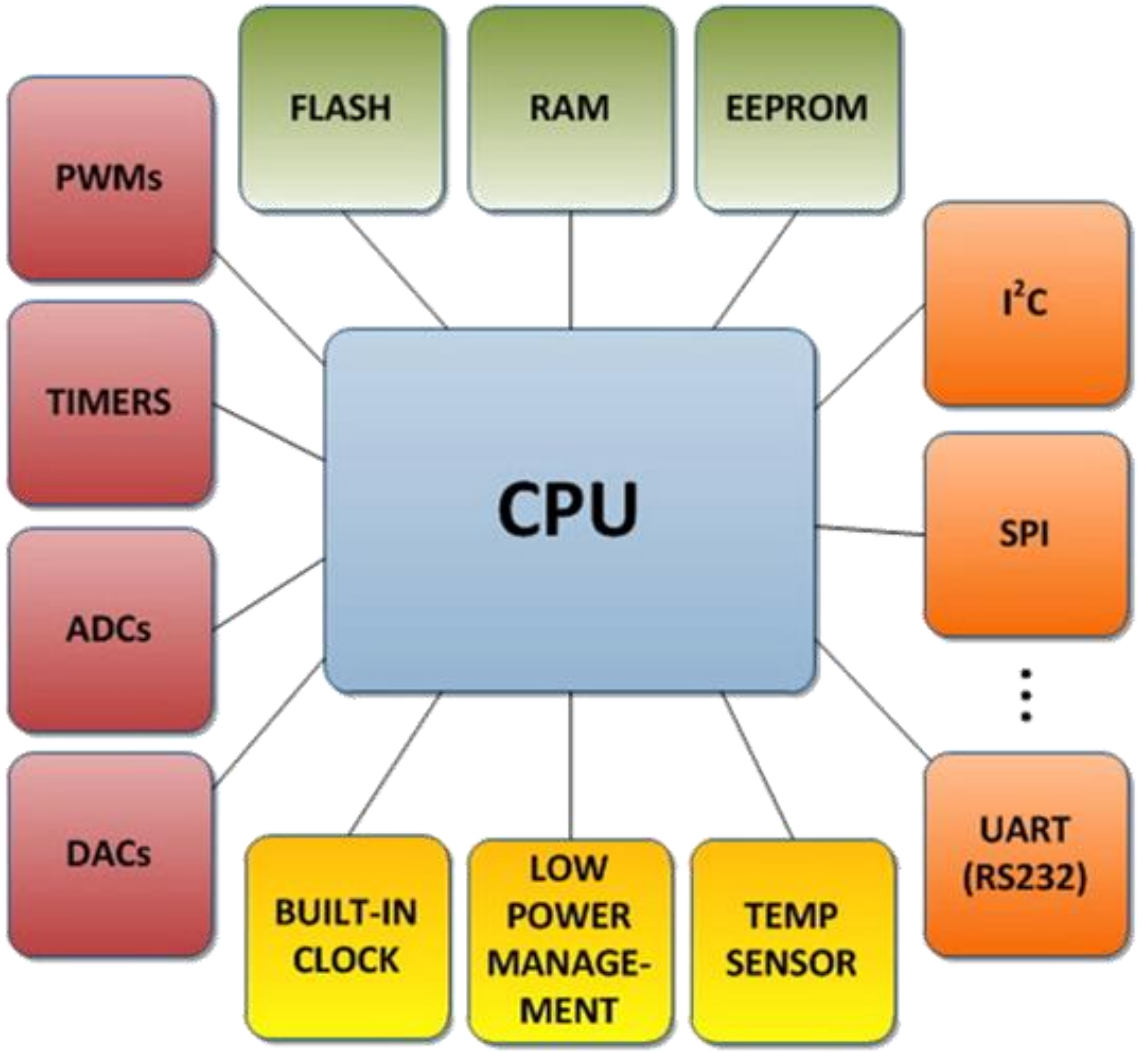
Microprocessador

- 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.



Microcontrolador

- Um microcrontolador, 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 x Microprocessador

Microcontrolador

Microprocessador

2+?=4

Arduino uno – 42mA e 7V = 0.29Watt

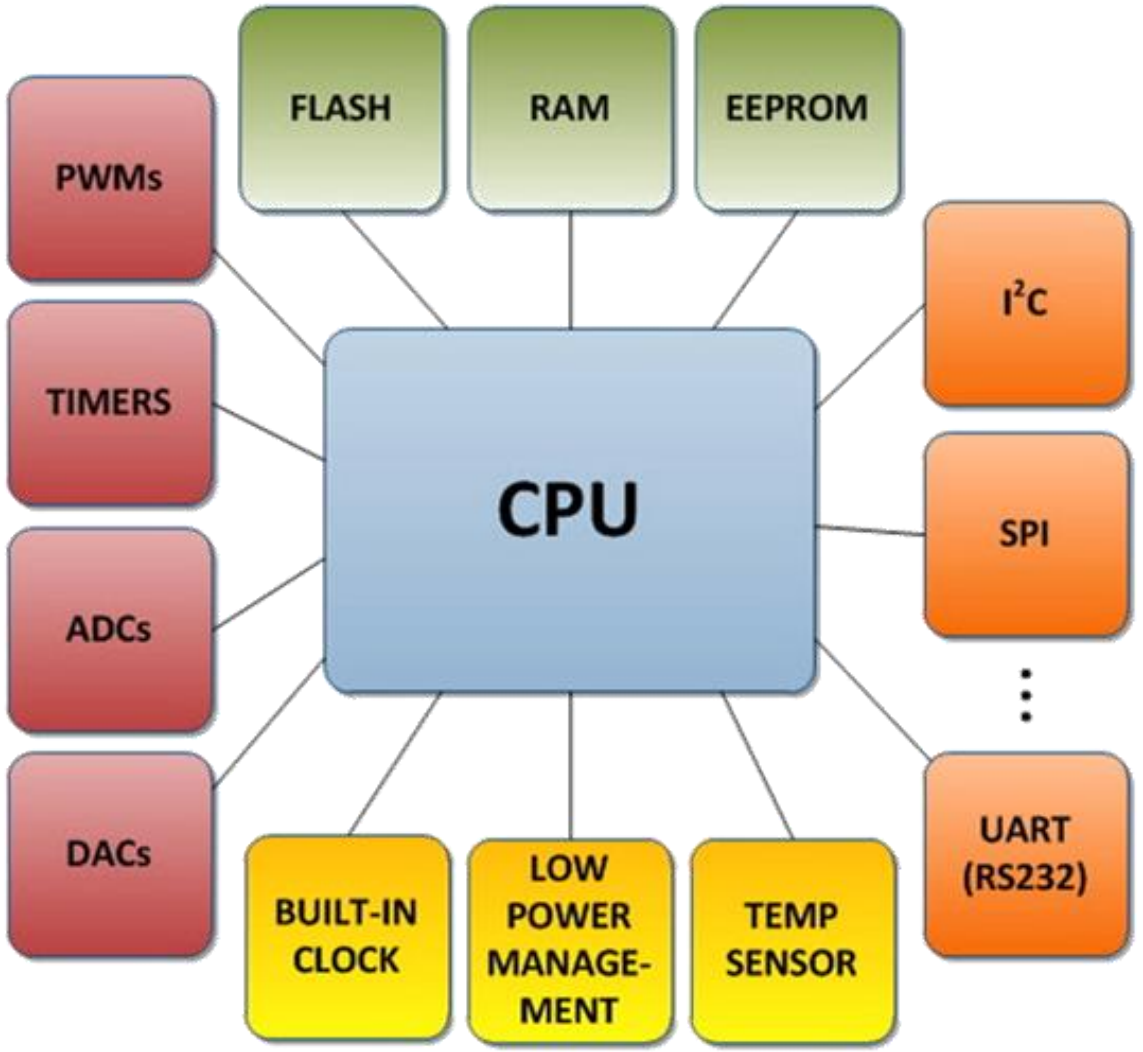
Raspberry Pi 3 – 500mA a 1 A e 5V = 2,5 a 5Watt

Arduino Uno

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

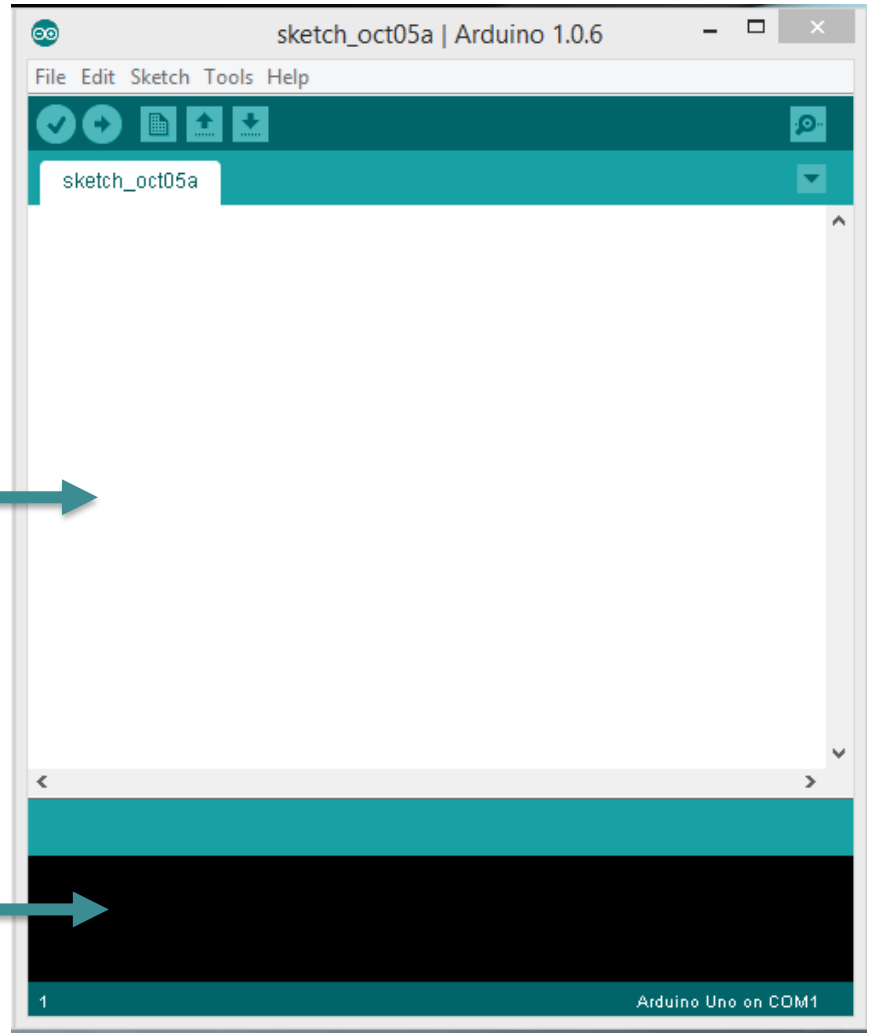
Memórias do Arduino!

- Flash – memória que guarda o sketch. 32k bytes no Arduino Uno;
- SRAM – memoria 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;



Arduino IDE

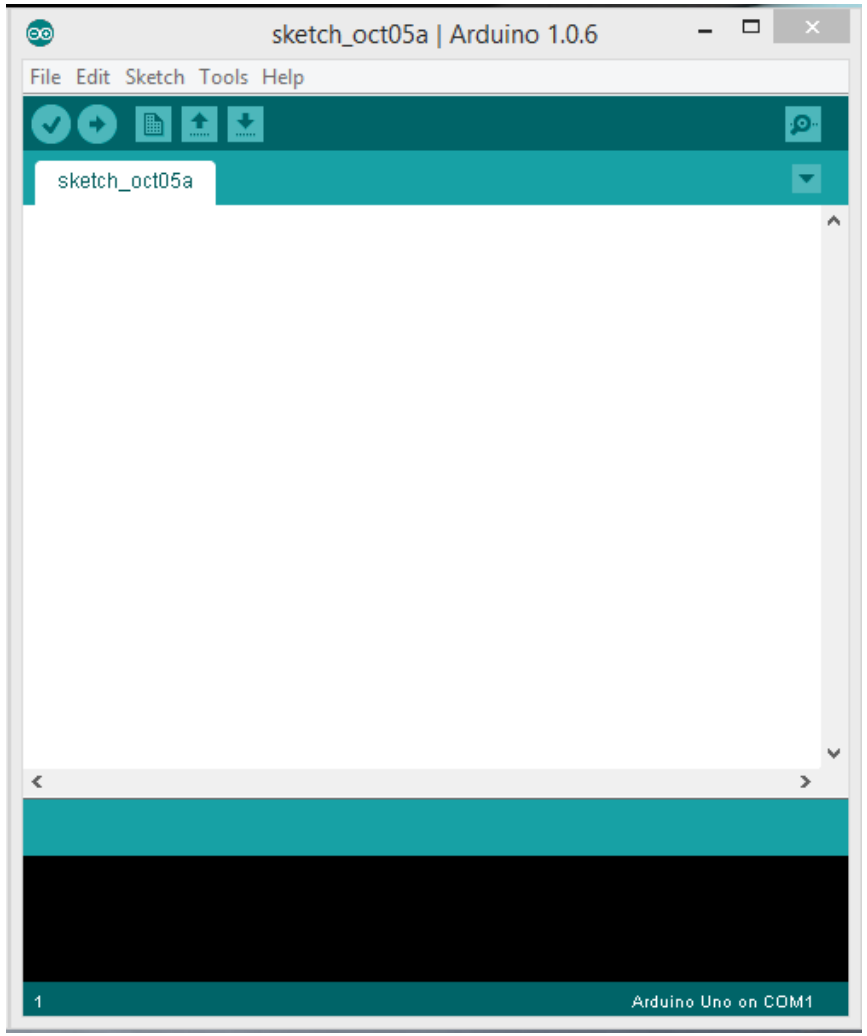
- Uma das vantagens que tornou o Arduino popular é sua IDE (Integrated Development Environment) e deve ser baixada do site: https://www.arduino.cc/en/software;
- A IDE possuir diversos elementos. O maior e mais visível é a área de **editor de texto**, parte branca, no qual o código é digitado.
- Logo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



Arduino IDE

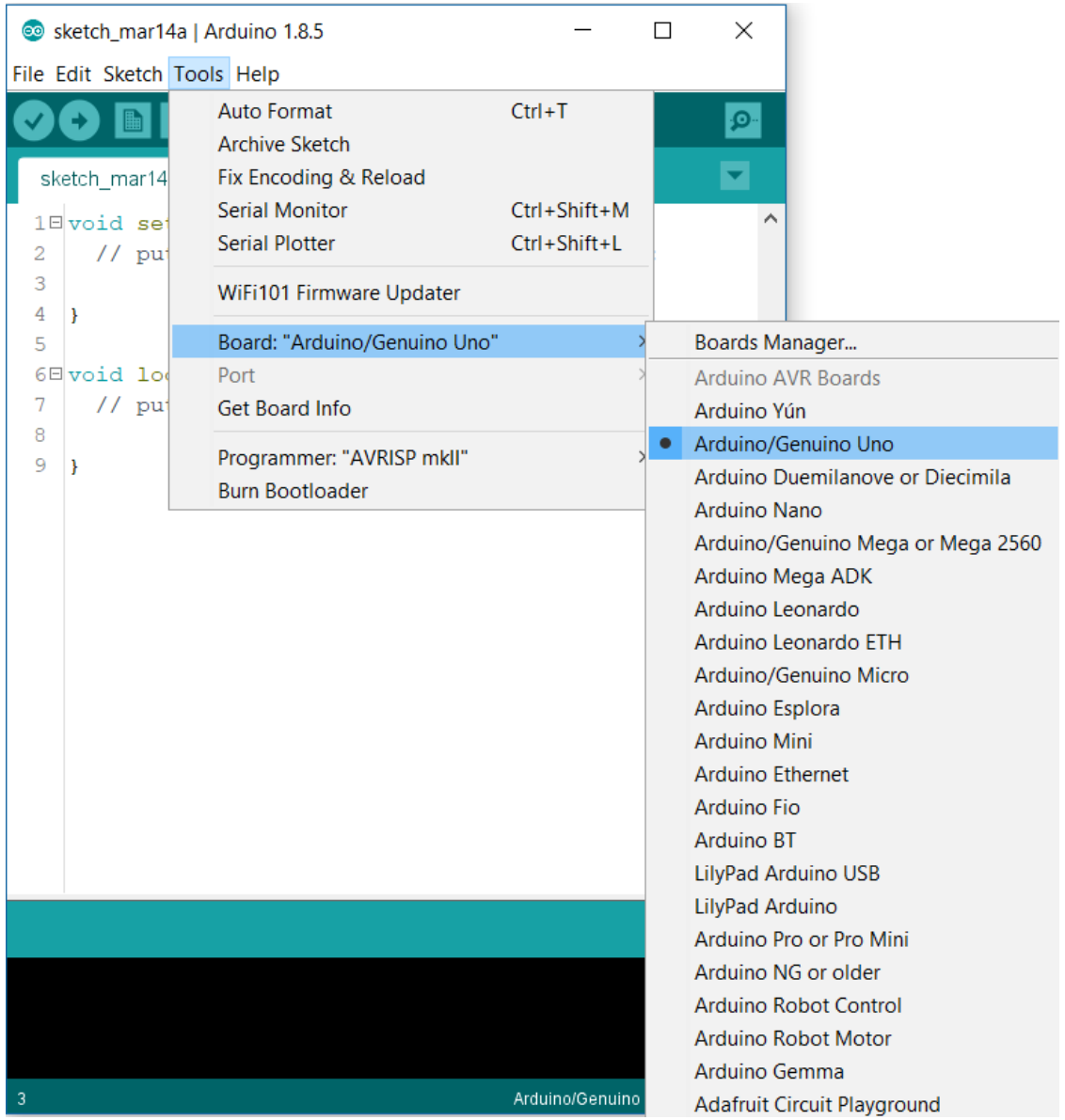
As funções dos botões de comando são:

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



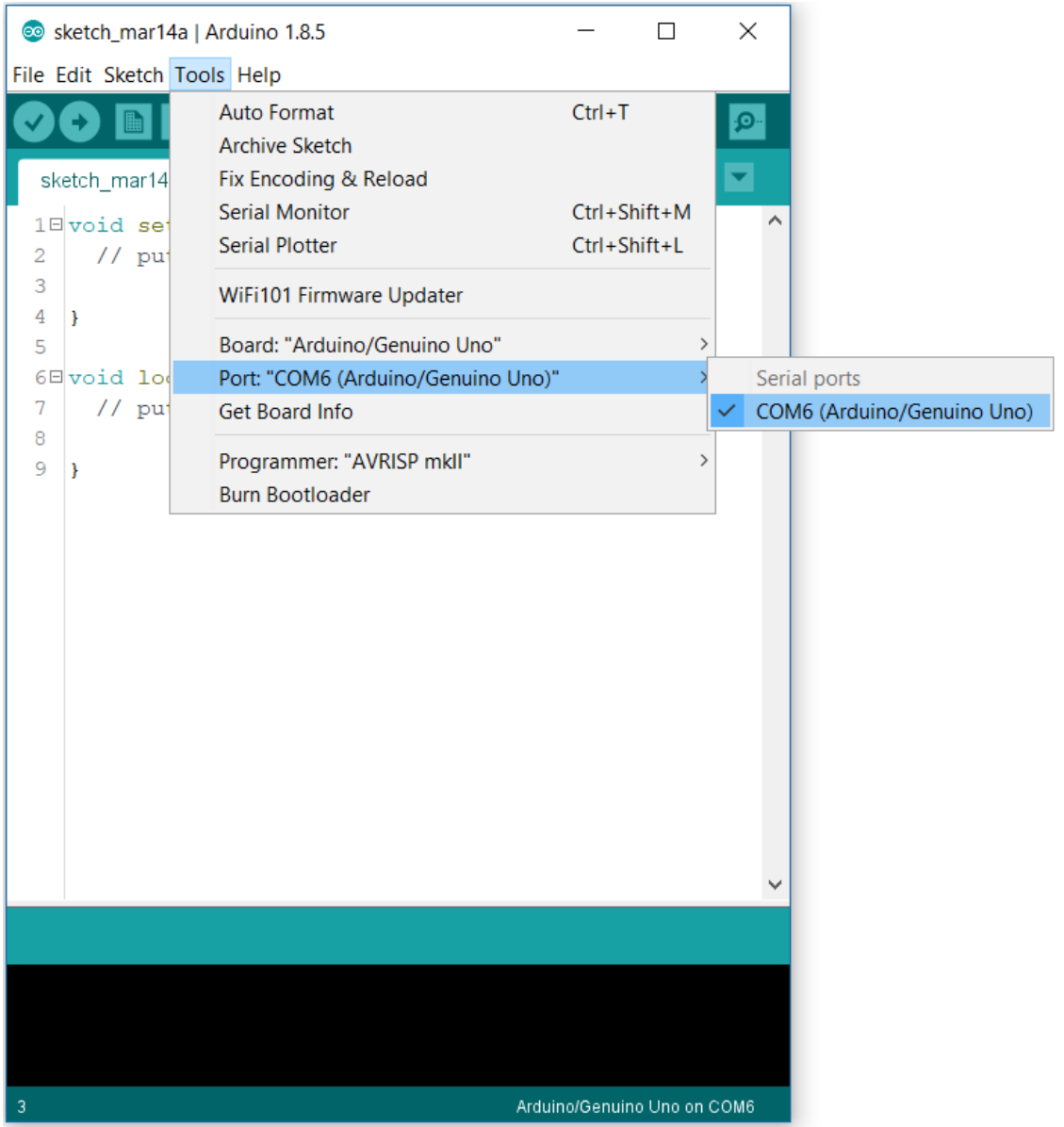
Arduino IDE – selecionando a placa correta

- 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*.



