



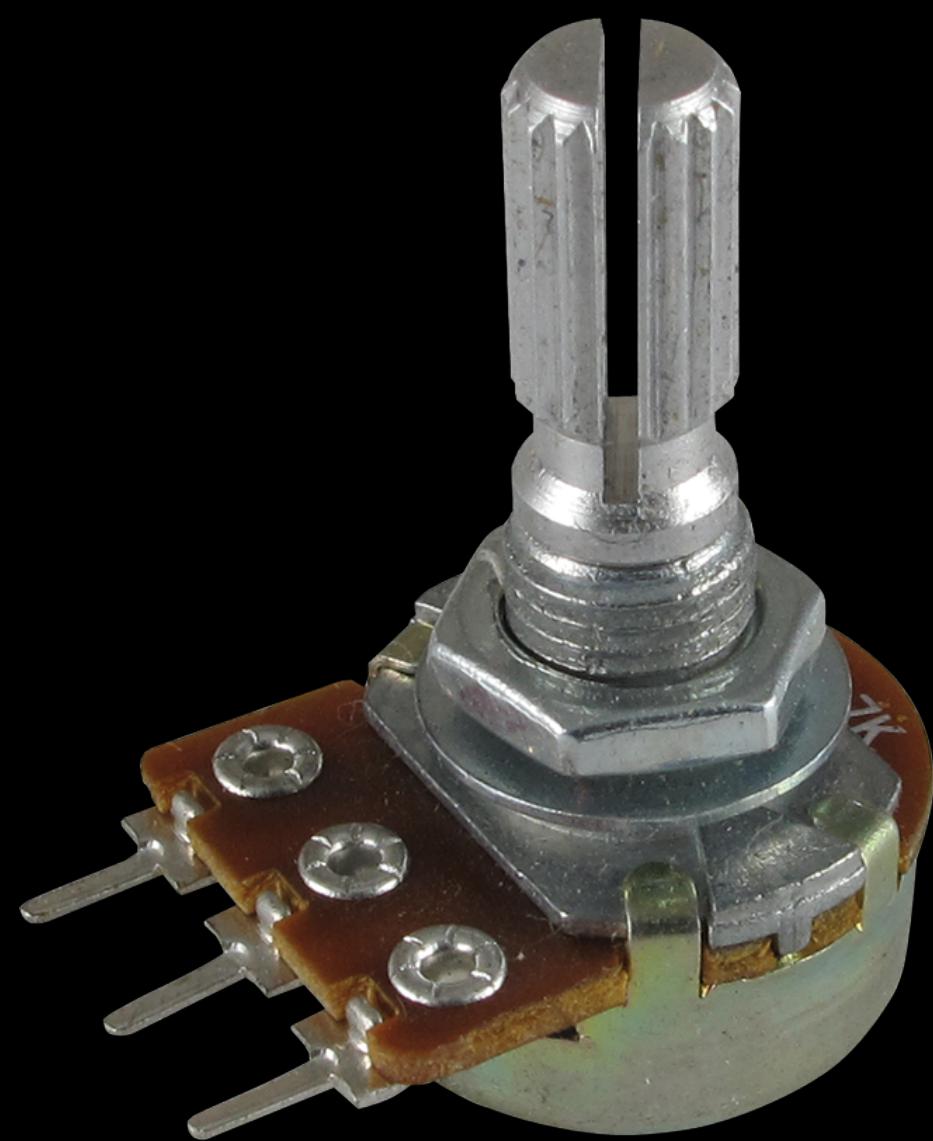
Projeto 08

Controle Analógico – Teoria

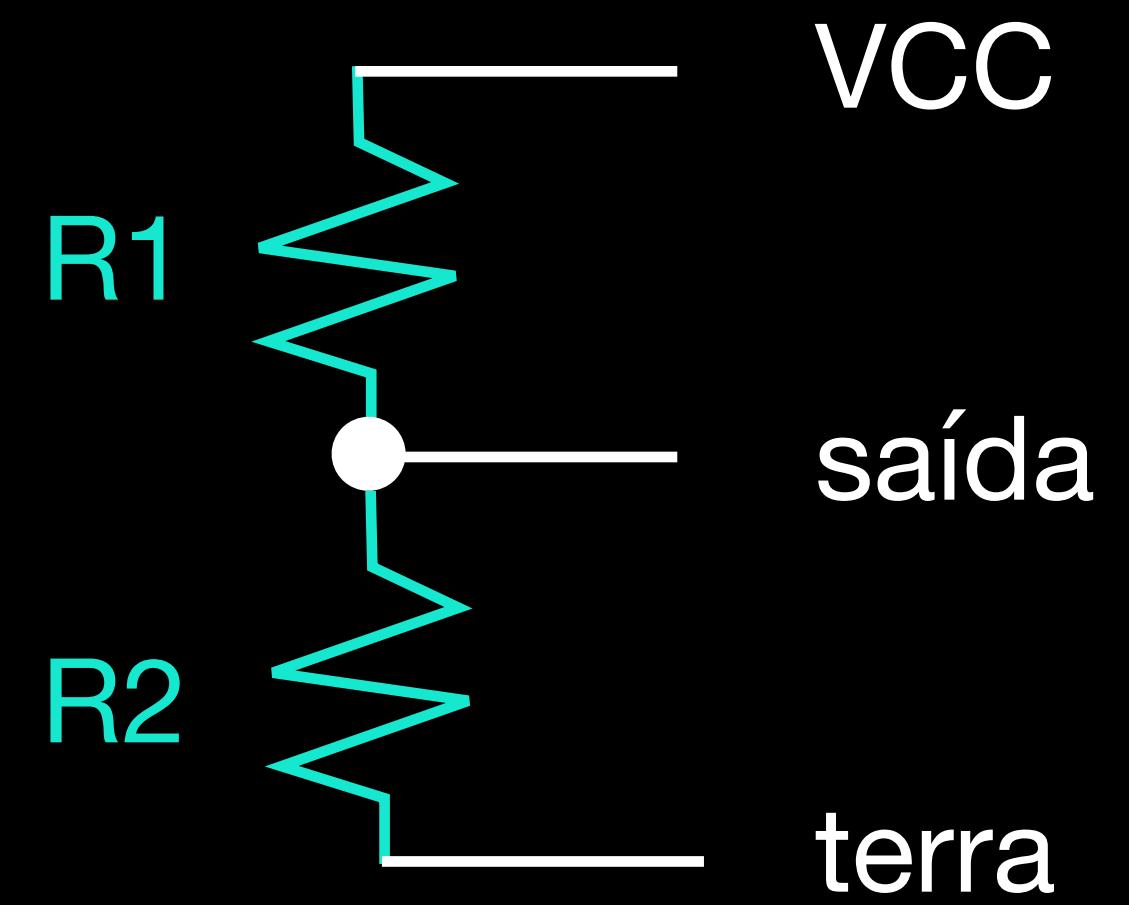
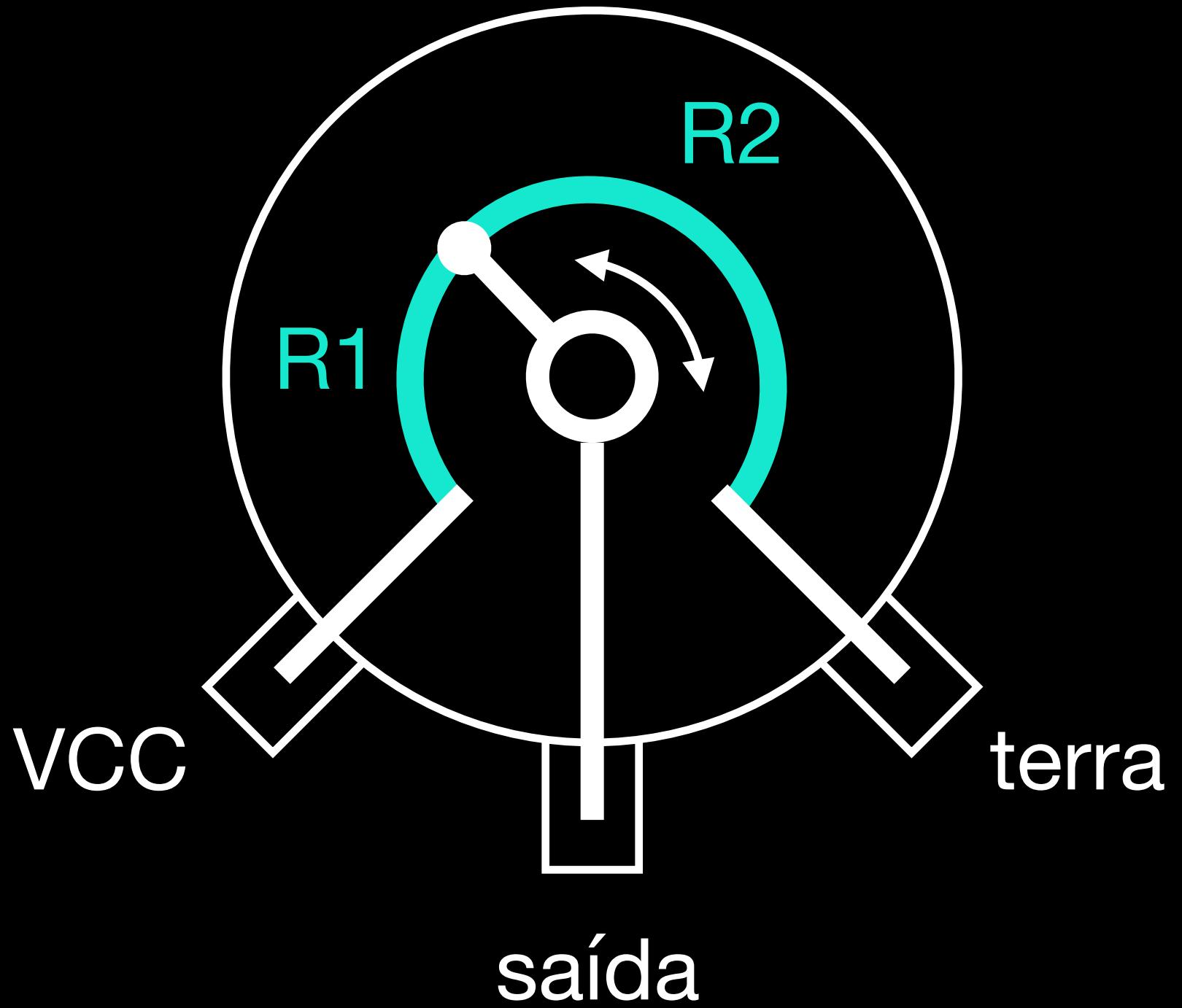
Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