Arduino IDE – selecionando a porta de comunicação COM

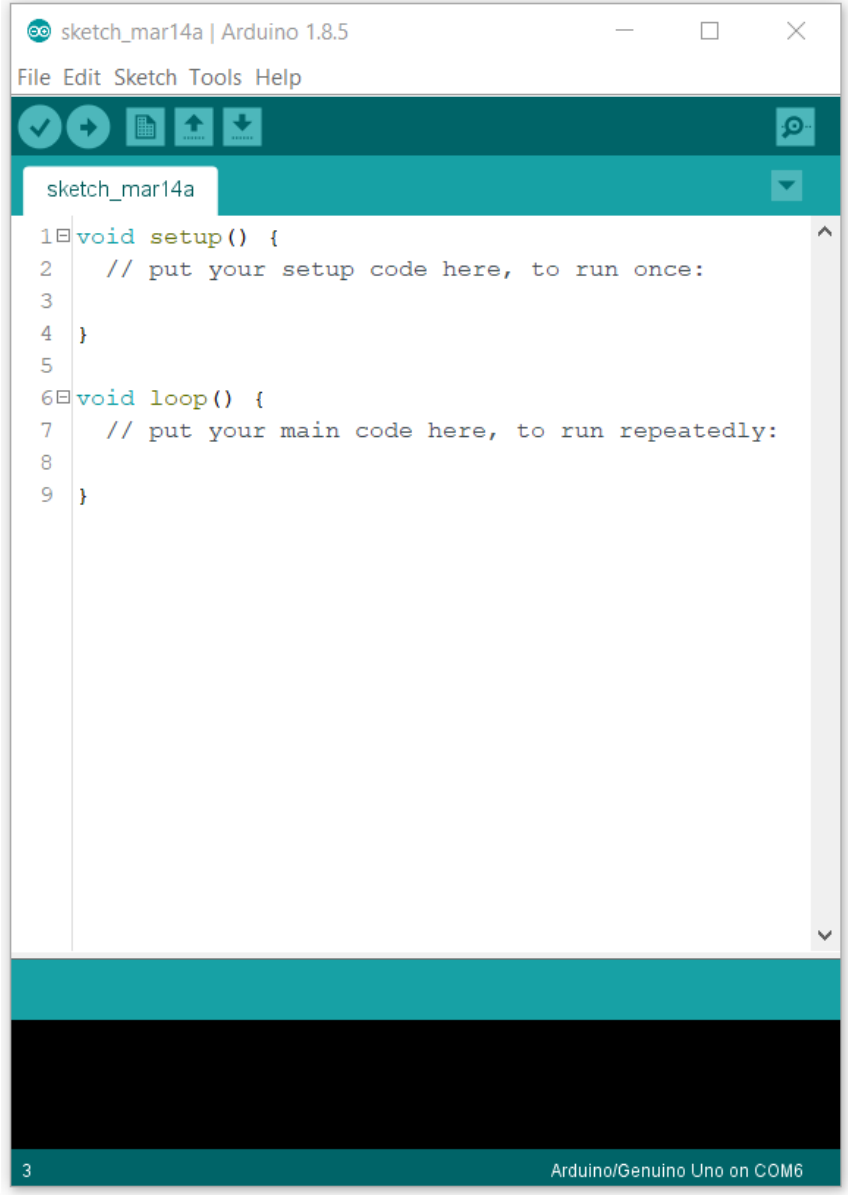
- Como você pode ter conectado mais de um Arduino ao seu computador, temos que informar à IDE em qual porta de comunicação COM está o Arduino Uno que desejamos usar;
- Clique em *Tools*, depois em *Port* e selecione a COM que estiver indicado o *Arduino/Genuino Uno*.



Arduino IDE – estrutura do sketch

- 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()**;

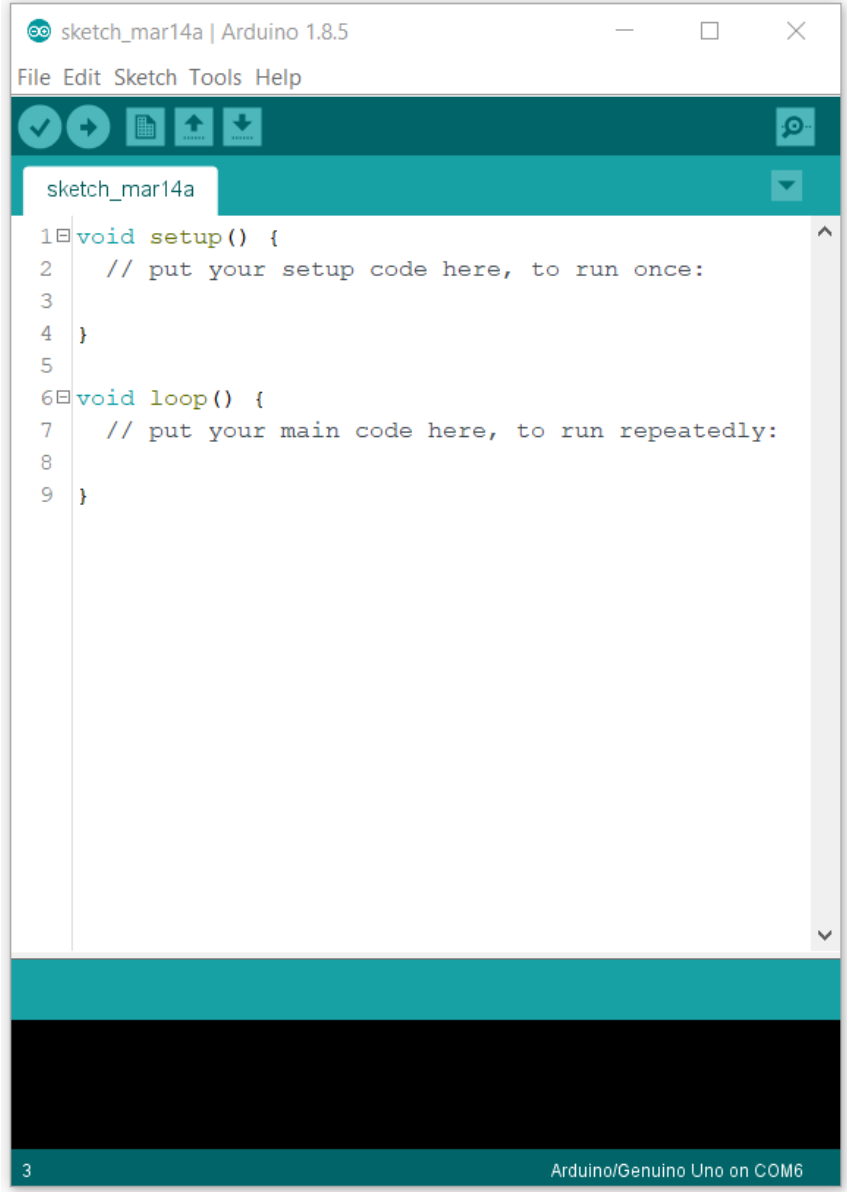
- Escopo de variáveis: É a parte na qual são declaradas as **variáveis globais** do sketch, acessíveis de qualquer função do seu código, assim como as bibliotecas utilizadas. Os tipos de dados comumente utilizados são: byte, int, float, double, boolean e char.



Arduino IDE – estrutura do sketch

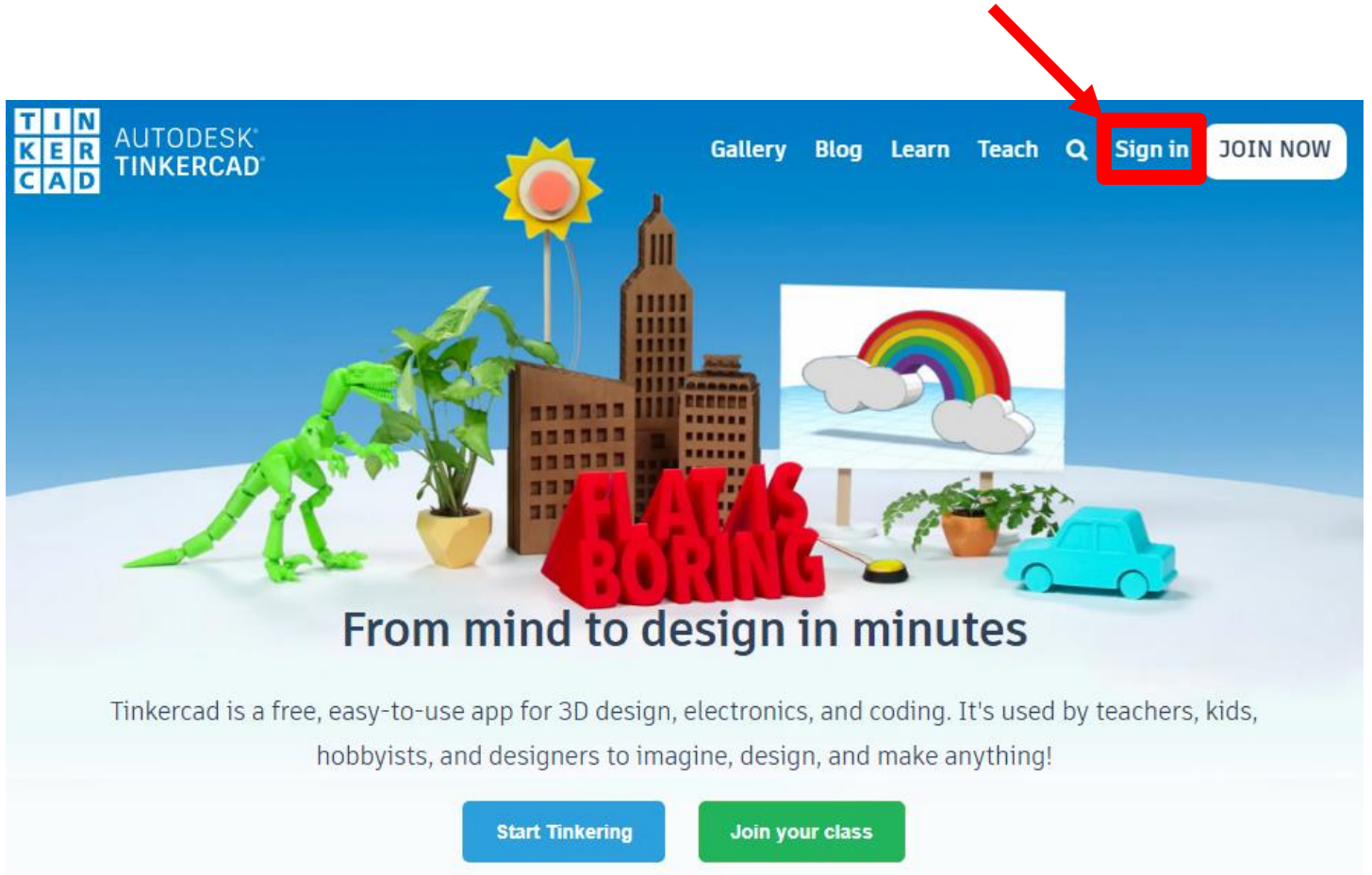
- Função **setup()**: é chamada uma vez apenas na execução do programa, normalmente no início da execução. É onde se especifica, por exemplo, se um pino é de entrada ou saída e são inicializadas algumas variáveis.

- **loop()**: a função será executada em loop até que o programa seja terminado.

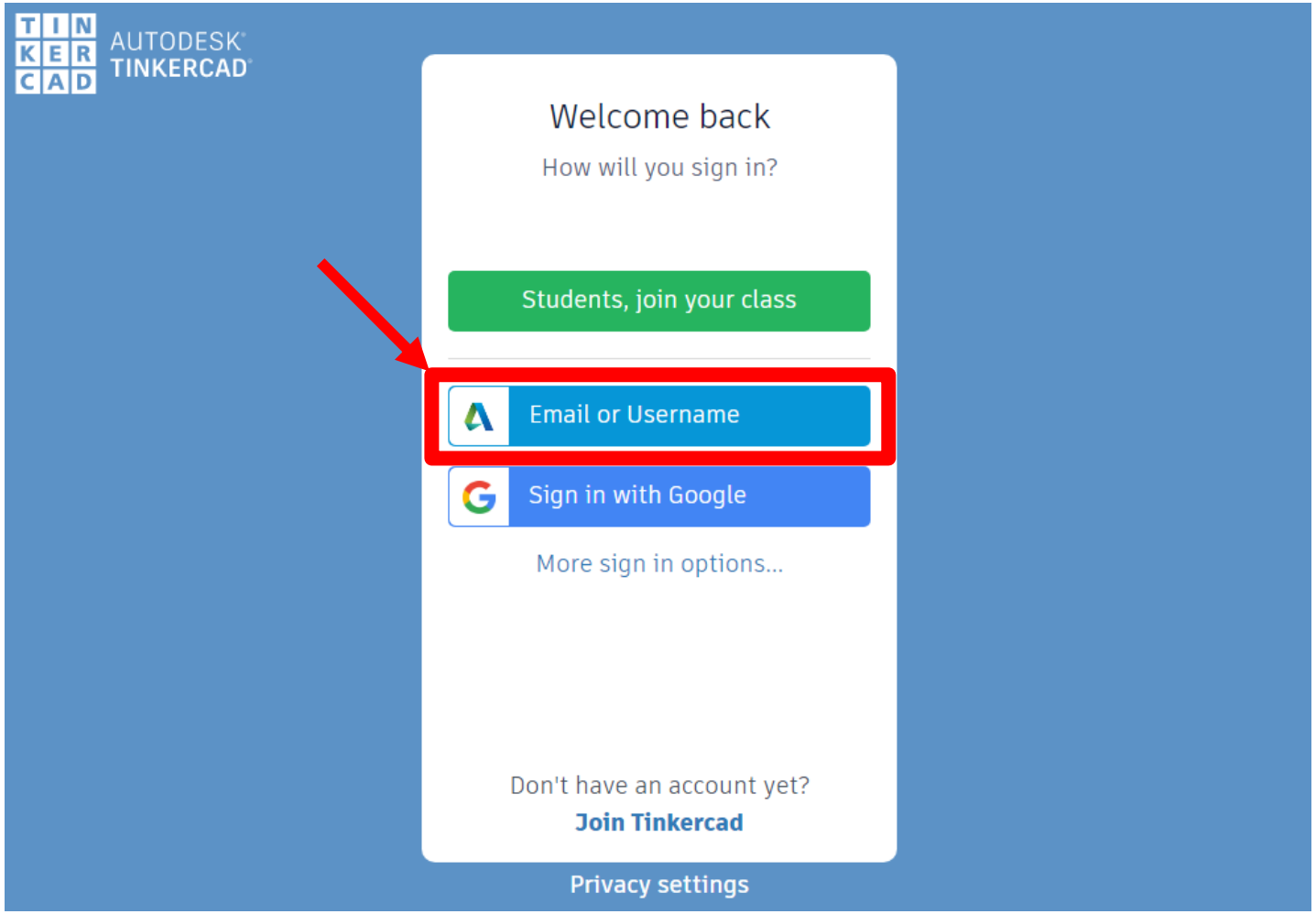


TinkerCAD

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

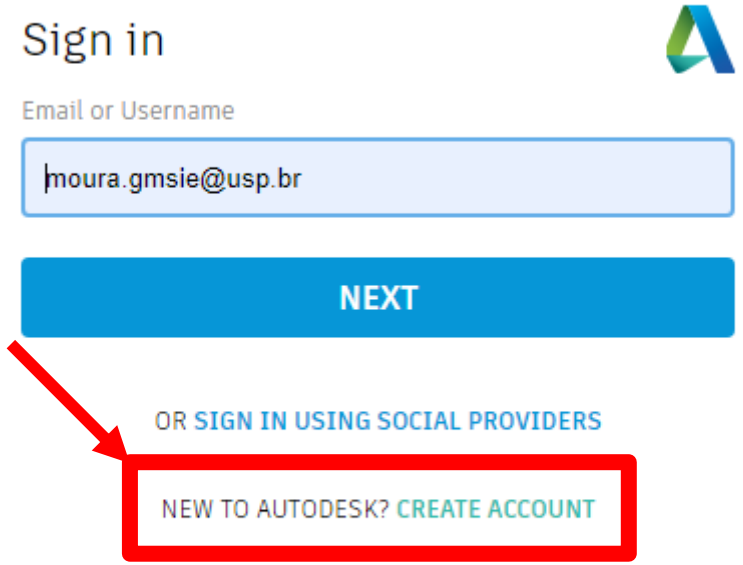


Clique em **Email or Username**, conforme figura ao lado.

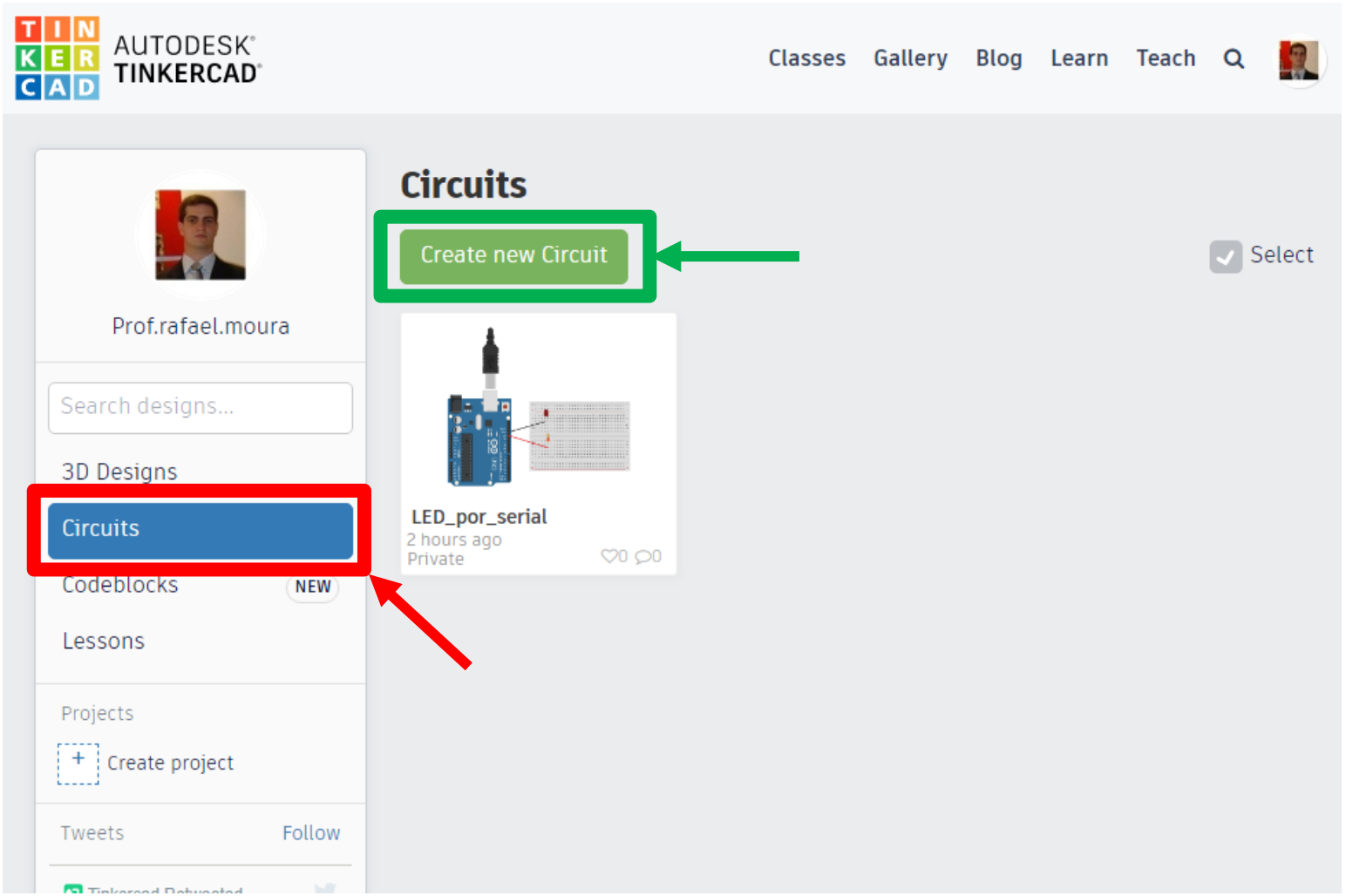


Caso já tenha feito uma conta, faça o login normalmente.

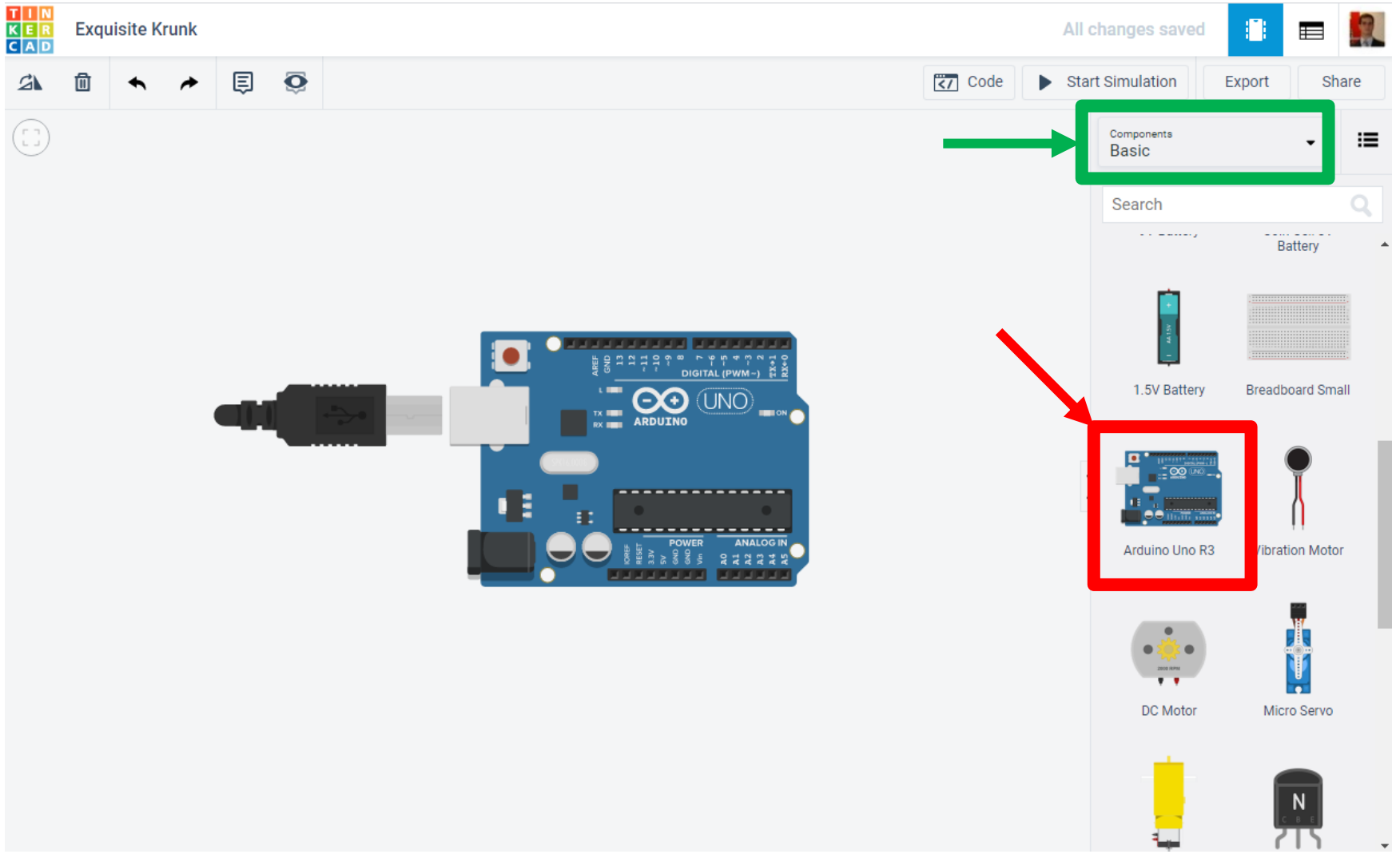
Caso não tenha criado uma conta antes, clique em **Create Account**, conforme figura ao lado. Então refaça os passos anteriores e faça o Login.