Hardware



Resistor Variável (Potenciômetro)



$$\text{saída} = VCC \frac{R_2}{R_1 + R_2}$$

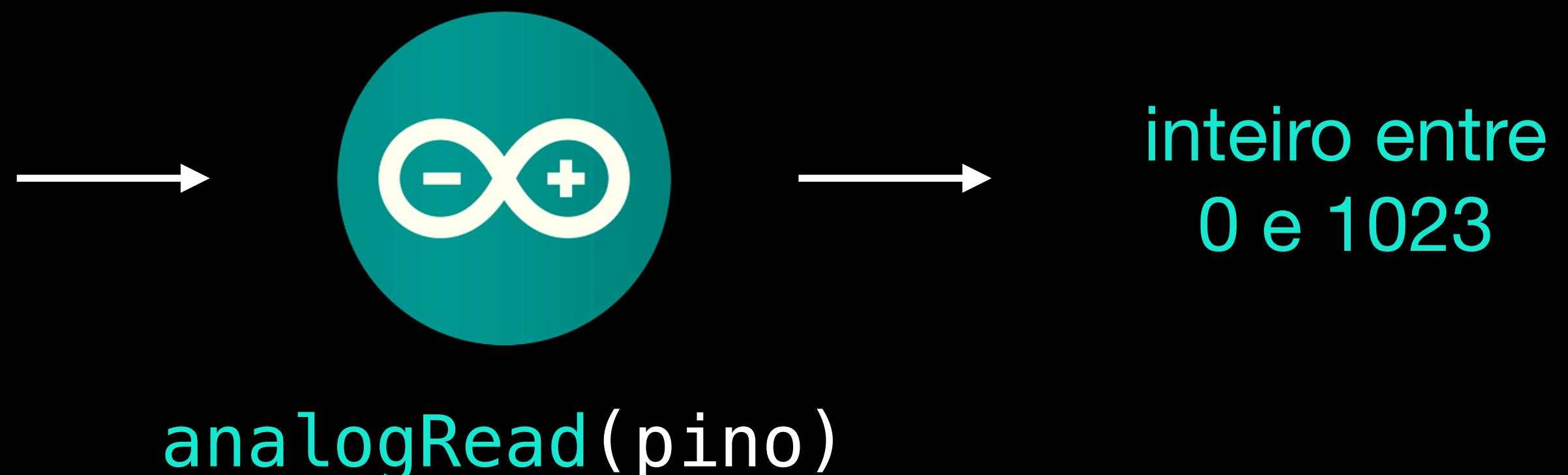
saída varia entre 0 e VCC