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

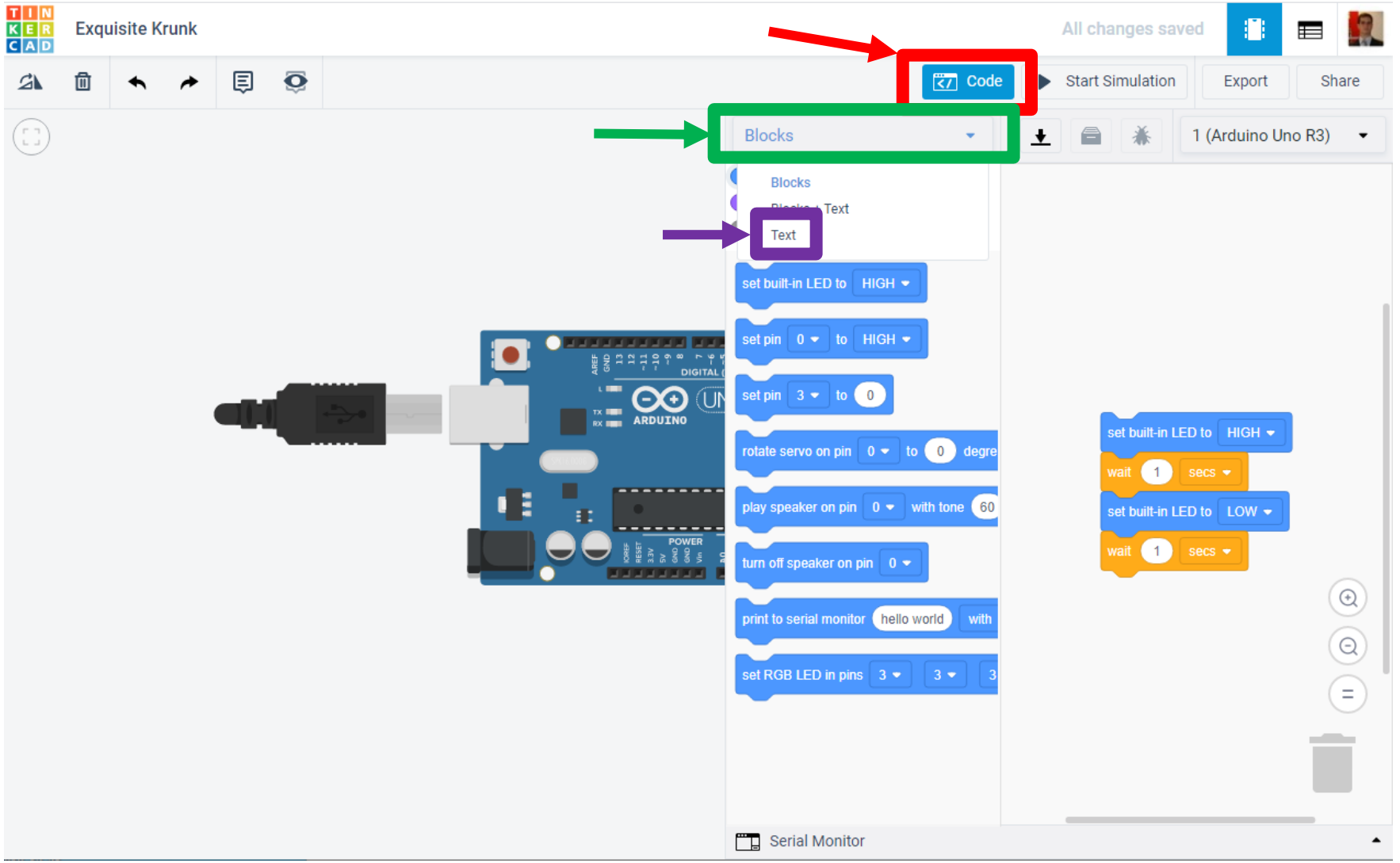


Arraste um **Arduino Uno R3** dos itens de **Components Basics** para a área na esquerda.



Clique em **Code**, depois em **Blocks** e selecione **Text**.

Surgirá um pop-up. Nele, selecione **Continue**.



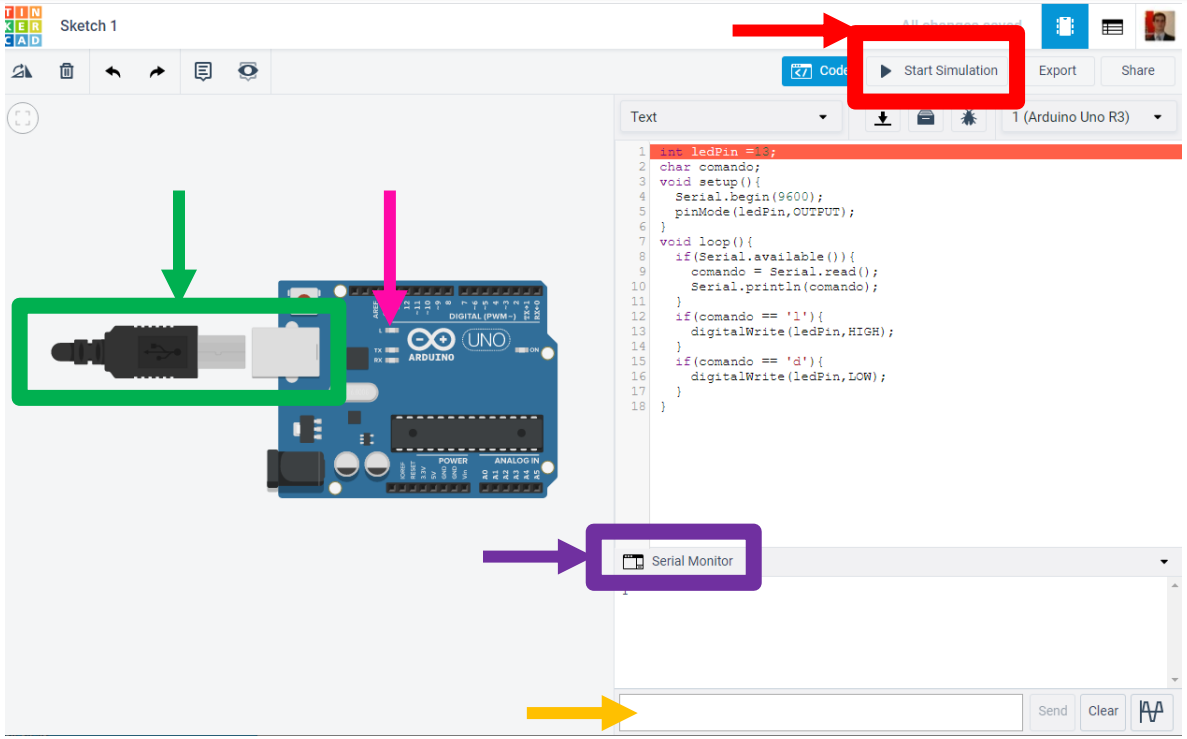
Exercício 01 – Comunicação Console – Sketch_01

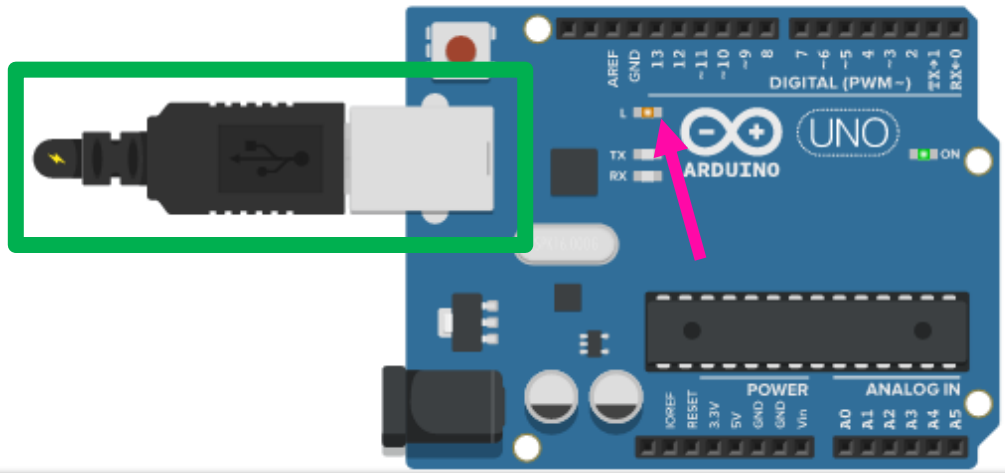
Digite o código e clique em **Start Simulation.**

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 ***l*** (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 == 'l'){
    digitalWrite(ledPin,HIGH);
  }
  if(comando == 'd'){
    digitalWrite(ledPin,LOW);
  }
}
```



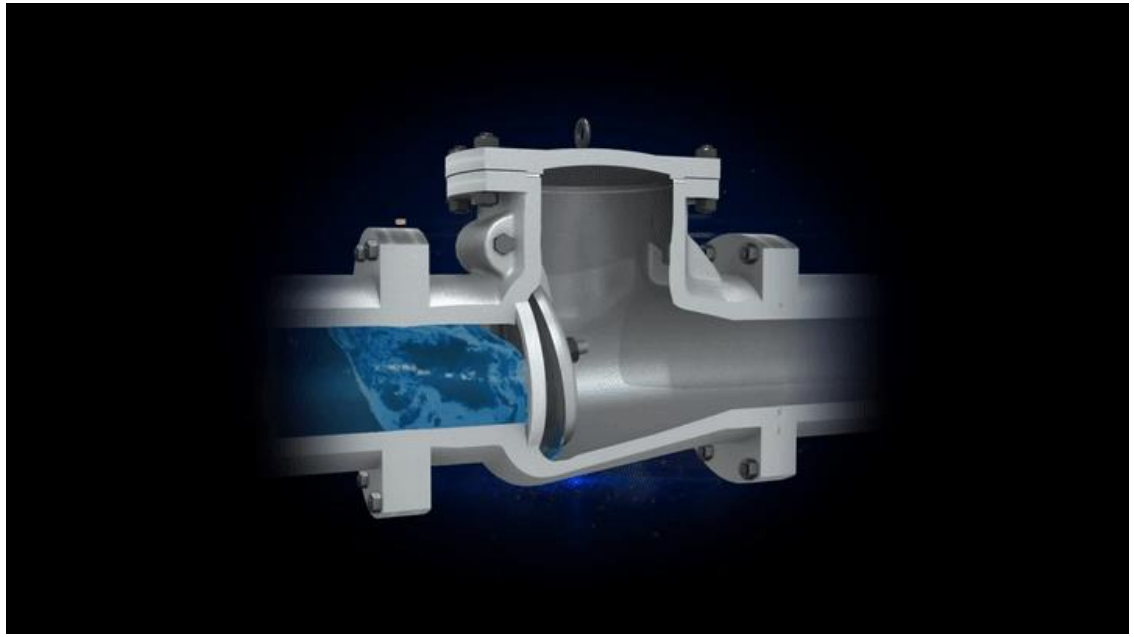


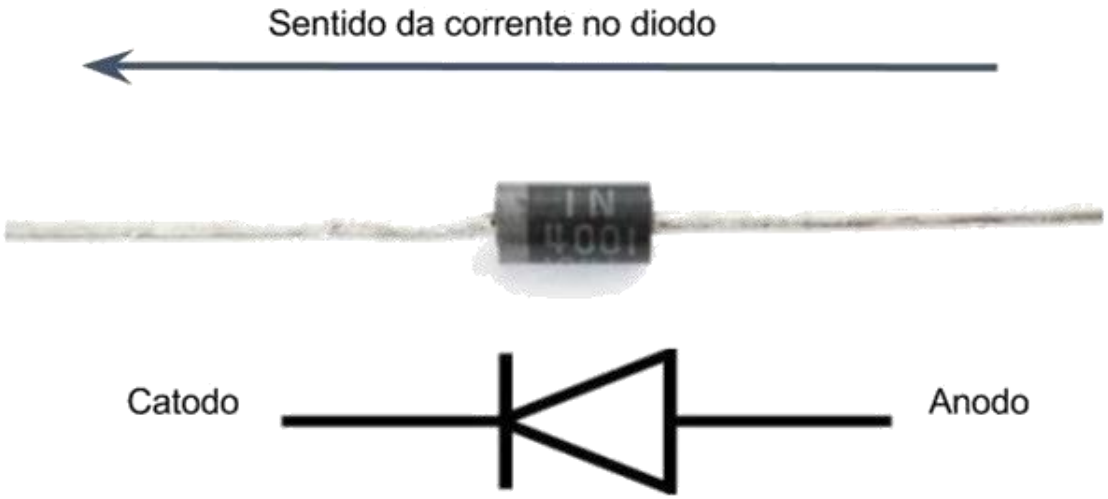
Arduino IDE – Comunicação Console

- 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 e realizada quando há dados disponível. A quantidade máxima de bytes no buffer é 64.
- `pinMode()` Configura um pino especifico 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.

Eletrônica Básica – Diodo

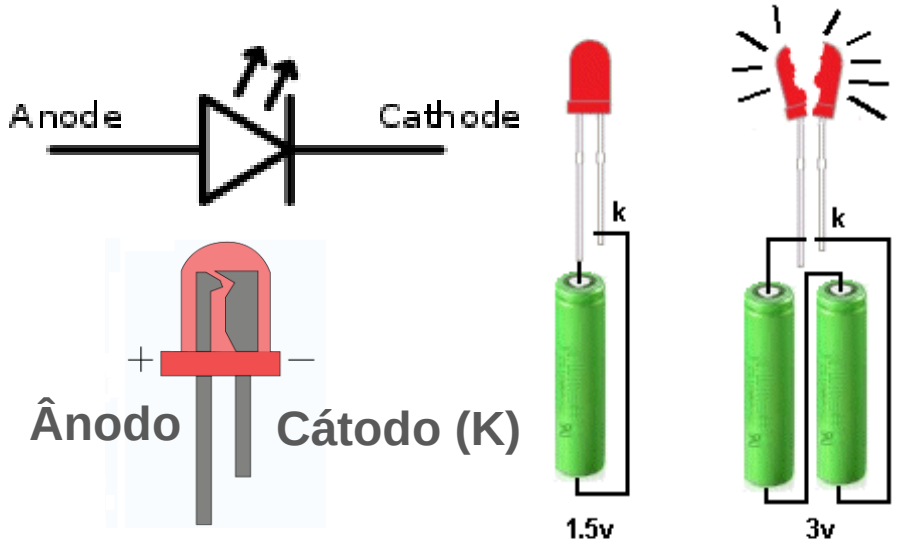
- O diodo é um componente eletrônico que permite corrente em uma direção, mas a impede de fluir na outra direção.
- No diodo, existe uma queda de voltagem constante. O mesmo possui uma potência máxima, e, consequentemente, uma corrente máxima. O encapsulamento do diodo está diretamente relacionado com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

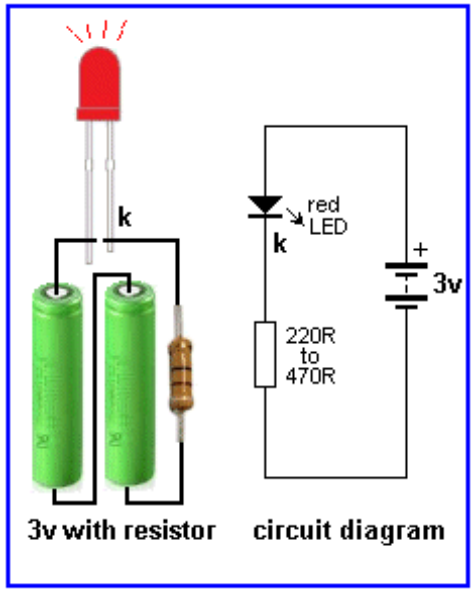




Eletrônica Básica – LED

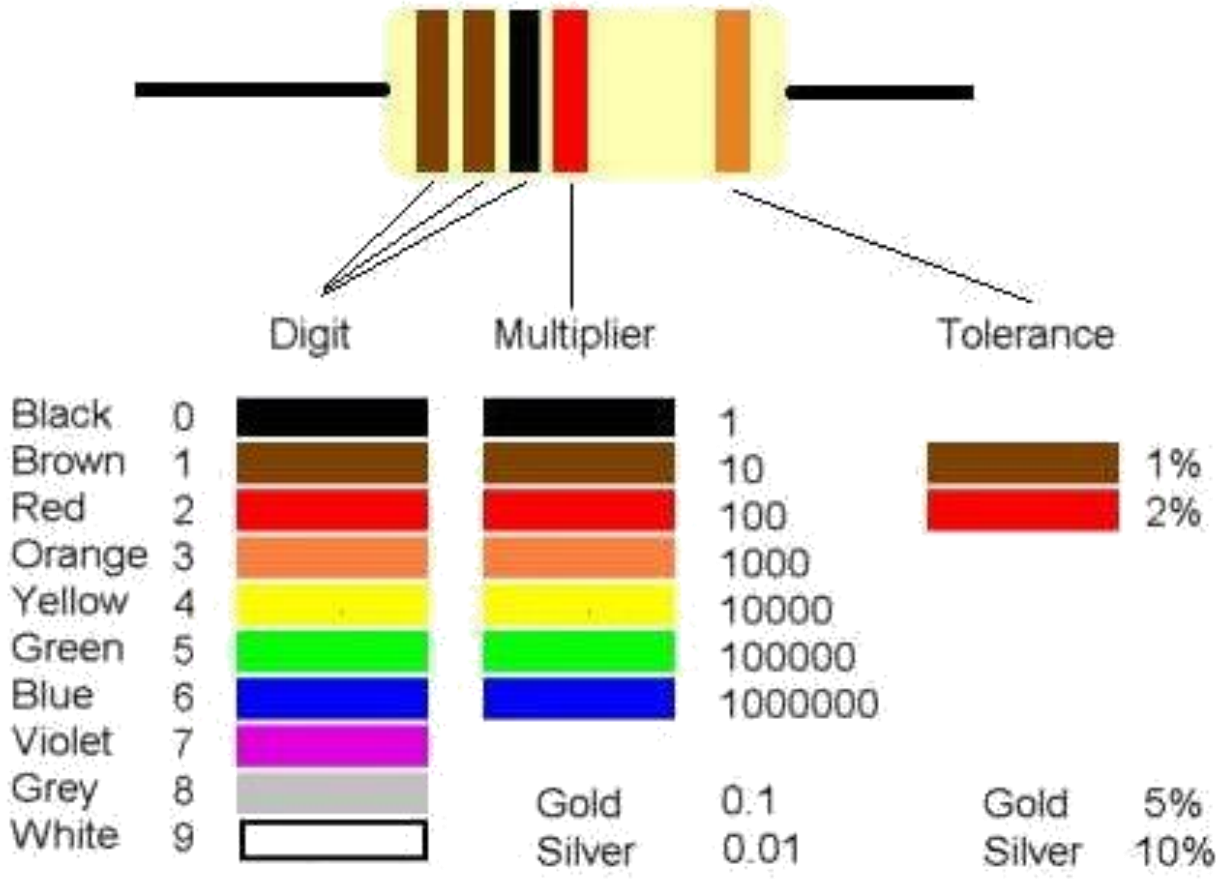
- LED significa Light Emitting Diode. Ou seja, ele é um diodo que emite uma luz quando há passagem de corrente. Assim como nos diodos, podemos considerar que há uma queda de voltagem constante no LED, independente da corrente.
- O LED possui uma potência máxima suportada. Acima, o mesmo queima. Como a queda de voltagem 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, consequentemente, com a potência e conseguinte corrente máximas.

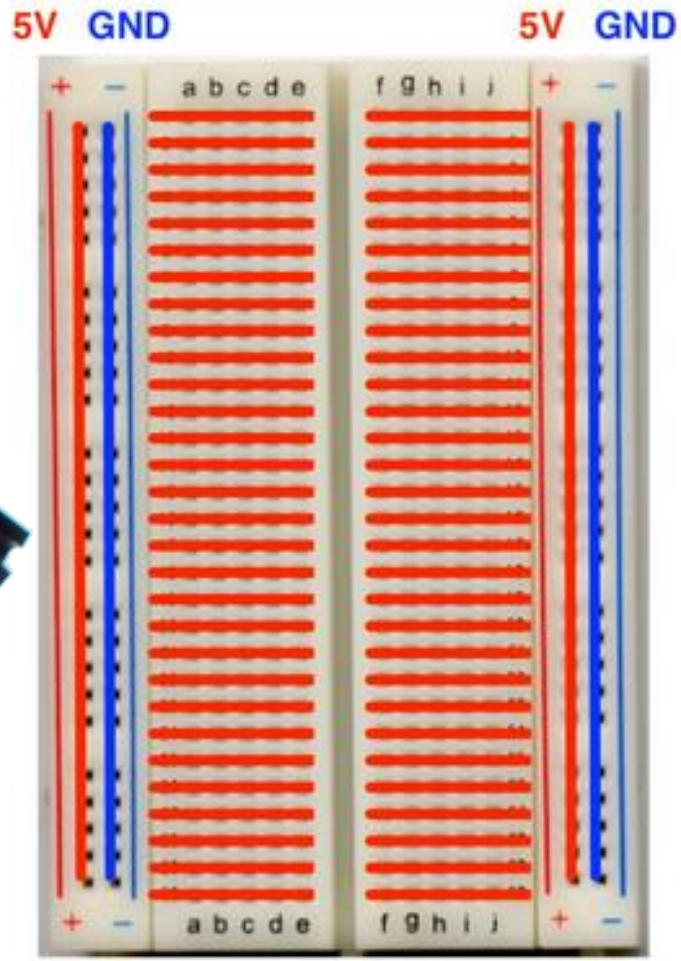


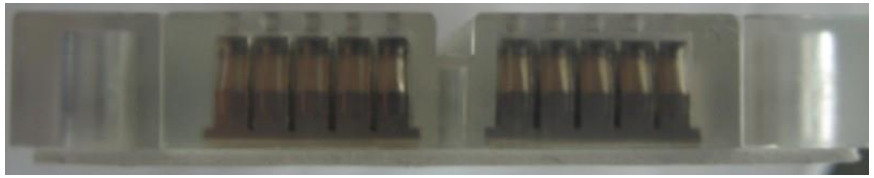
Eletrônica Básica – Calculado resistores em série com LEDs

- 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 Ω;
- 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 Ω e 1k Ω.

Eletrônica Básica – Protoboard (breadboard)

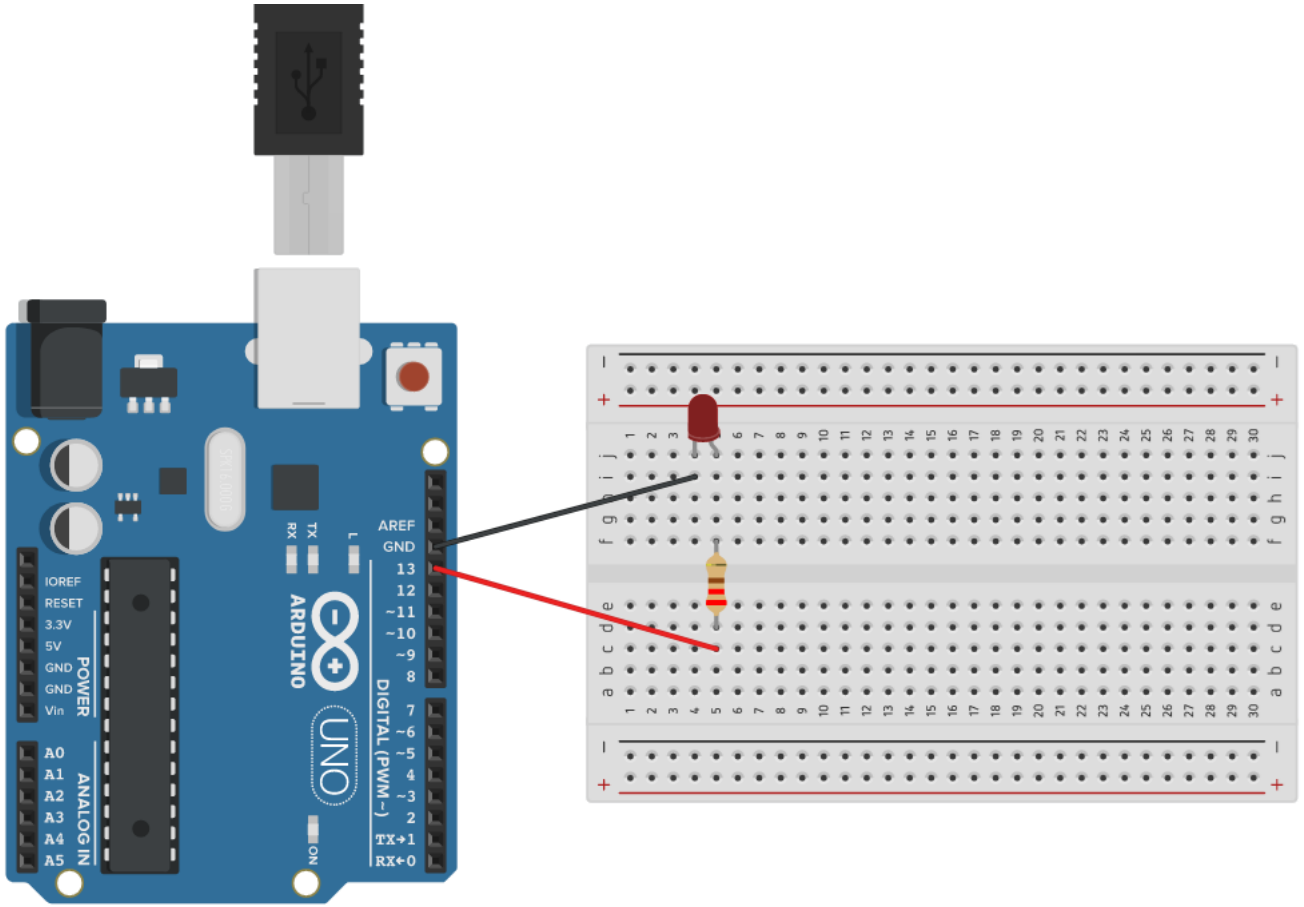
- Mas como podemos interligar os componentes de maneira que seja prático alterar e testar circuitos, ou seja, sem soldar os componentes?
- Para tal utilizamos tanto a protoboard como cabos jumper. A protoboard tem furos conectados entre si conforme a imagem ao lado.
- Já os cabos jumper servem para facilitar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





Exercício 02 – Sketch_01

- 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.



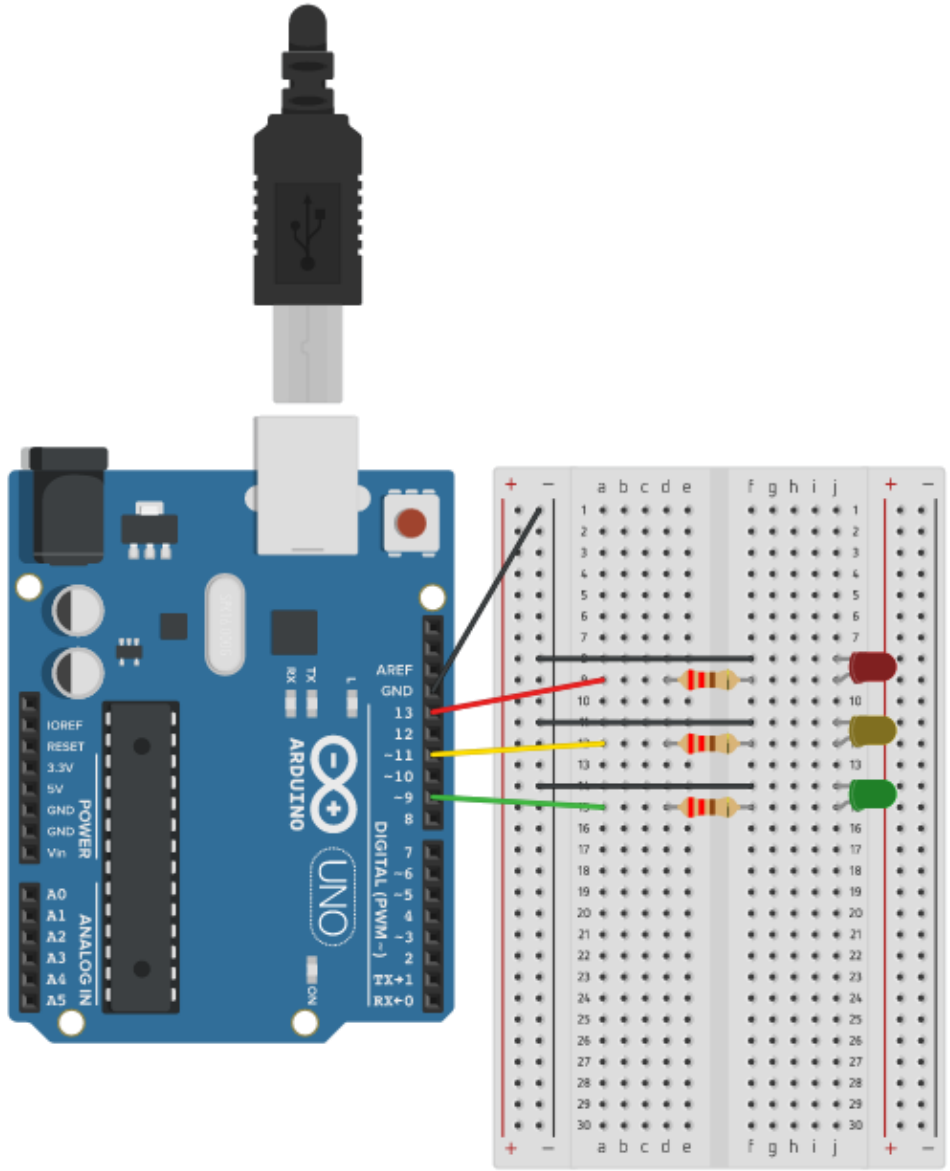
Exercício 03 – Farol – Sketch_02

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.

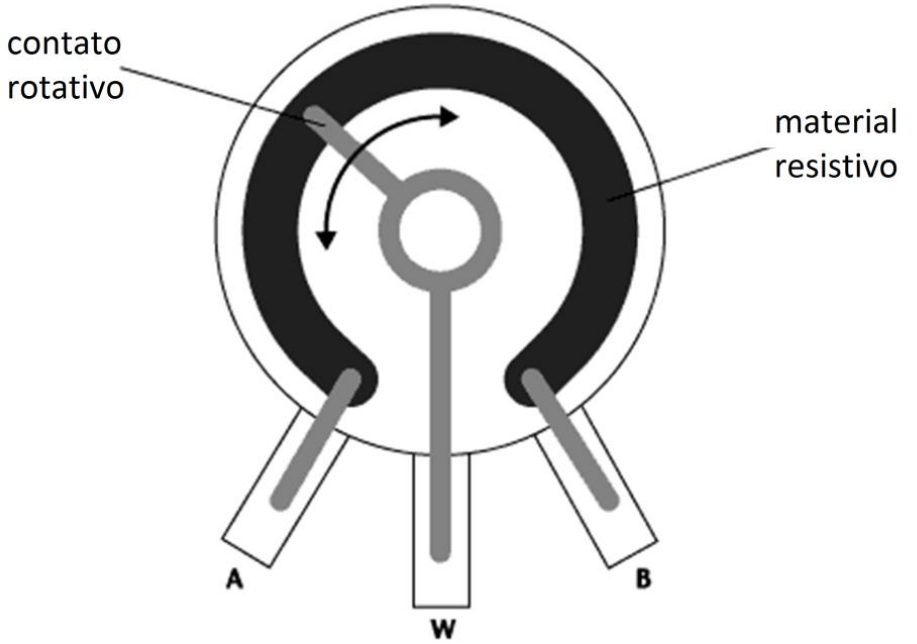
Atenção para utilizar um terra comum nos três LEDs.

```
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);
}
```



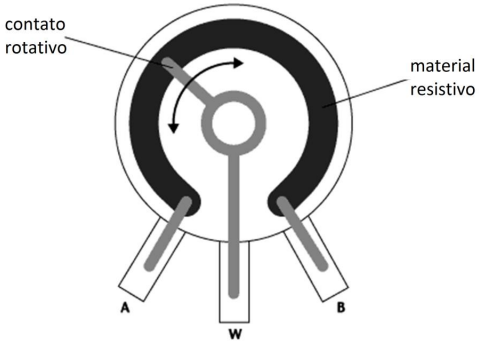
Eletrônica Básica – Potenciômetro

- 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 analogico-digital. No arduino, como o ADC é de 10 bits, temos 1024 níveis lógicos: de 0 a 1023.



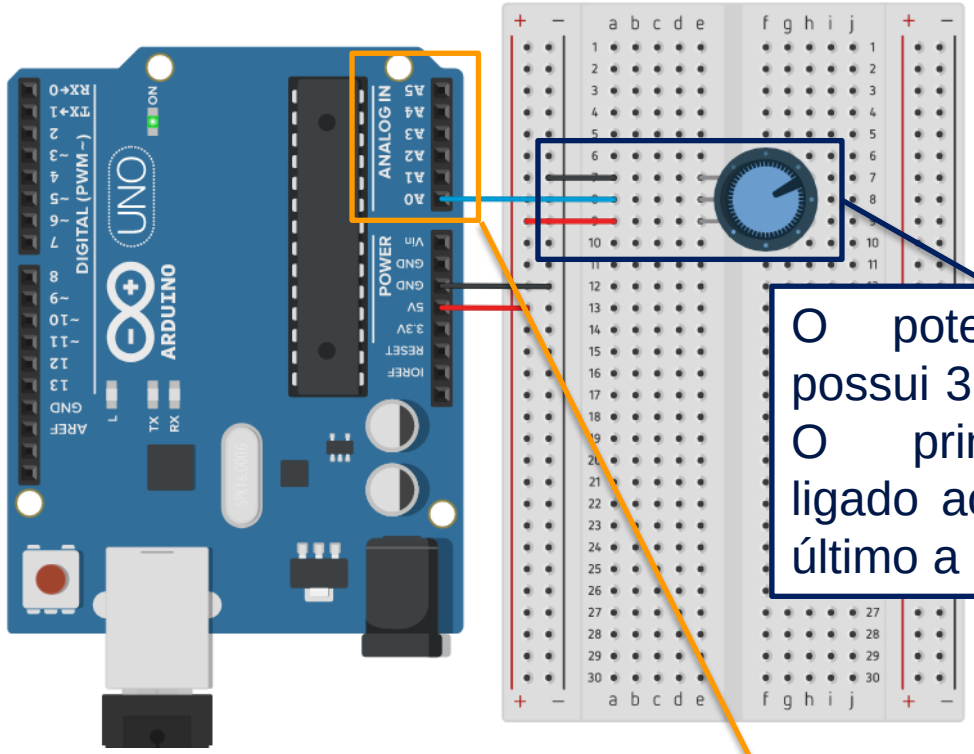
Eletrônica Básica – Potenciômetro

- 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.

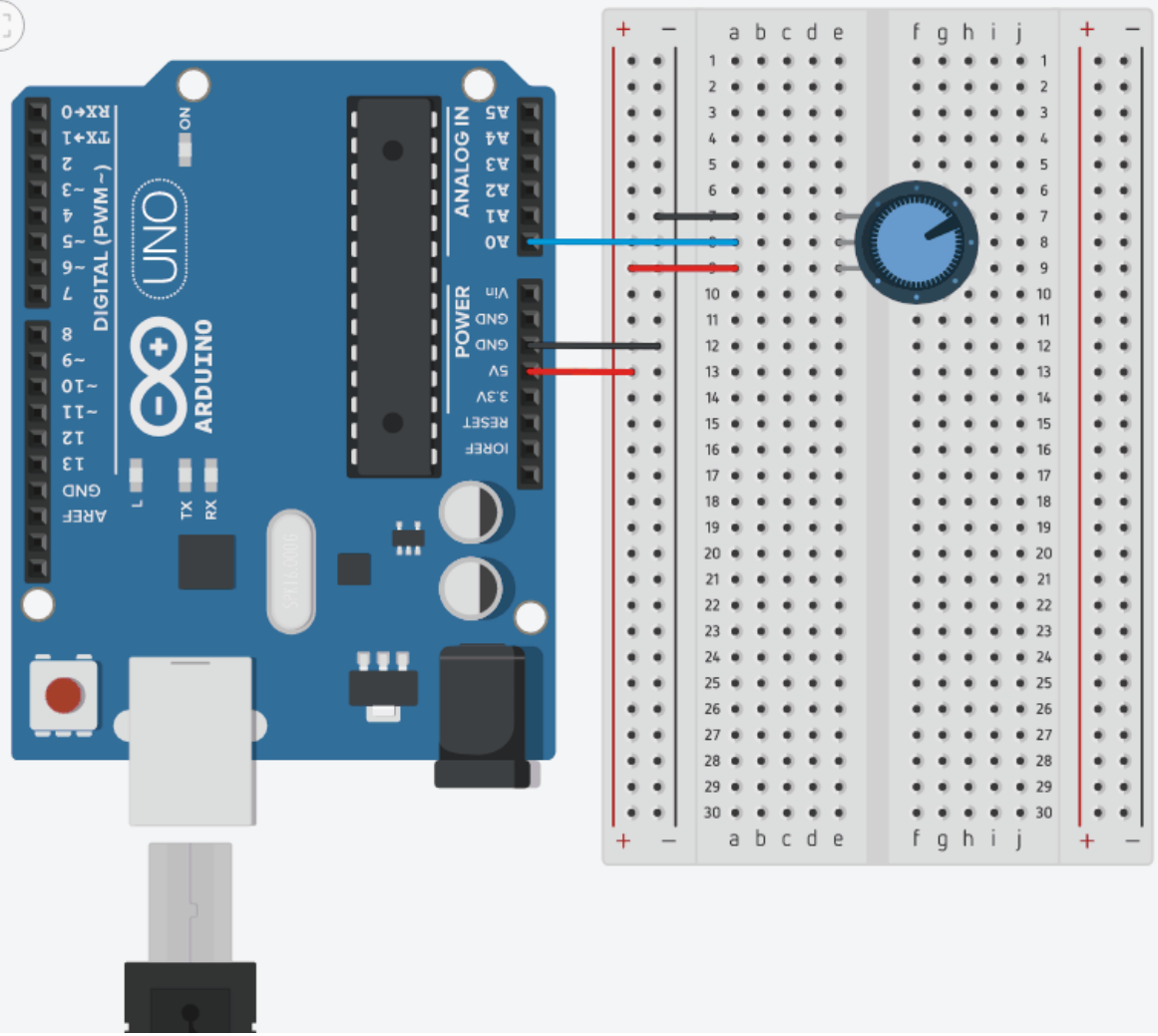
O terminal central do potenciômetro deve ser ligado ao pino analógico, que pode assumir valores diversos, e não apenas 0 e 1 como o caso do pino digital. *Internamente, existe um conversor analógico-digital, que converte os 5V em 1024 níveis, indo de 0 a 1023.*

Exercício 04 – potenciômetro – Sketch_03

Sketch_03

All changes saved

Code | Start Simulation | Export | Share



Text

1 (Arduino Uno R3)

```
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);
}

```

Serial Monitor

Send | Clear

Eletrônica Básica – Sensor para seguidor de linha

- 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 mecatrônica, 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.

Exercício 05 – sensor de linha – **Sketch_04**

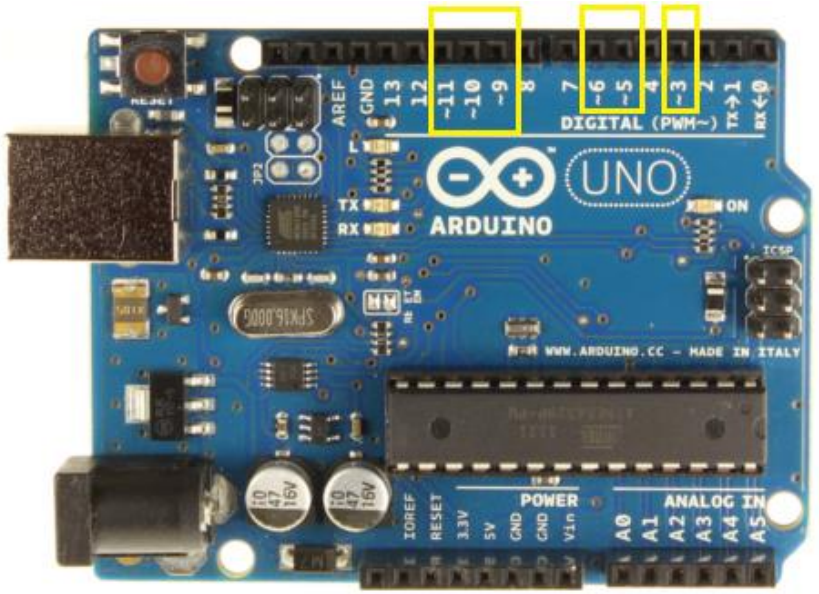
```
int sensorPin = A0;
int sensorValue = 0;
int emissorPin = 2;
void setup() {
 Serial.begin(9600);
 pinMode(emissorPin,
OUTPUT);
 digitalWrite(emissorPin,
HIGH);
 }
void loop() {
 sensorValue =
analogRead(sensorPin);
 Serial.print("Nivel Logico ADC:
");
 Serial.println(sensorValue);
}
```

- Ligue os 4 fios do módulo se 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?

Acionamentos – PWM

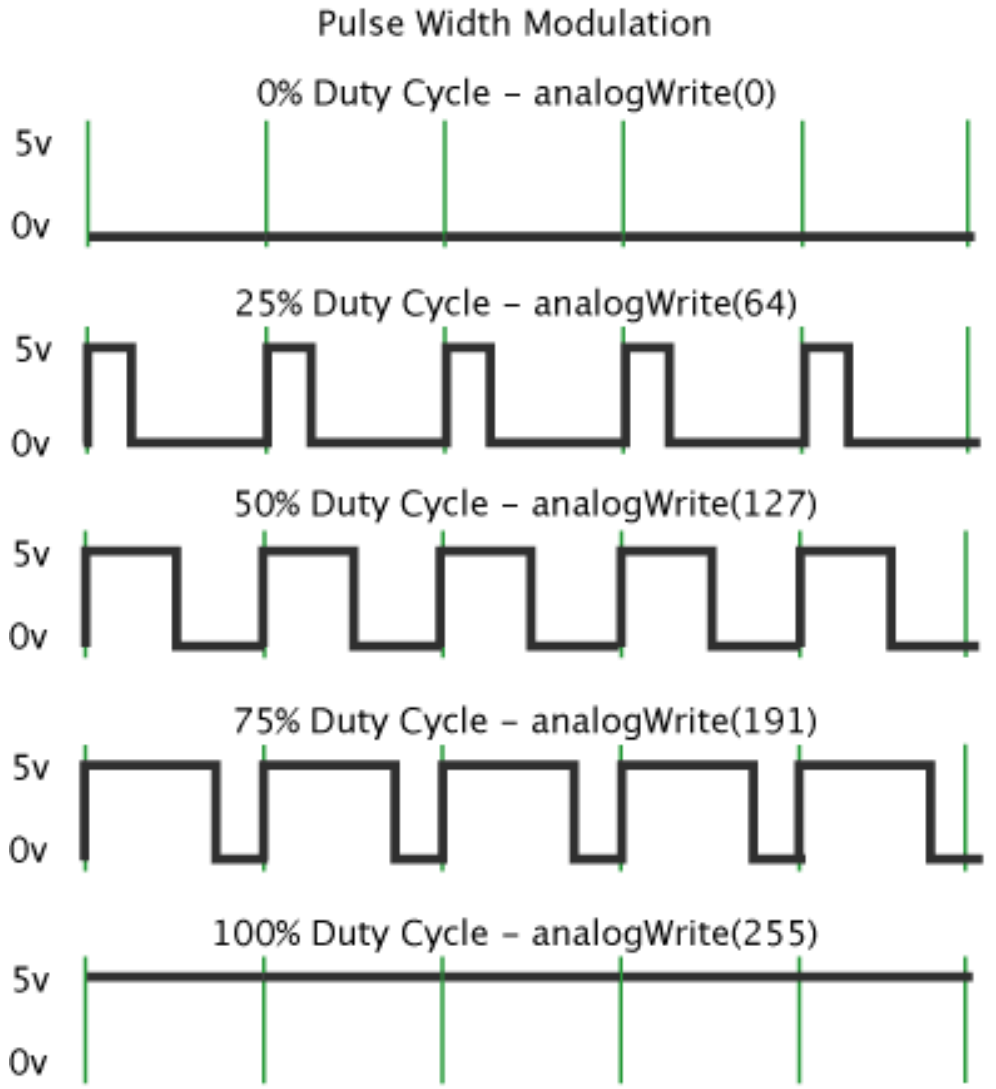
- PWM é a modulação da largura de ondas quadradas.
- PWM, do inglês Pulse Width Modulation, é uma técnica utilizada por sistemas digitais para variação do valor médio da voltagem, enquanto só se aplica na saída voltagens nula ou máxima. A técnica consiste em manter a frequência de uma onda quadrada fixa 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



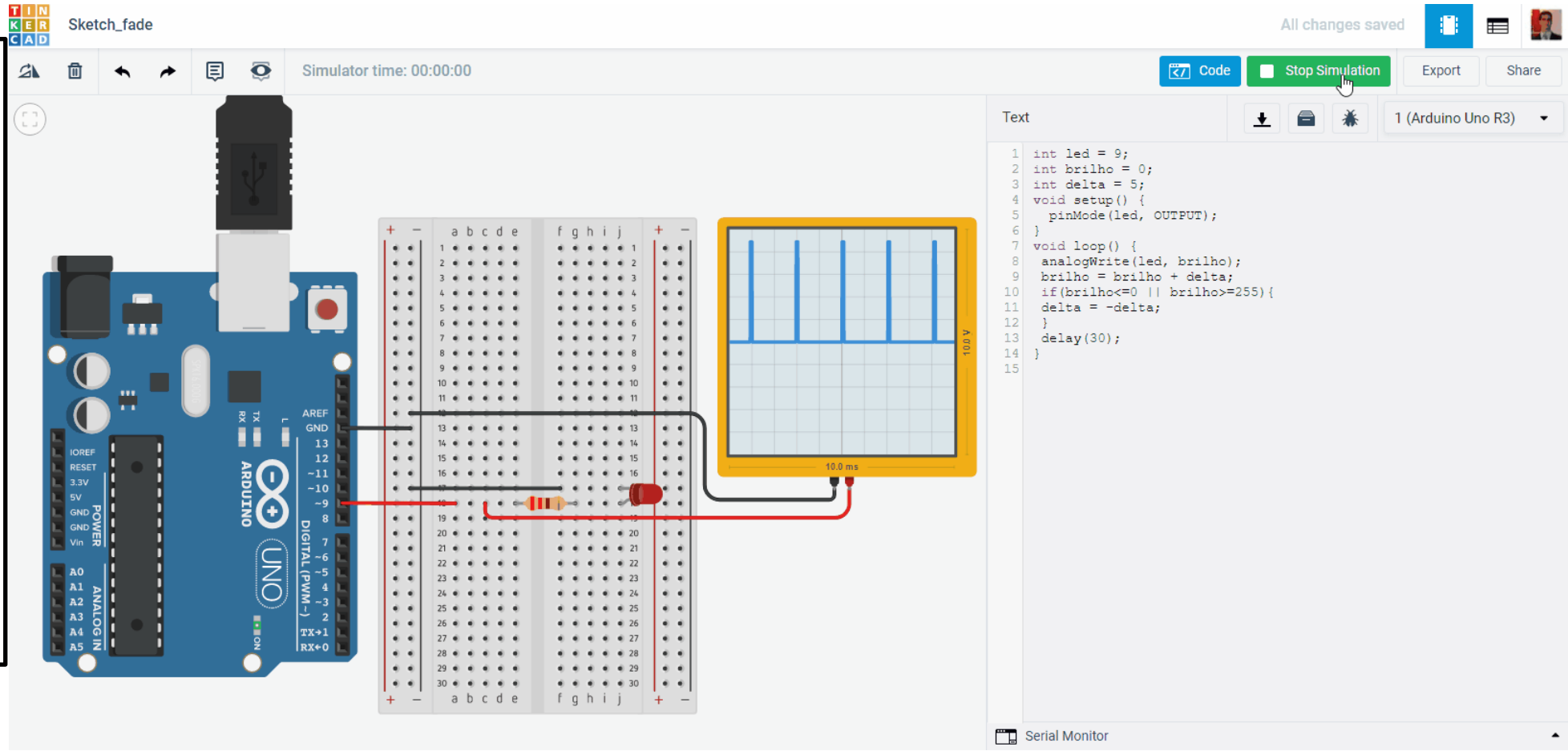
Acionamentos – PWM

- O gráfico ao lado ilusta 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).



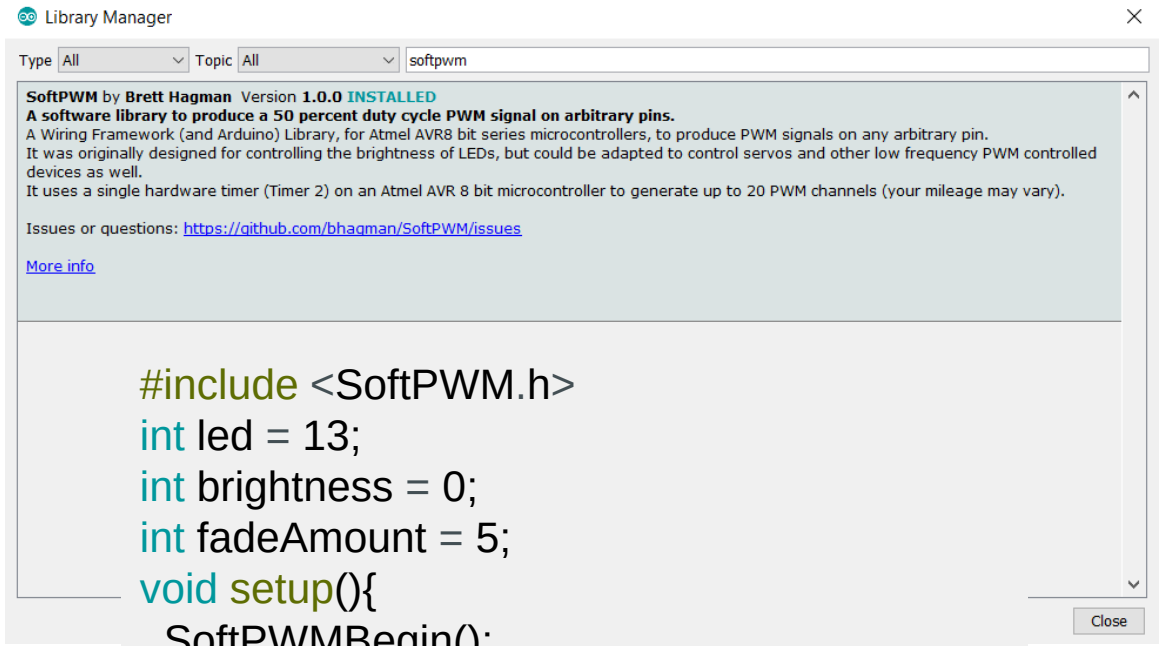
Acionamentos – PWM – **Sketch_05**