entrada digital
(0V ou 5V)



digitalRead(pino)

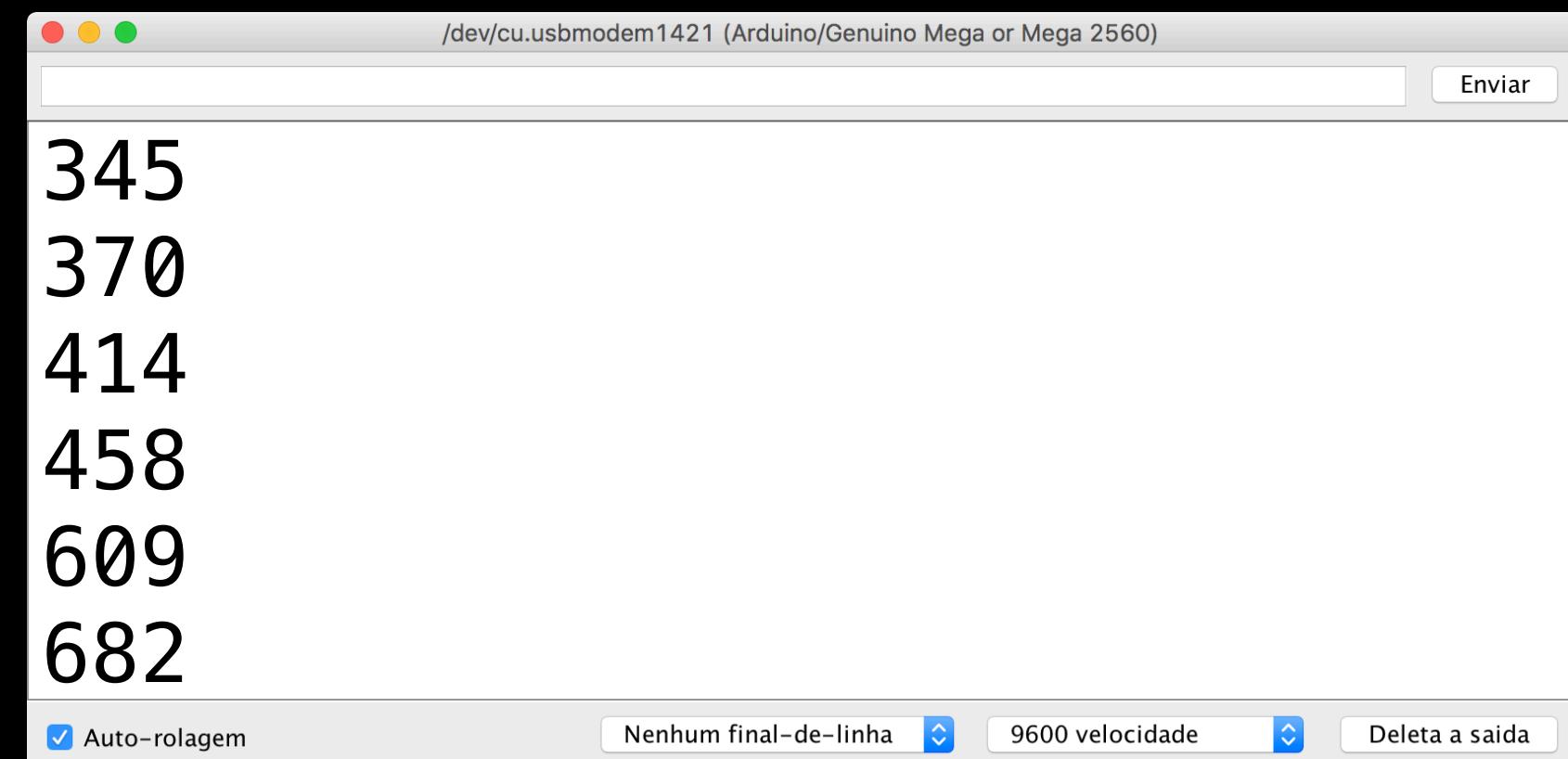
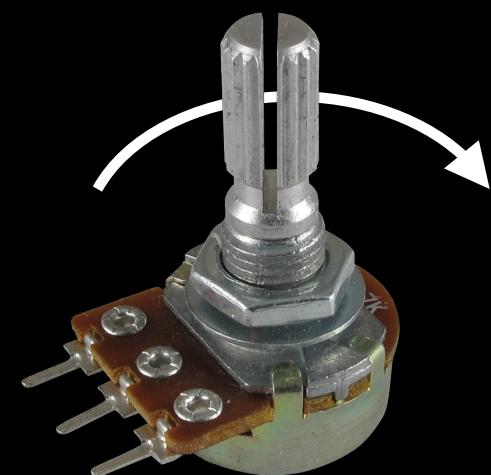
entrada analógica
(entre 0V e 5V)