```
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);
}
```



Acionamentos – SoftPWM – **Sketch_06**

- 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 bliblioteca, 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.
- **Não há suporte no TinkerCAD para SoftPWM.**




```
#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);
}
```

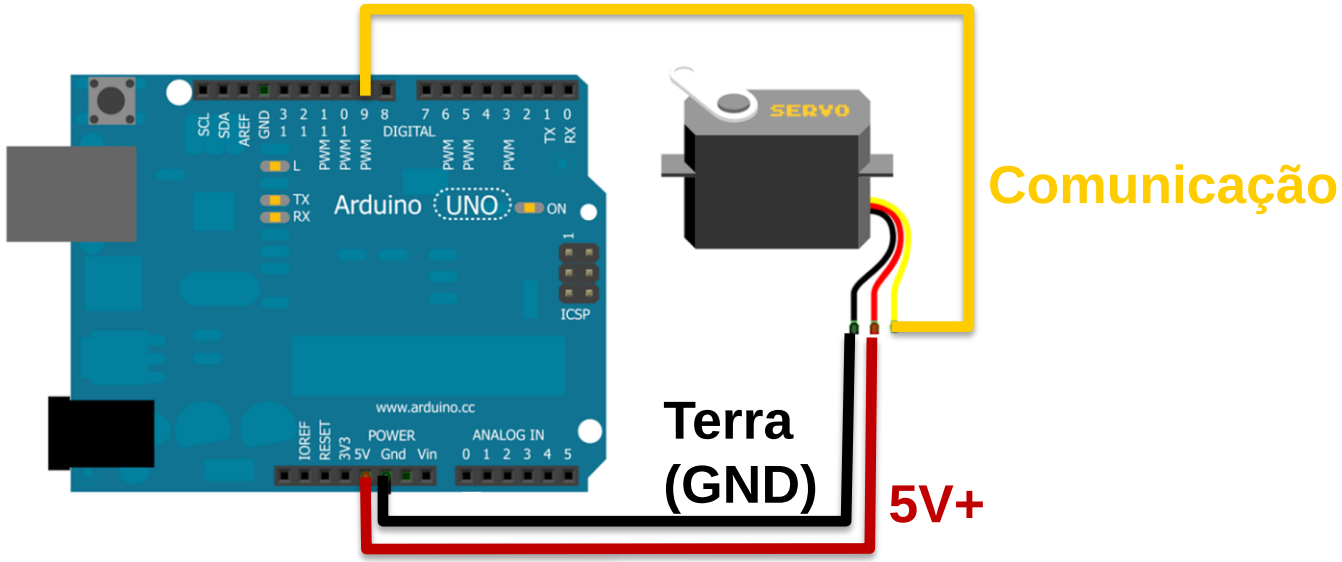
Acionamentos – PWM para servo motores

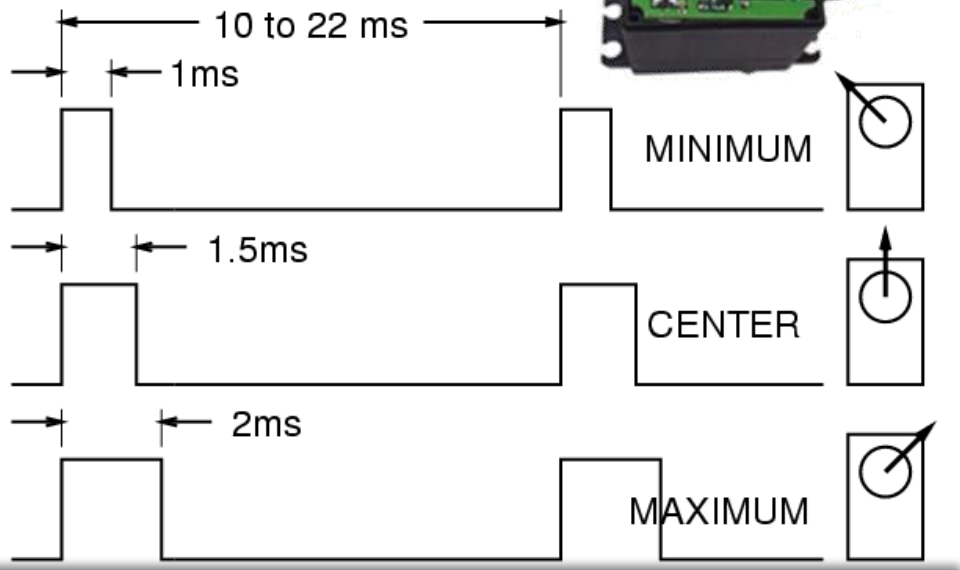
O Servo motor é na verdade um sistema: possui uma placa de controle, um potenciômetro (que serve de encoder), um motor CC e uma redução.

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.**

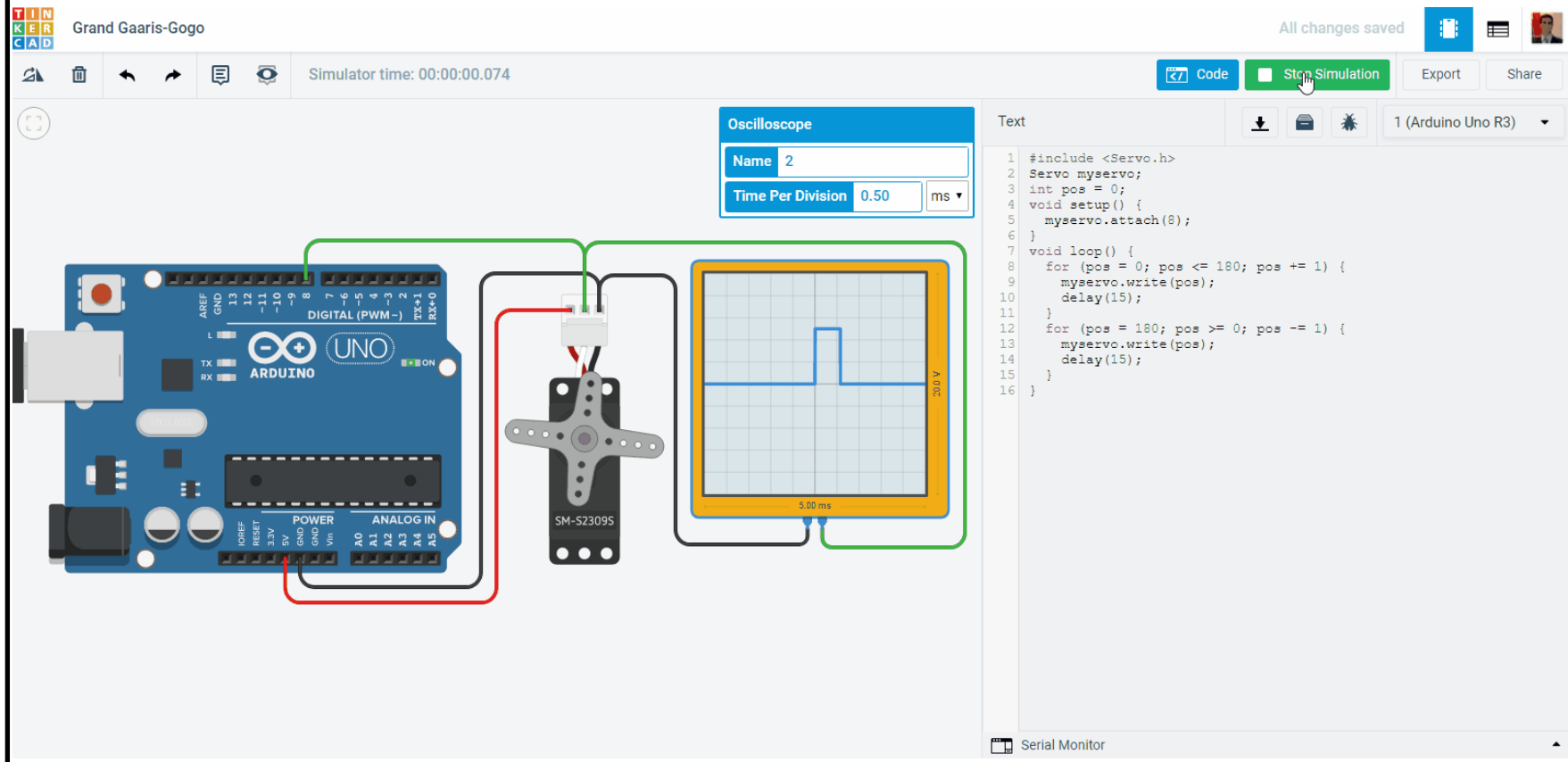
Caso contrário, vá para o próximo slide ver como simular no TinkerCAD.





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);
  }
}
```

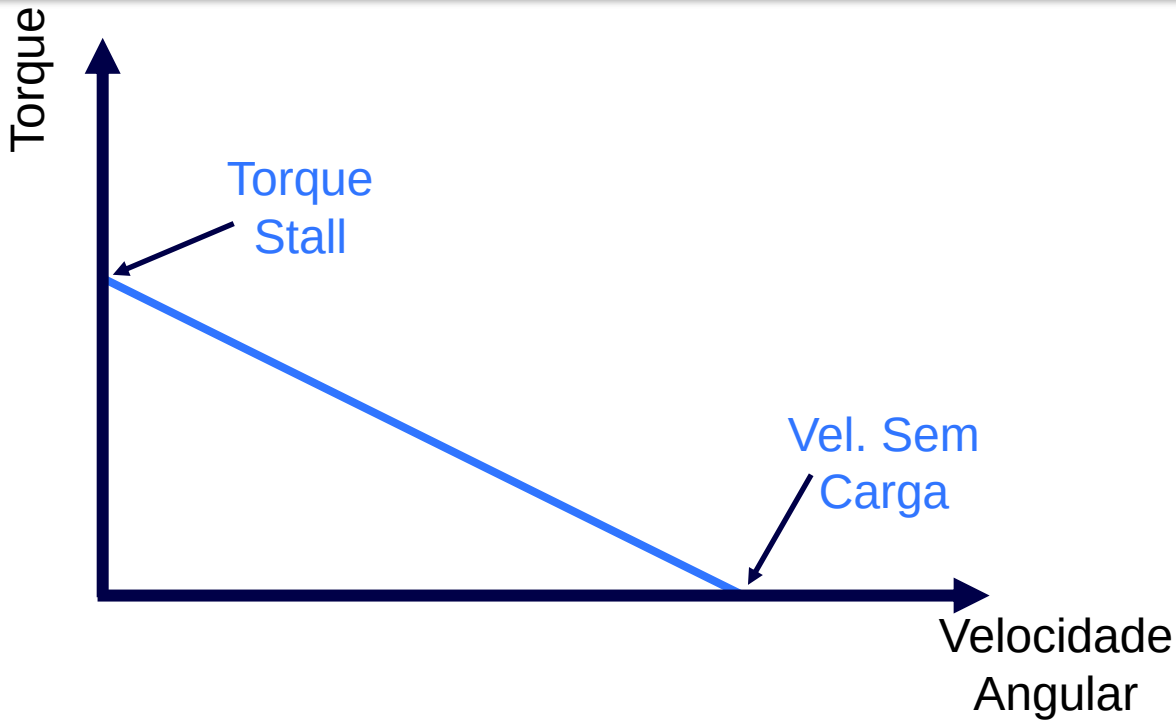


Motor DC

Considerando-se um motor DC, podemos dizer que:

- Quando um a carga é mantida constante, a velocidade de rotação é proporcional à voltagem aplicada.
- Quando a voltagem aplicada é constante, a velocidade angular é inversamente proporcional à carga aplicada.

Desta forma, podemos equacionar a carga (torque) de um motor DC por:



$$T = k \cdot I$$

$$V = I \cdot R + k \cdot \omega$$

$$T = \frac{V - \omega \cdot k}{R} \cdot k$$

Back EMF

Usar PWM para variar voltagem sem perdas elétricas

T = torque V= voltagem ω = velocidade angular R = resistência elétrica k = constante do motor I = corrente

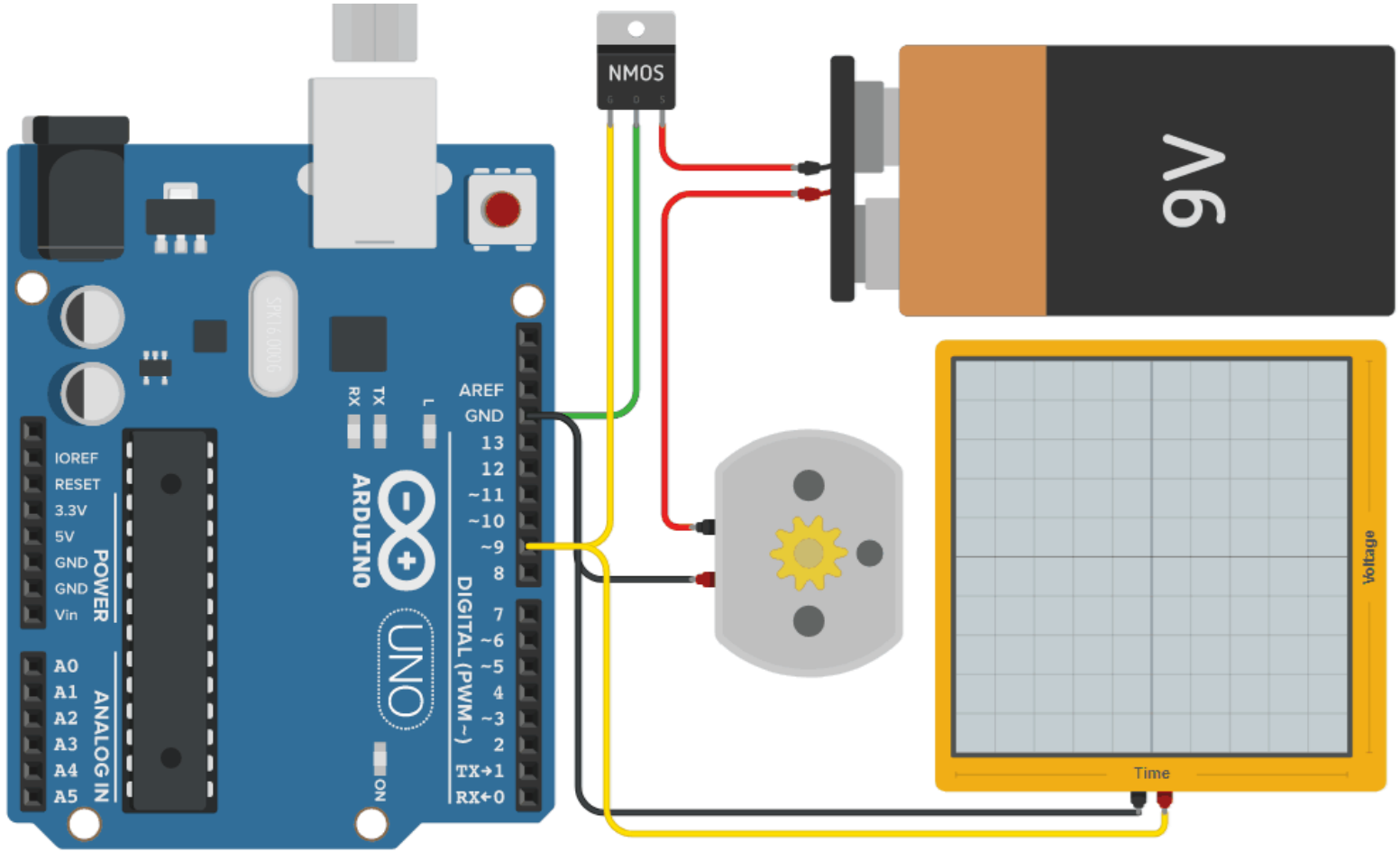
Controlando um Motor DC

Desta forma, se desejamos obter o máximo torque em um motor, devemos aplicar a maior corrente possível.

Entretanto, o Arduino possui limitações na quantidade de corrente que um pino digital pode fornecer, conforme a tabela abaixo (retirada do datasheet do ATmega328P), de 40mA ou 0,04A.

Para contornar essa limitação, podemos usar um transistor MOSFET. Quando existe uma voltagem aplicada em G (gate), a corrente passa de D (drain) para S (source).

DC Current per I/O Pin	40.0 mA
DC Current V_{CC} and GND Pins	200.0 mA

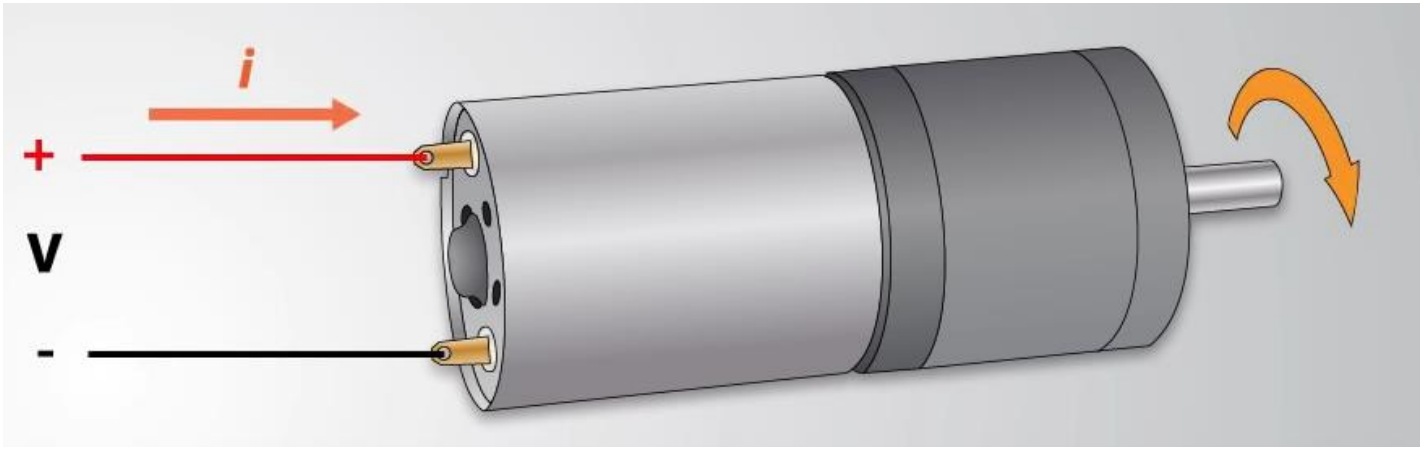


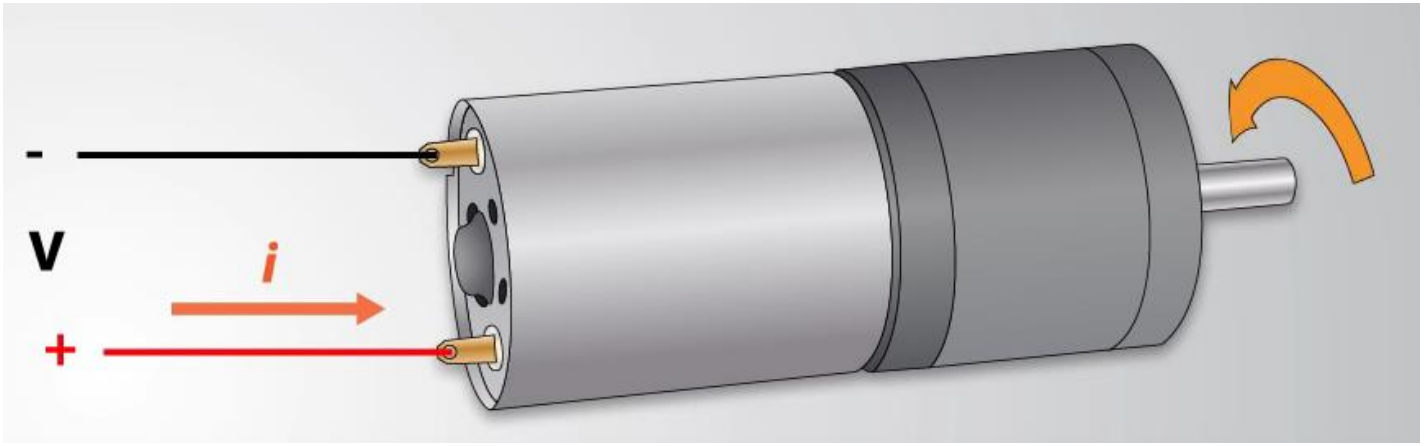
Controlando um Motor DC

Com a ajuda do transistor, conseguimos resolver o problema da limitação de corrente e proporcionar mais torque ao nosso motor.

Entretanto, precisamos ser capazes de escolher a direção de rotação do motor, de modo a controlarmos nosso dispositivo.

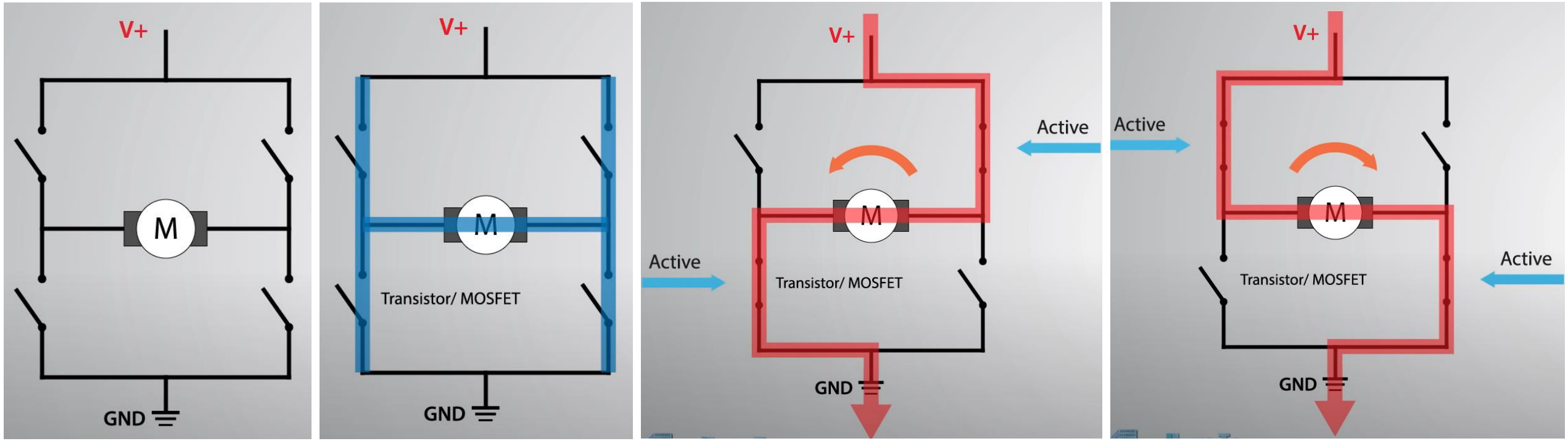
Sabemos que, se invertemos a polaridade da voltagem aplicada aos terminais do motor, ele irá mudar sua direção de rotação.





Controlando um Motor DC

A resposta para nosso problema é o circuito elétrico conhecido como ponte-H. Ela é constituída por 4 transistores atuando como chaves. Dependendo da combinação de chaves fechadas, o motor gira para um lado ou para o outro.



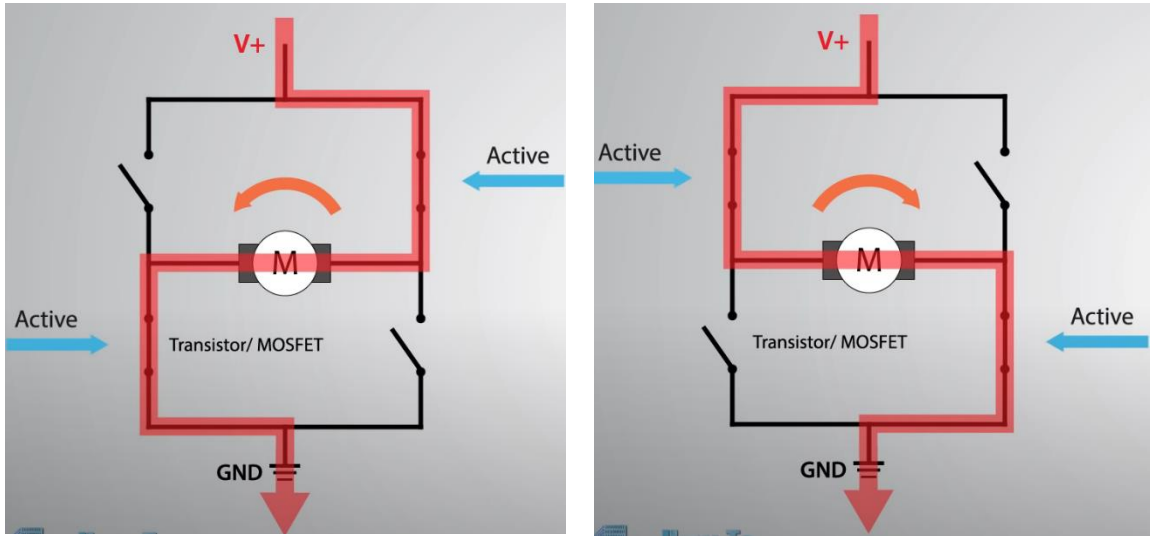
Controlando um Motor DC

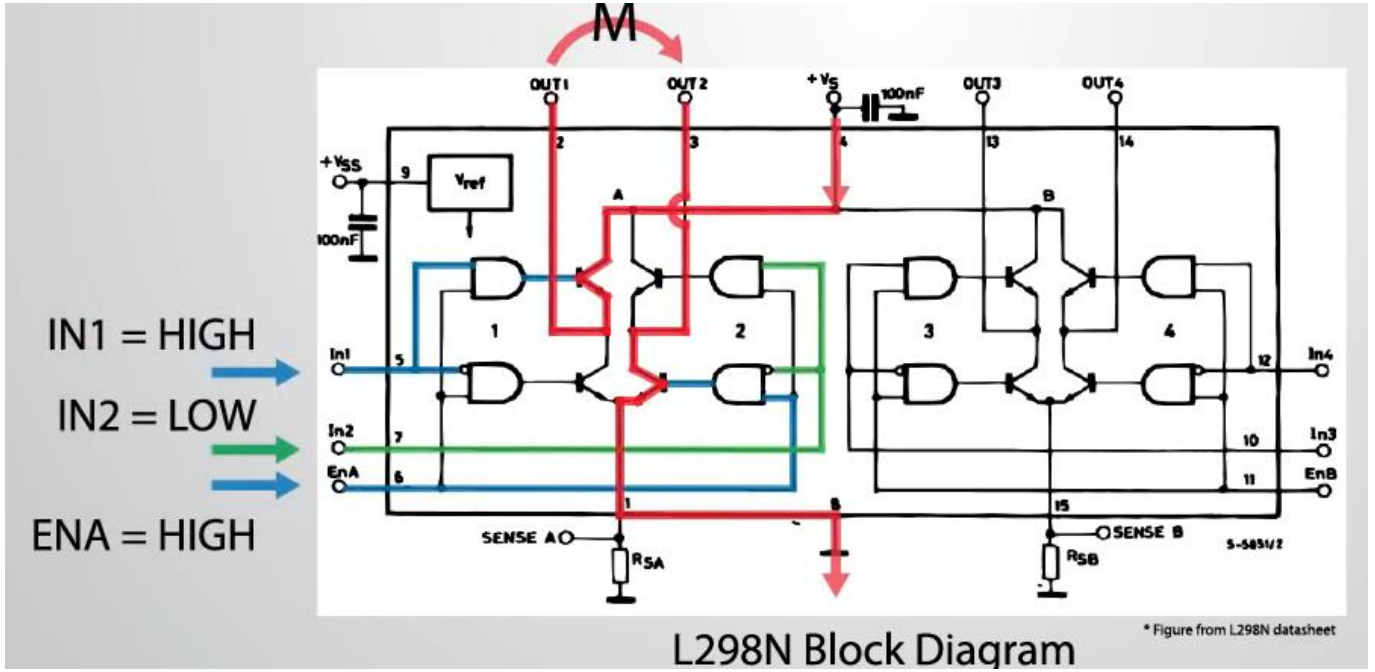
Abaixo está o diagrama de blocos do L298N, que possui duas pontes-H.

A velocidade do motor é controlada por um sinal PWM no pino ENABLE.

Se colocarmos HIGH no pino IN1 e LOW no pino IN2, o motor gira para um lado.

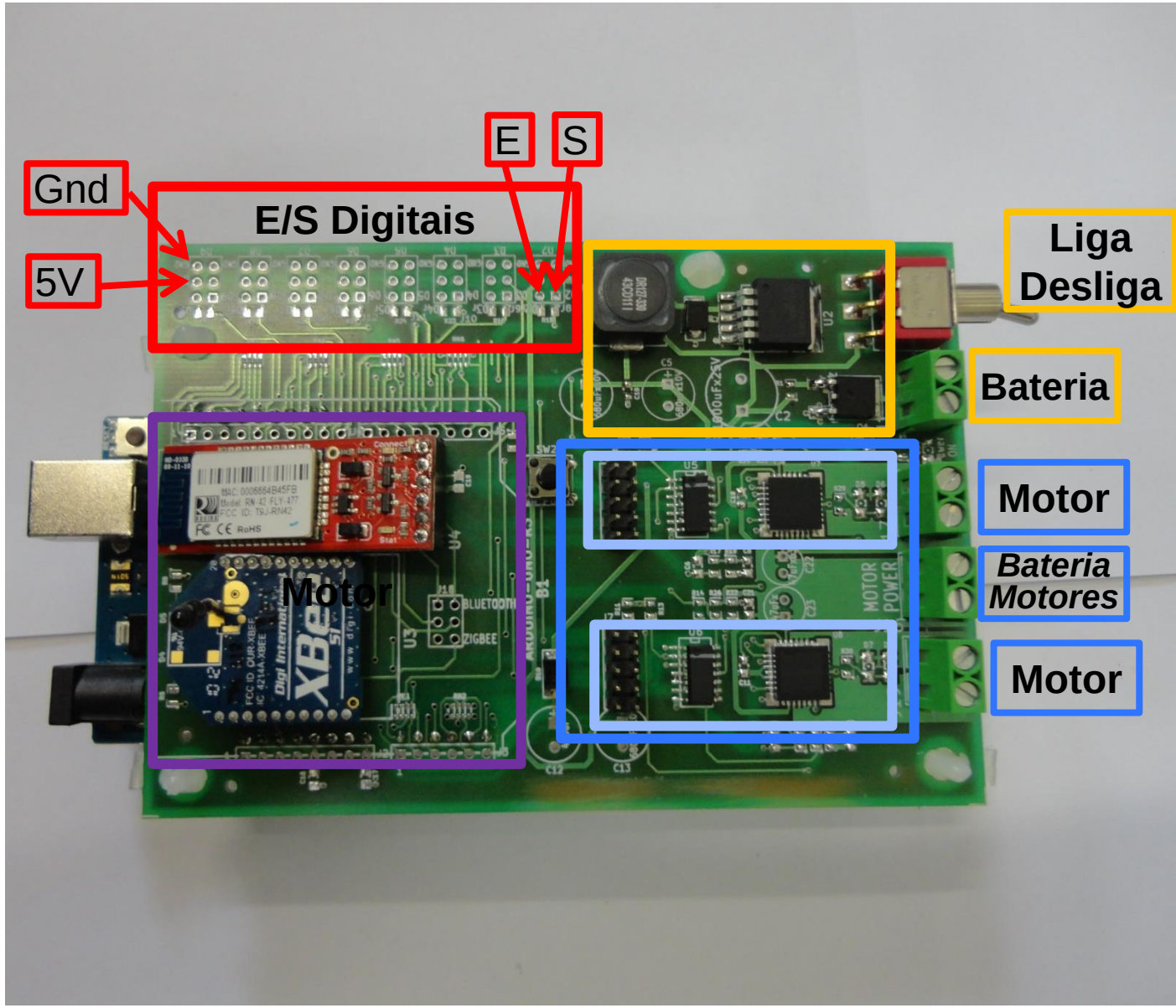
Se colocarmos LOW no pino IN1 e HIGH no pino IN2, o motor gira para o outro lado.





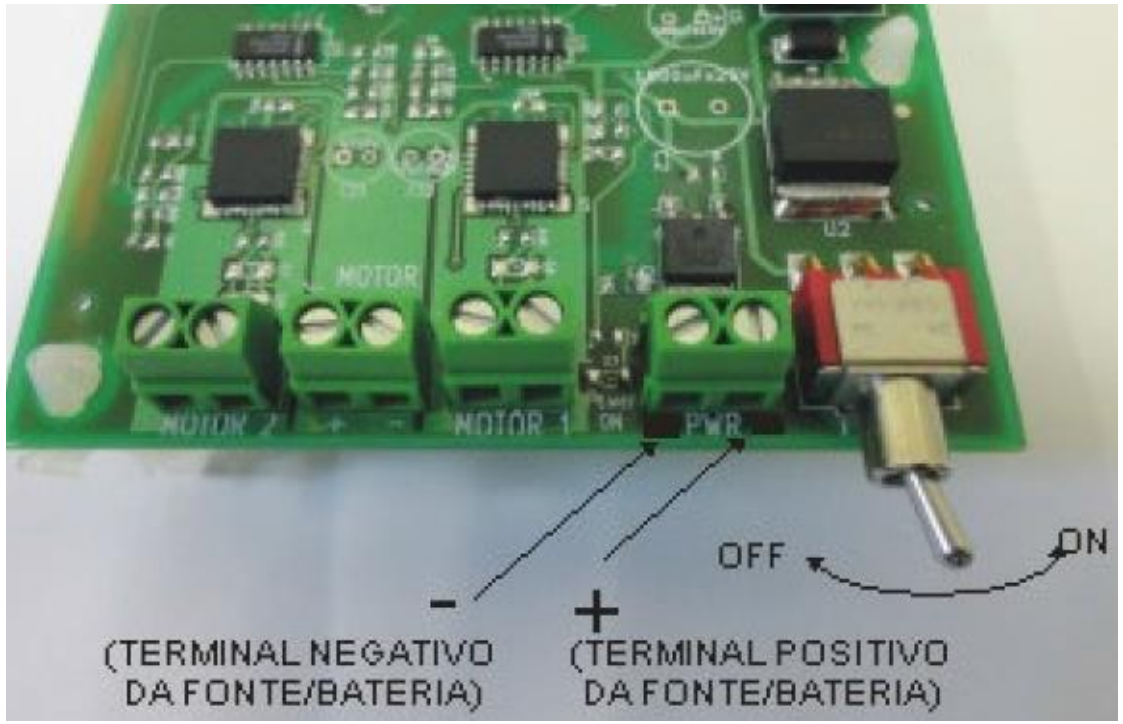
Acionamentos com PACA – PWM, drivers e motores DC

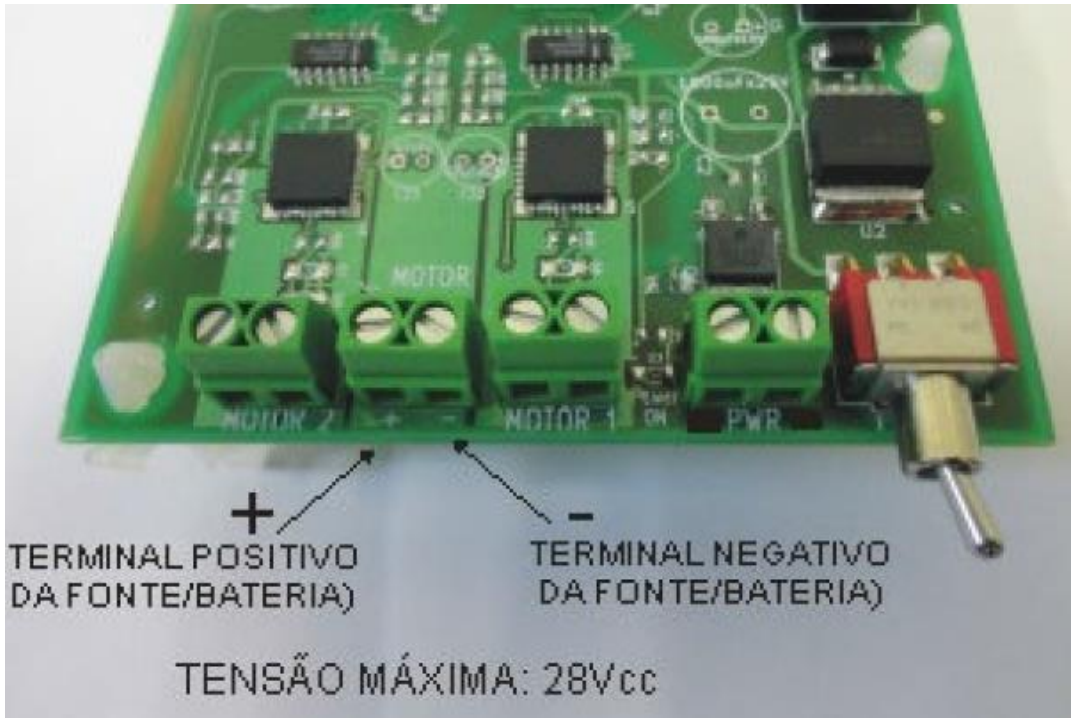
- A PACA contém
  - **Módulo de comunicações**, com um módulo bluetooth;
  - **Módulo de acionamento** com dois **drivers** de motores DC;
  - **Módulo de regulagem de voltagem da bateria**;
  - **Módulo de entradas e saídas digitais**, com 5V vindos da bateria (S) ou do Arduino (E);
- Um comentário é que o módulo de controle de motores deve ser energizado externamente, através da conexão denominada ***Bateria Motores*** na figura ao lado!



Acionamentos com PACA – PWM, drivers e motores DC

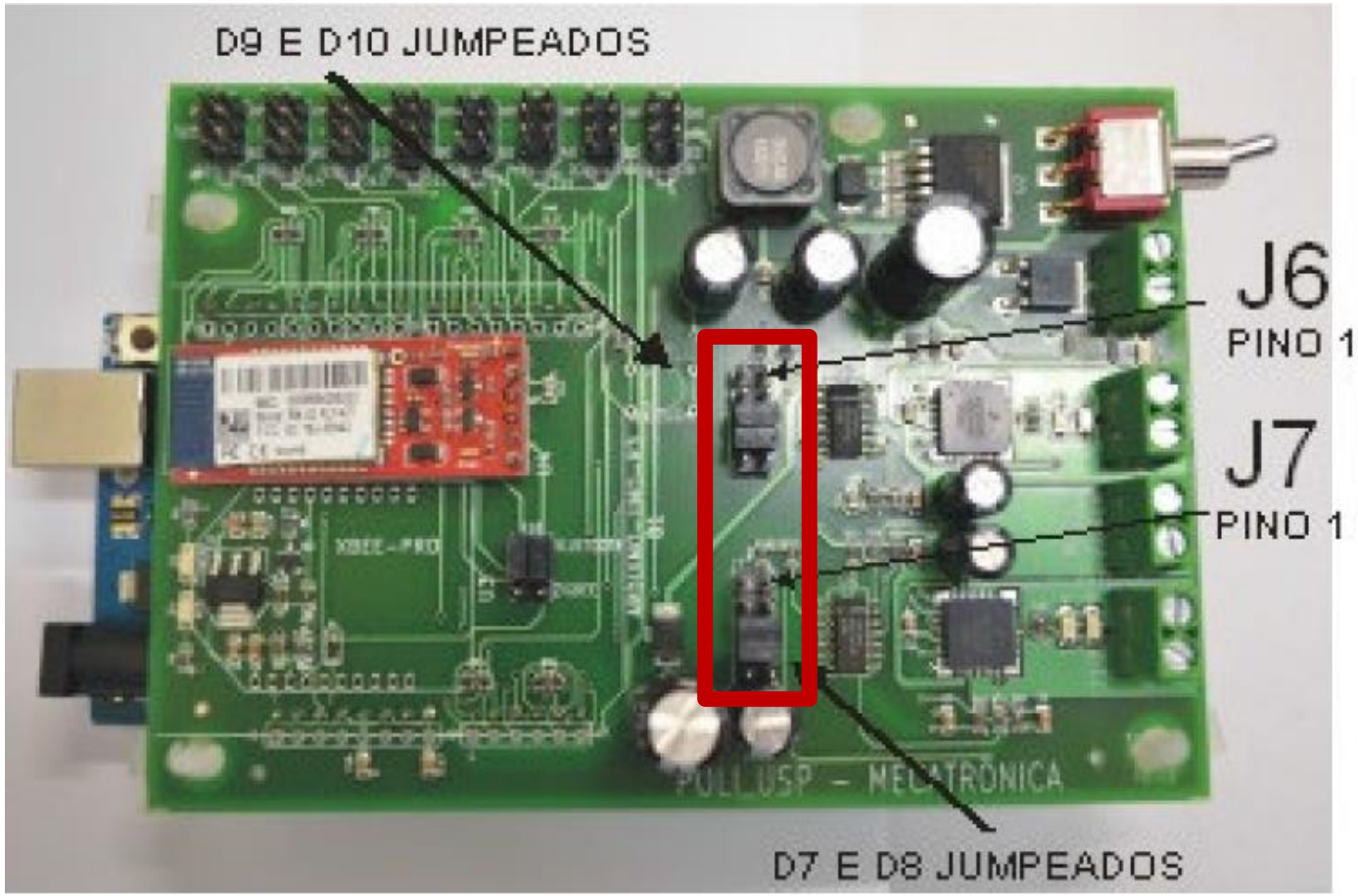
- Cuidado com as ligações entre bateria, PACA e módulo de driver de motor. Alguns textos de + e – estão trocados!
- SIGA ESTAS FIGURAS!