analogRead(pino)

Leitura de Valores Analógicos

```
int potenciometro = A5;  
void setup () {  
    Serial.begin(9600);  
    pinMode(potenciometro, INPUT);  
}  
  
void loop () {  
    int valorAnalogico = analogRead(potenciometro);  
    Serial.println(valorAnalogico);  
    delay(200);  
}
```



Leitura de Valores Analógicos

0 ... 1023

Y_{Min} ... Y_{Max}

$$y = ax + b$$

$$a = \frac{Y_{\text{Max}} - Y_{\text{Min}}}{1023 - 0}$$

$$b = Y_{\text{Min}} - 0a$$

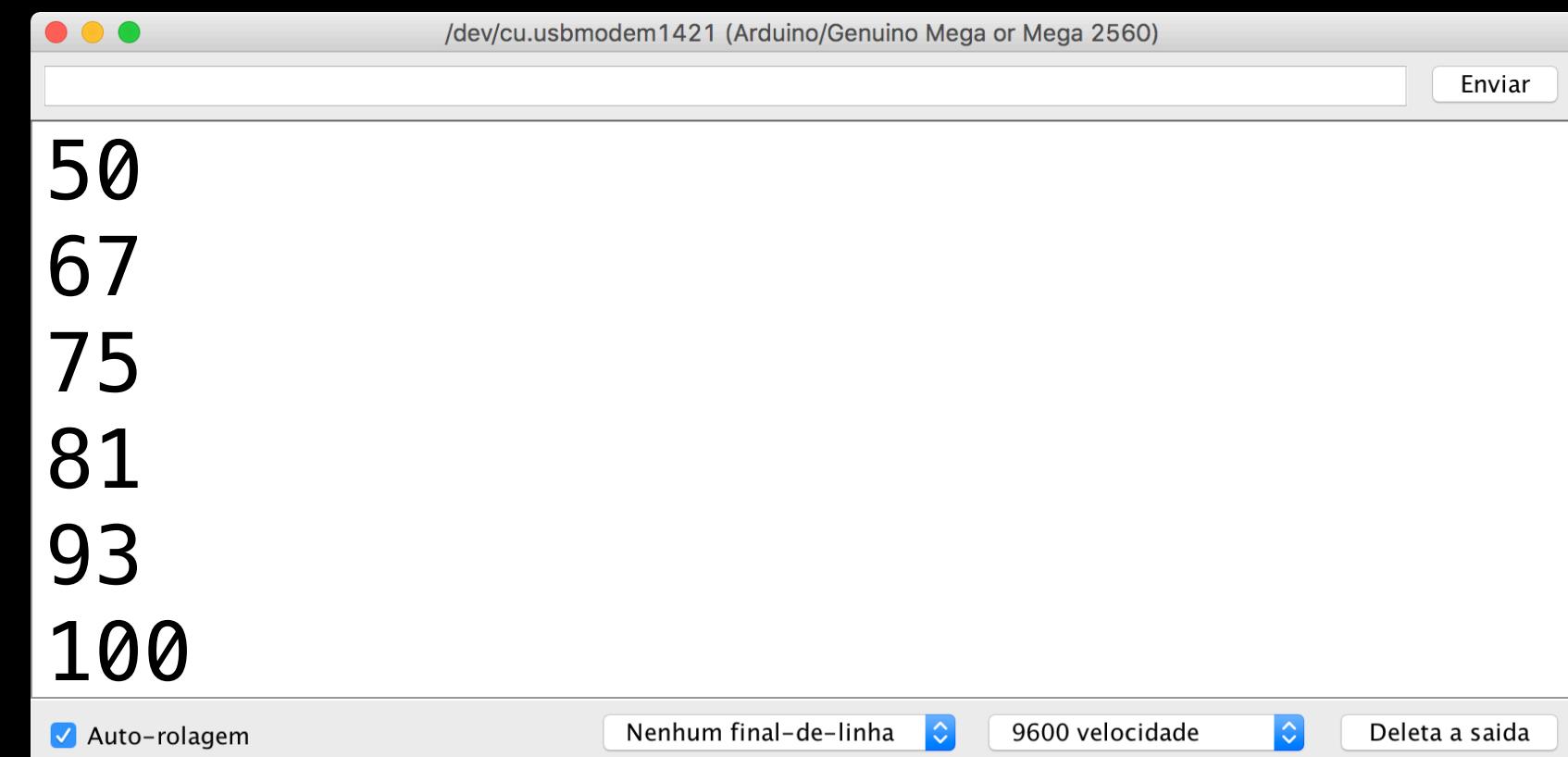
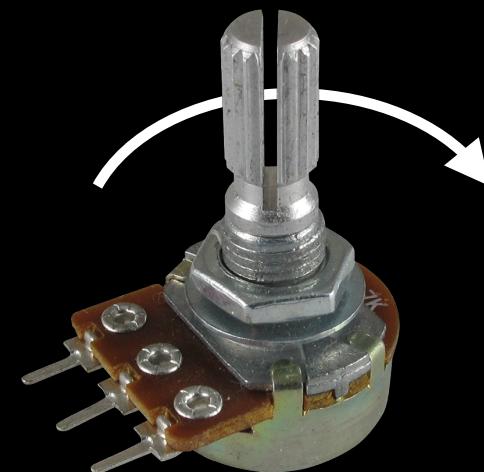
int y = map(valorLido, 0, 1023, yMinimo, yMaximo);

```

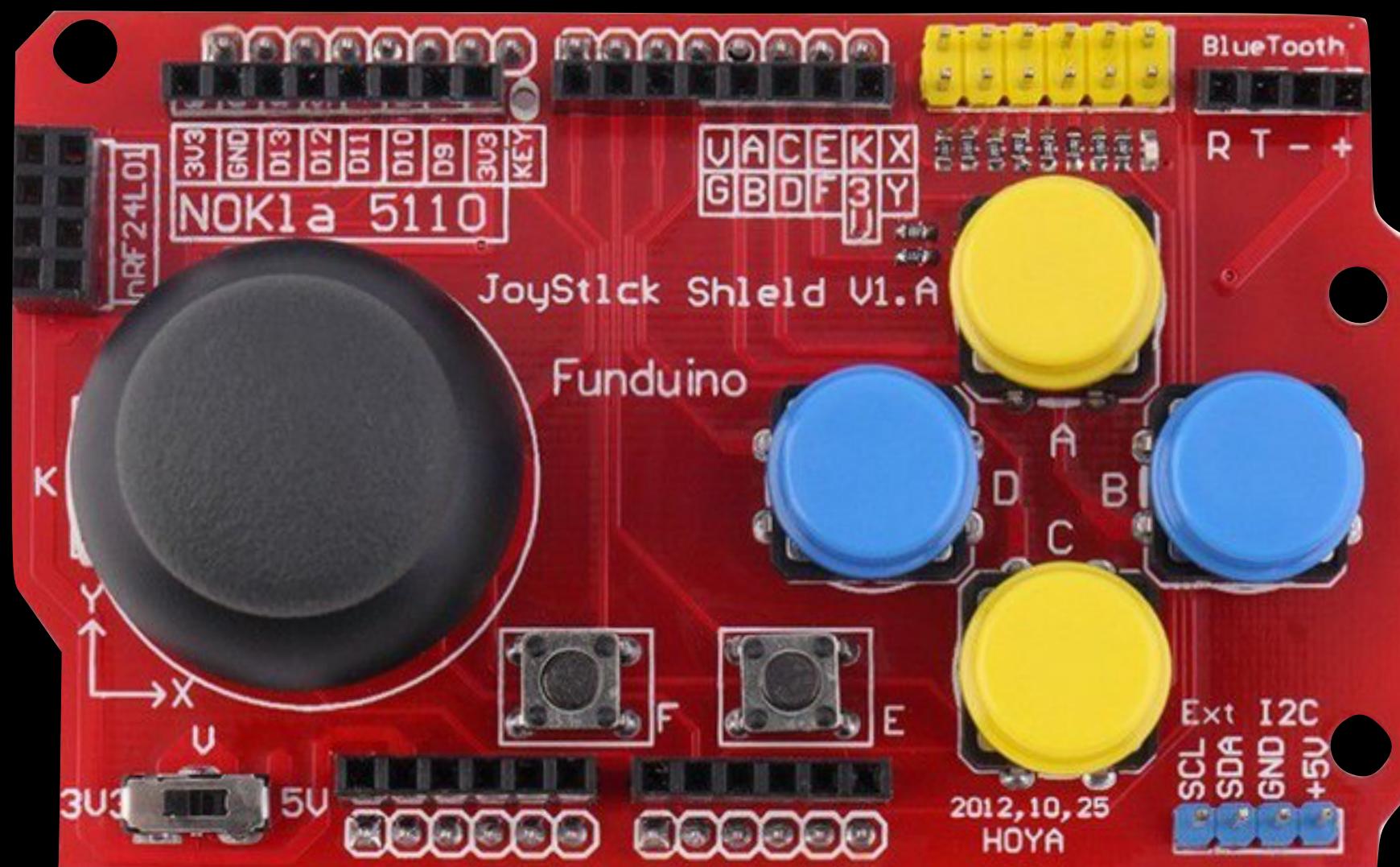
int potenciometro = A5;
void setup () {
    pinMode(potenciometro, INPUT);
}

void loop () {
    int valorLido = analogRead(potenciometro);
    int valorFinal = map(valorLido, 0, 1023, 50, 100);
    Serial.println(valorFinal);
    delay(500);
}