Acionamentos com PACA – PWM, drivers e motores DC

- 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!



Acionamentos com PACA – PWM, drivers e motores DC – **Sketch_08**

```
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:

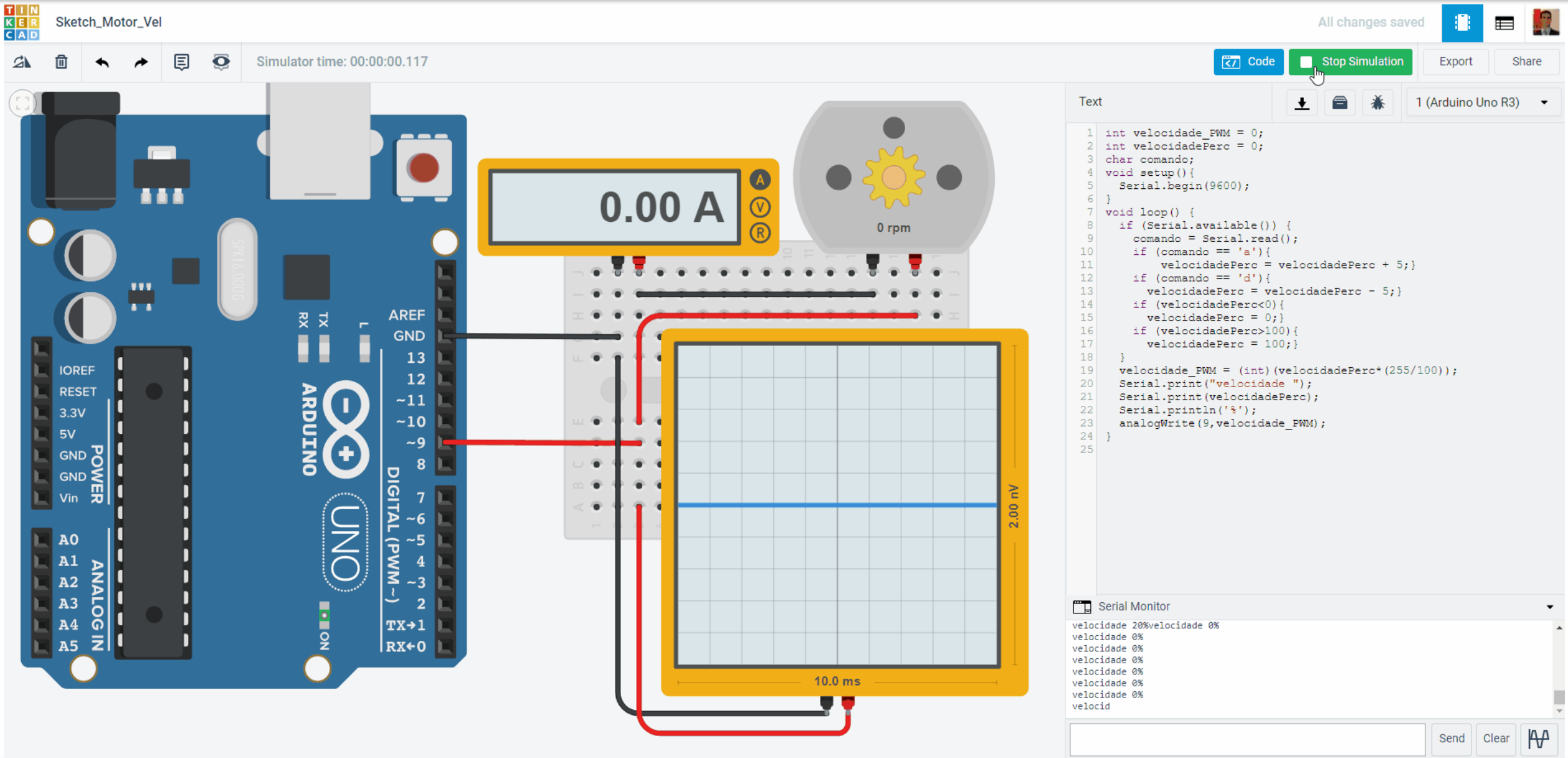
  Faça as ligações do **motor 1** conforme explicado no slide anterior;

  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**



```
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.

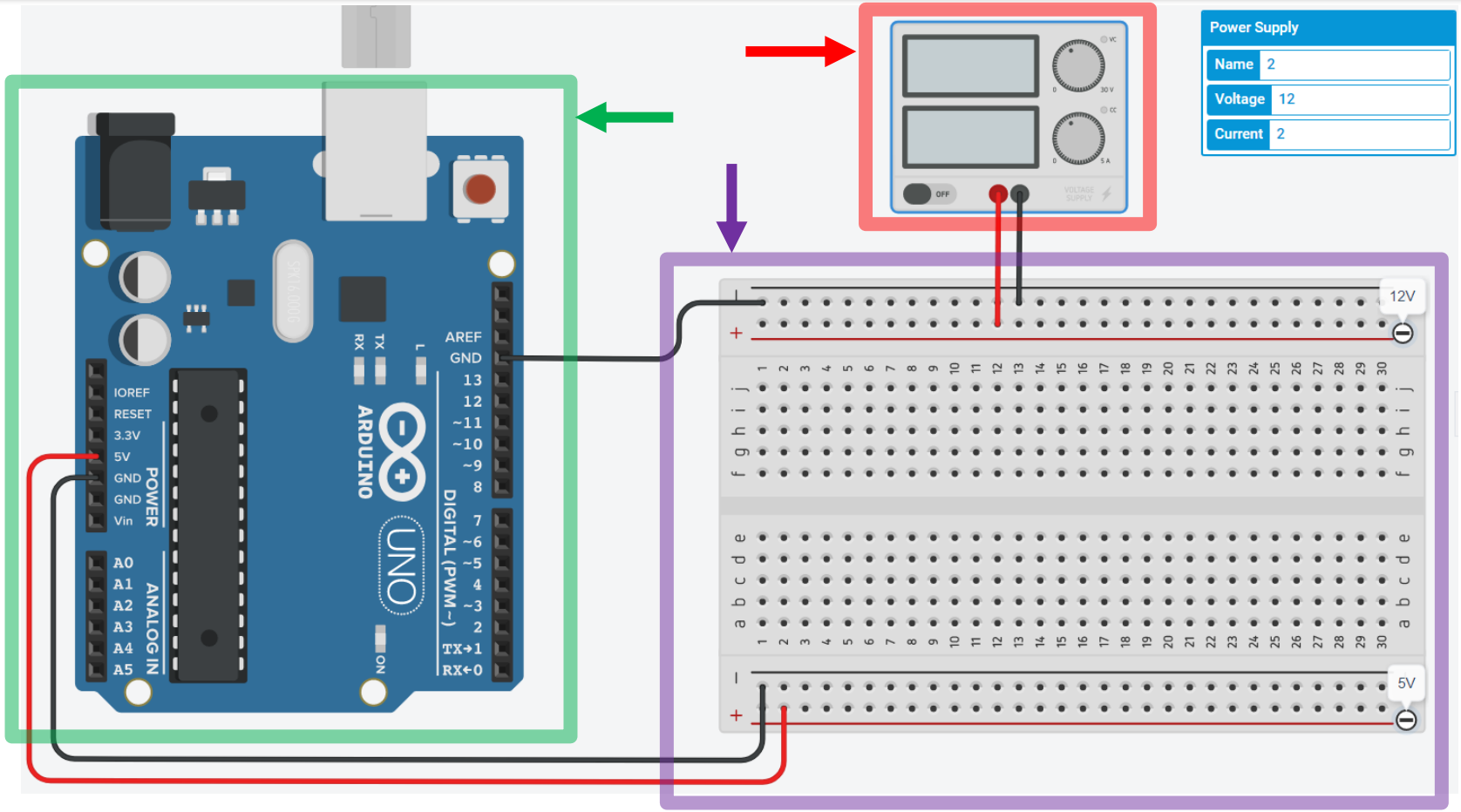
  Ao digitar “a” no serial monitor, o programa deve acelerar o motor em 5% e ao digitar “d” deve desacelerá-lo em 5%.

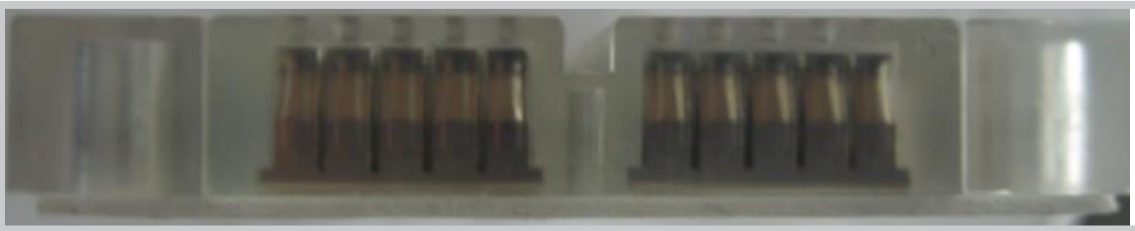
  Ao digitar “b” no serial monitor, o motor deve girar em um sentido e ao digitar “c”, o motor deve girar no sentido oposto.

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

TinkerCAD – 5V, 12V e terra

Coloque um **Arduino Uno**, um **Power Suply (Fonte)** e uma **breadboard (Protoboard)** . Ligue os fios conforme indicado para criar um trilha de 12V e terra na parte de cima da protoboard e um trilha com 5V e terra na parte de baixo da protobord. Note que o terra deve ser comum, ou seja, deve-se ligar o terra da **Fonte** com o terra do **Arduino Uno**.





Lembrando que na protoboard, os pinos de uma mesma linha estão em curto, ou seja, conectados.

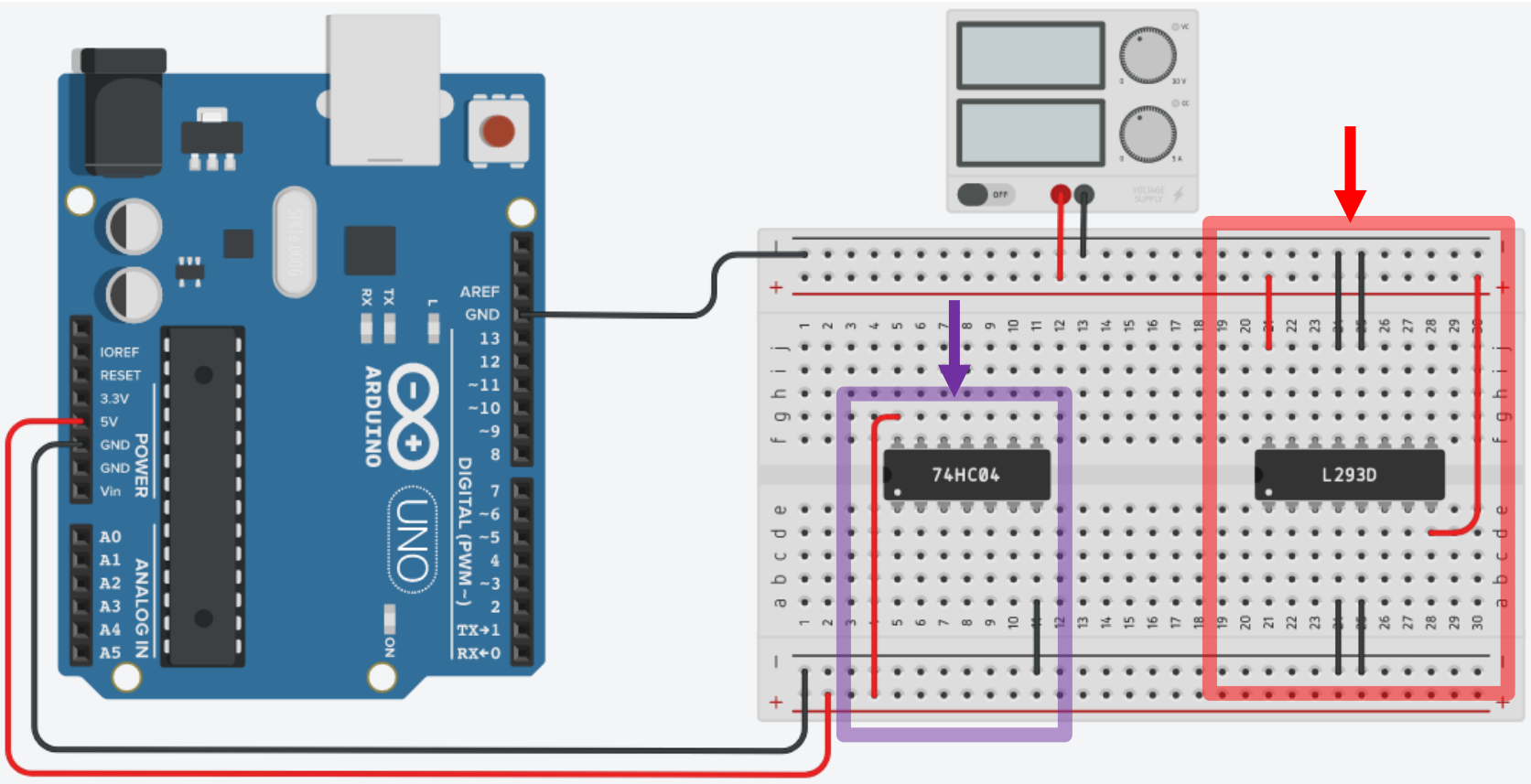
TinkerCAD – M1

O chip **L293D**, disponível do TinkerCAD, é uma ponte H e iremos usa-lo para controlar nossos motores M1 e M2.

Vamos definir o pino 3 para o controle da velocidade (usando PWM) e o pino 2 para controlar a direção de rotação.

Do que definimos anteriormente:

- Se colocarmos HIGH no pino IN1 e LOW no pino IN2, o motor gira para um lado.
- Se colocarmos LOW no pino IN1 e HIGH no pino IN2, o motor gira para o outro lado.



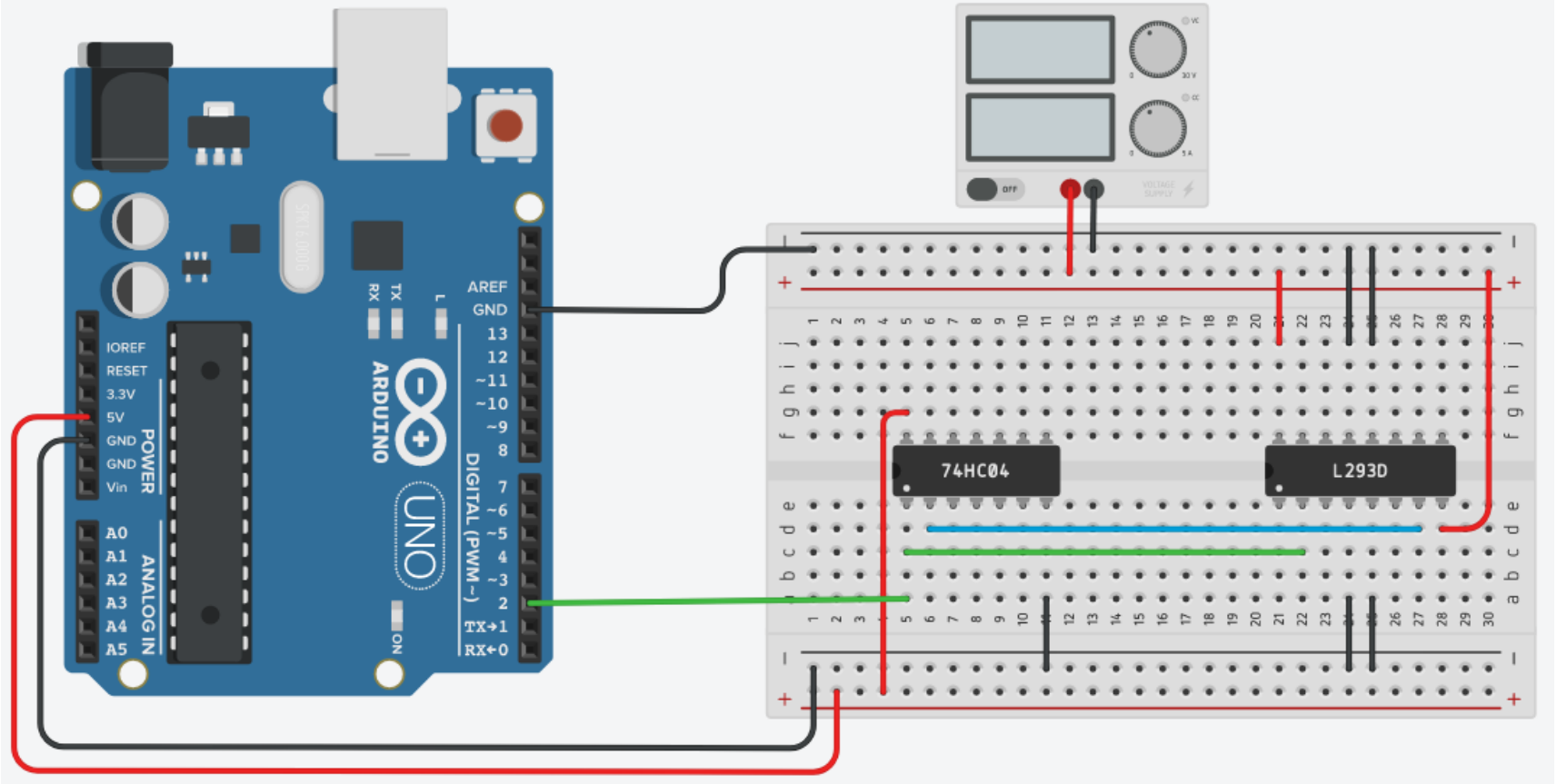
Desta forma, para controlarmos a direção com apenas um pino, precisamos de um chip que faça a inversão do valor lógico. Se você clicar em *components* e depois em *all*, irá encontrar o chip **74HC04**, que é um inversor com 6 entradas e saídas. La também se encontra o chip **L293D**.

Coloque estes dois componentes e ligue 5V, 12V e terra neles conforme a figura.

Vamos ligar o pino digital 2 do Arduino Uno tanto no pino *Input 1* do inversor 74HC04 quando no pino *Input 1* do L293D.

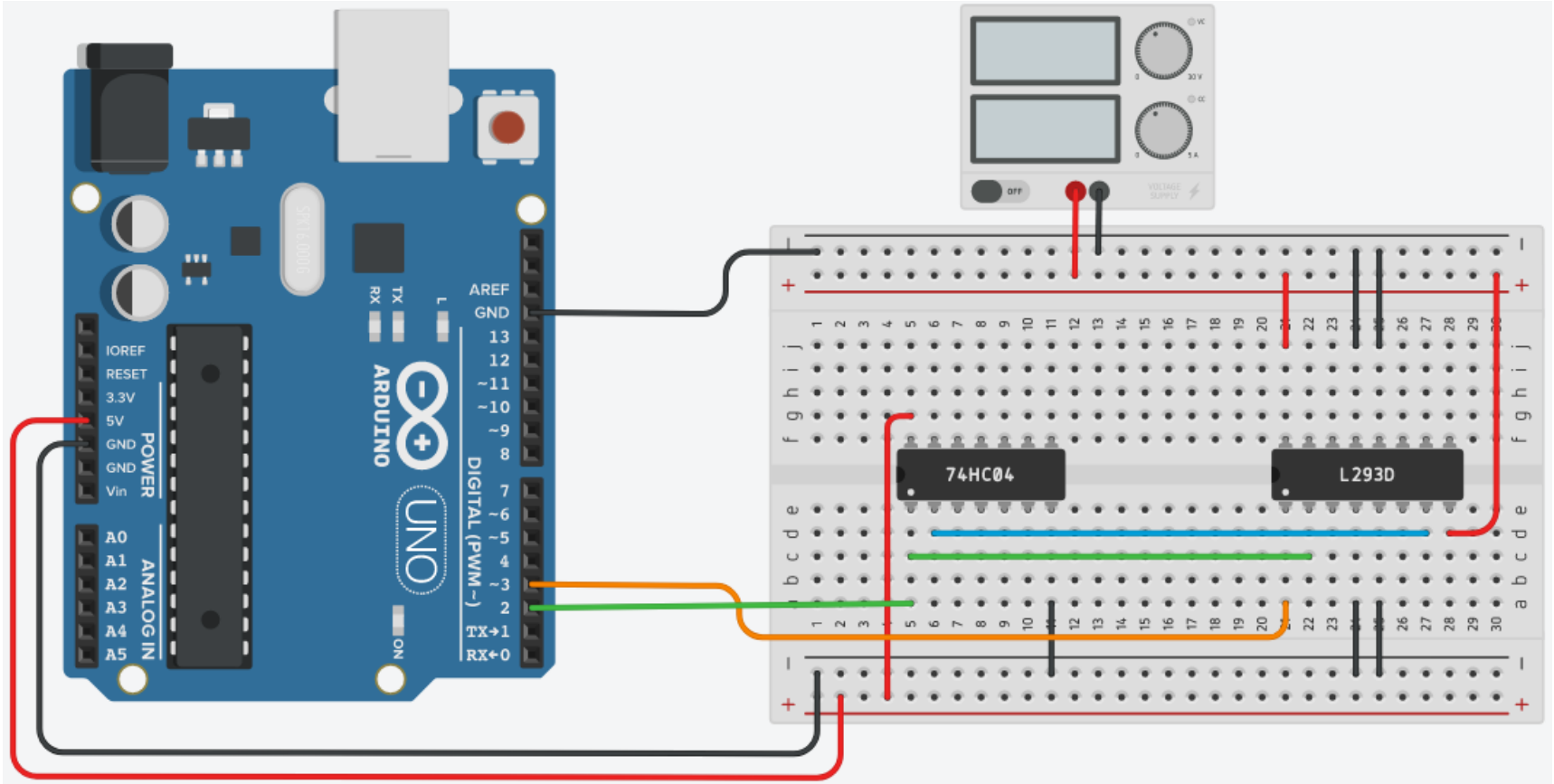
Depois ligue o pino *Output 1* do inversor 74HC04 no pino *Input 2* do L293D.

Estas ligações estão exemplificadas na figura ao lado.



Ligue o pino digital 3 do Arduino Uno no pino *Enable 1 & 2* do L293D.

Esta ligação está exemplificada na figura ao lado.



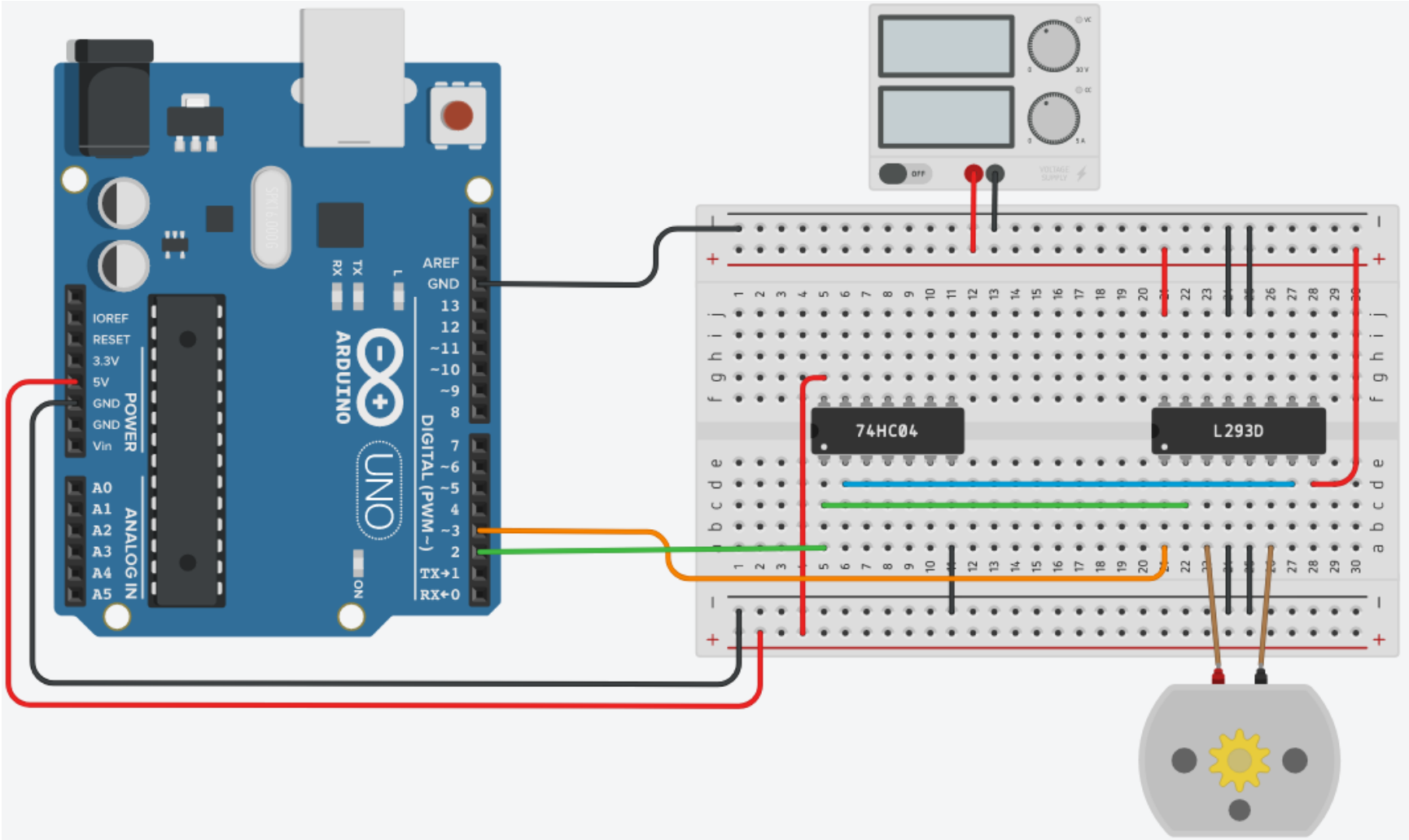
TinkerCAD –PWM, direção, drivers e motores DC – **Sketch_09**

Adicione o motor DC M1 e **ligue seus terminais aos pinos *Output 1* e *Output 2* do L293D**.

E Pronto!!!

Você já pode rodar o Sketch_09.

Lembre-se de **alterar** seu código para controlar a velocidade do motor M1 com PWM (analogWrite) no pino 3 e controlar o sentido de rotação do motor com o pino 2 (digitalWrite).



```
#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.

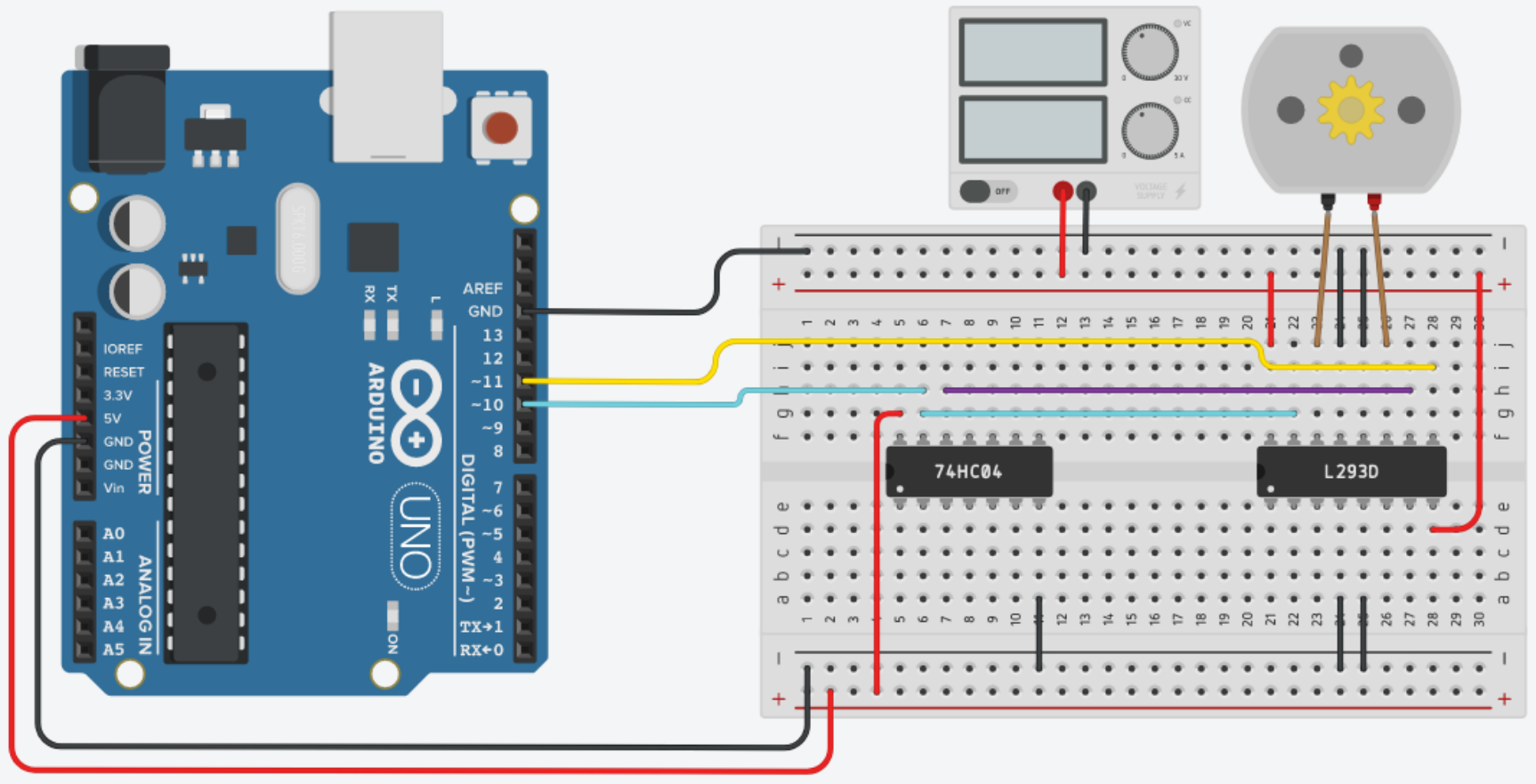
  Ao digitar “a” no serial monitor, o programa deve acelerar o motor em 5% e ao digitar “d” deve desacelerá-lo em 5%.

  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 implementação no TinkerCAD. Crie o circuito e use o código deste slide.

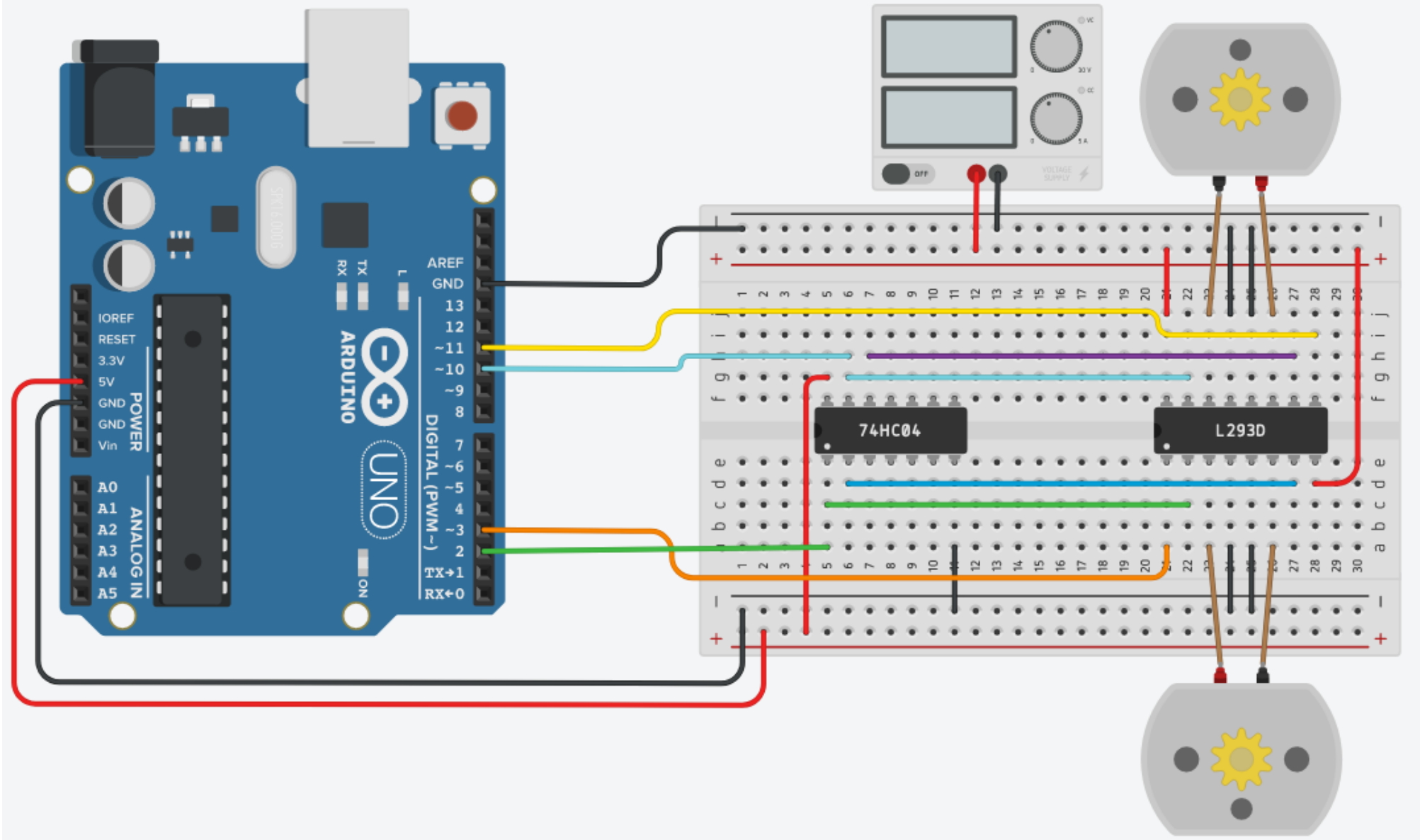
- **Lembre-se de que o TinkerCAD não suporta a biblioteca softpwm. Por isso o motor M1 é controlado com os pinos 2 e 3.**

Repita todos os procedimentos para o Motor M2, com o **controle de velocidade através de PWM (analogWrite) no pino 11** e **com o controle do sentido de rotação do motor no pino 10 (digitalWrite)**.



TinkerCAD – M1 e M2

Se você seguiu todos os passos corretamente, estará com um diagrama igual a este ao lado, controlando velocidade e sentido de rotação dos motores M1 e M2.

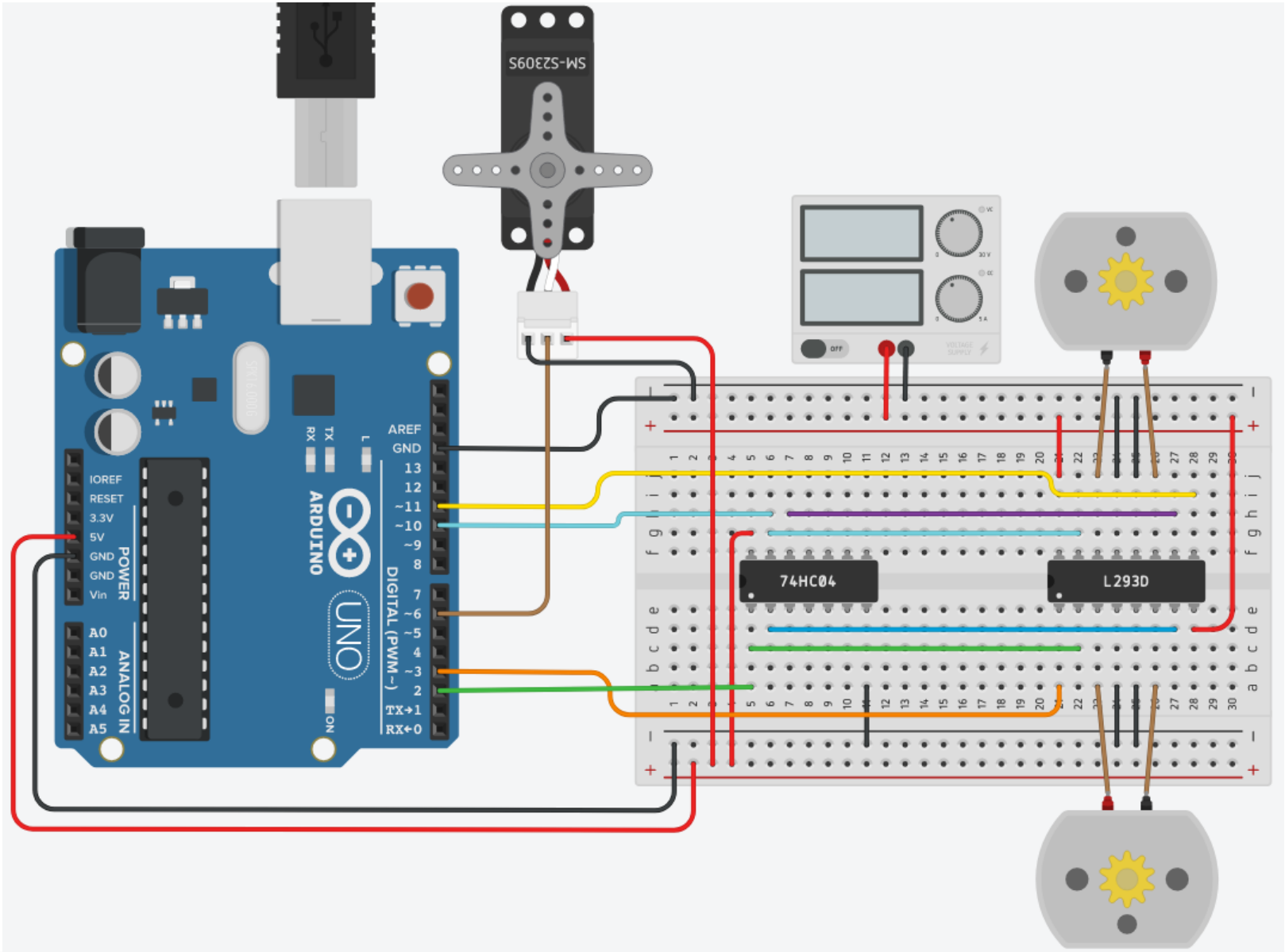


TinkerCAD – Exemplo adicionando um servo

Para simular o Servomotor que controlará o mecanismo do seu dispositivo, você pode seguir as ligações do diagrama ao lado.

Utilize também a biblioteca Servo no TinkerCAD, conforme ensinado anteriormente.

Neste caso, a posição ou velocidade de rotação do servo está sendo controlada através do pino 6 do Arduino Uno.

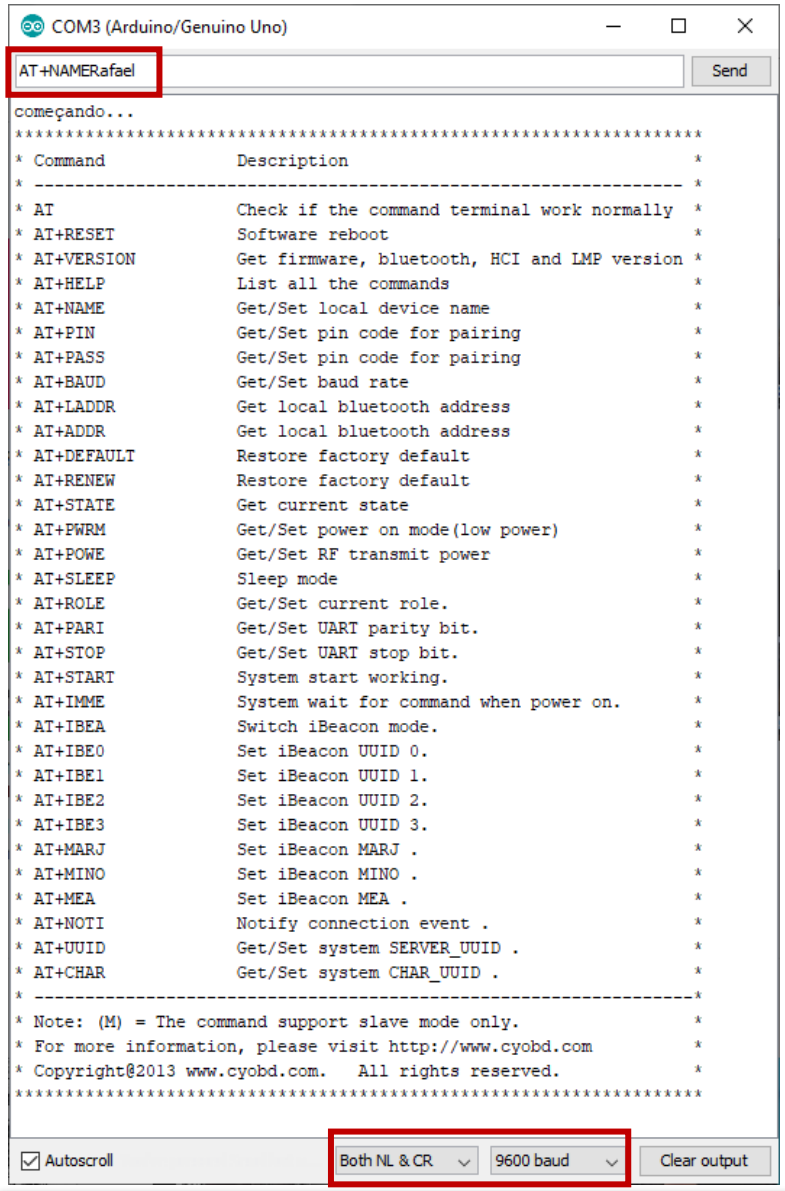


Comunicação com PACA – Bluetooth – **Sketch_11**

```
#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);
  }
}
```

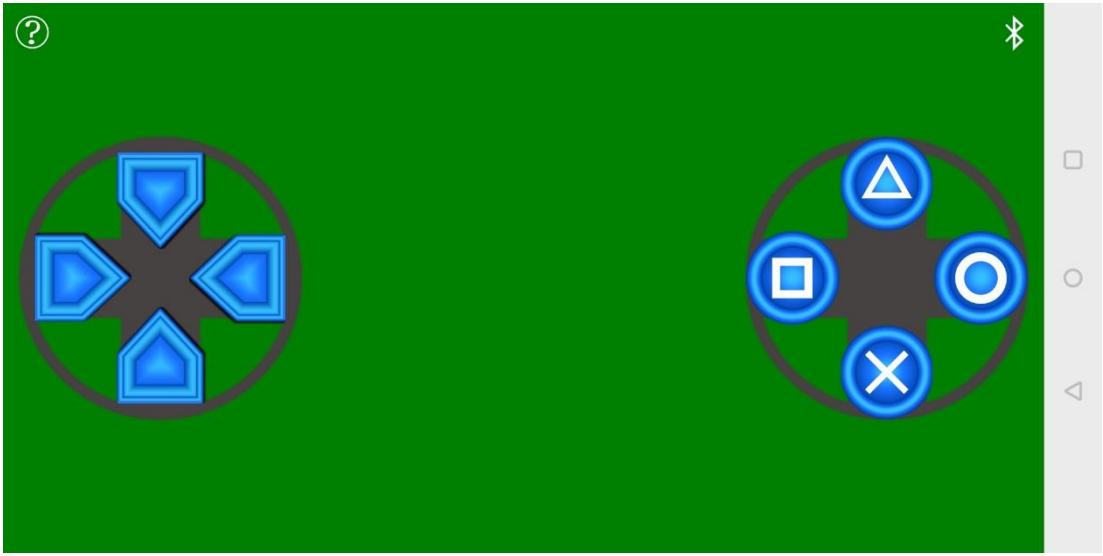
- Como o arduino uno não consegue. Ao estar sendo alimentado pelo cabo USB, dar energia suficiente para o módulo bluetooth, tenha certeza que a bateria está alimentando a PACA;
- A primeira coisa a se fazer para a comunicação bluetooth é dar upload do código ao lado para o arduino uno.

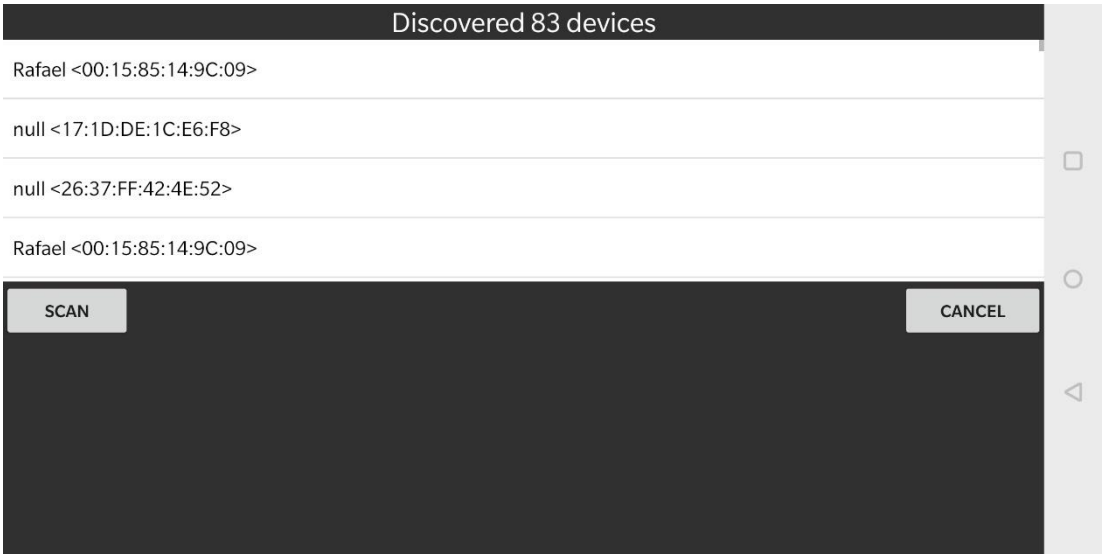
Comunicação com PACA – Bluetooth



- 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 Bluethooth;
- Então digite *AT+NAMEseunome* (sim, é tudo junto) e de enter.
- Voce receberá uma confirmação do novo nome do bluetooth.

Comunicação com PACA – 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.