```

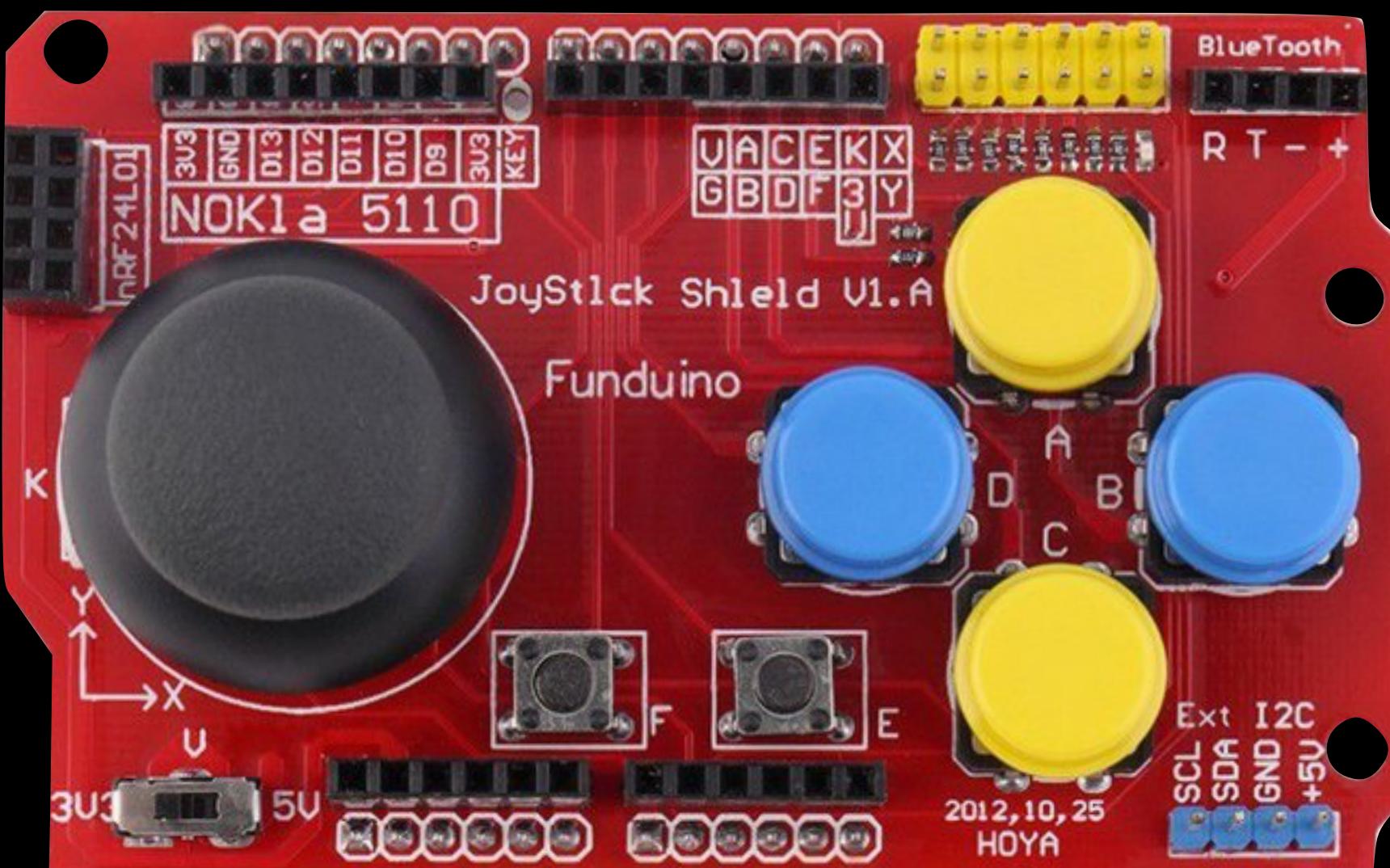
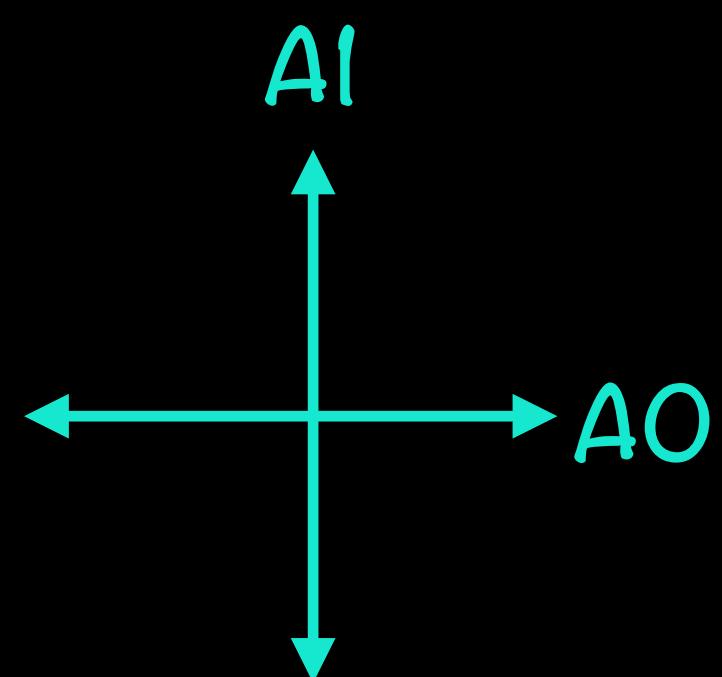


Mapeamento de Valores Analógicos



Shield Joystick

eixos analógicos

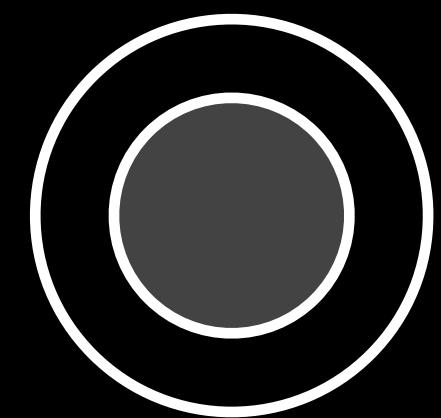
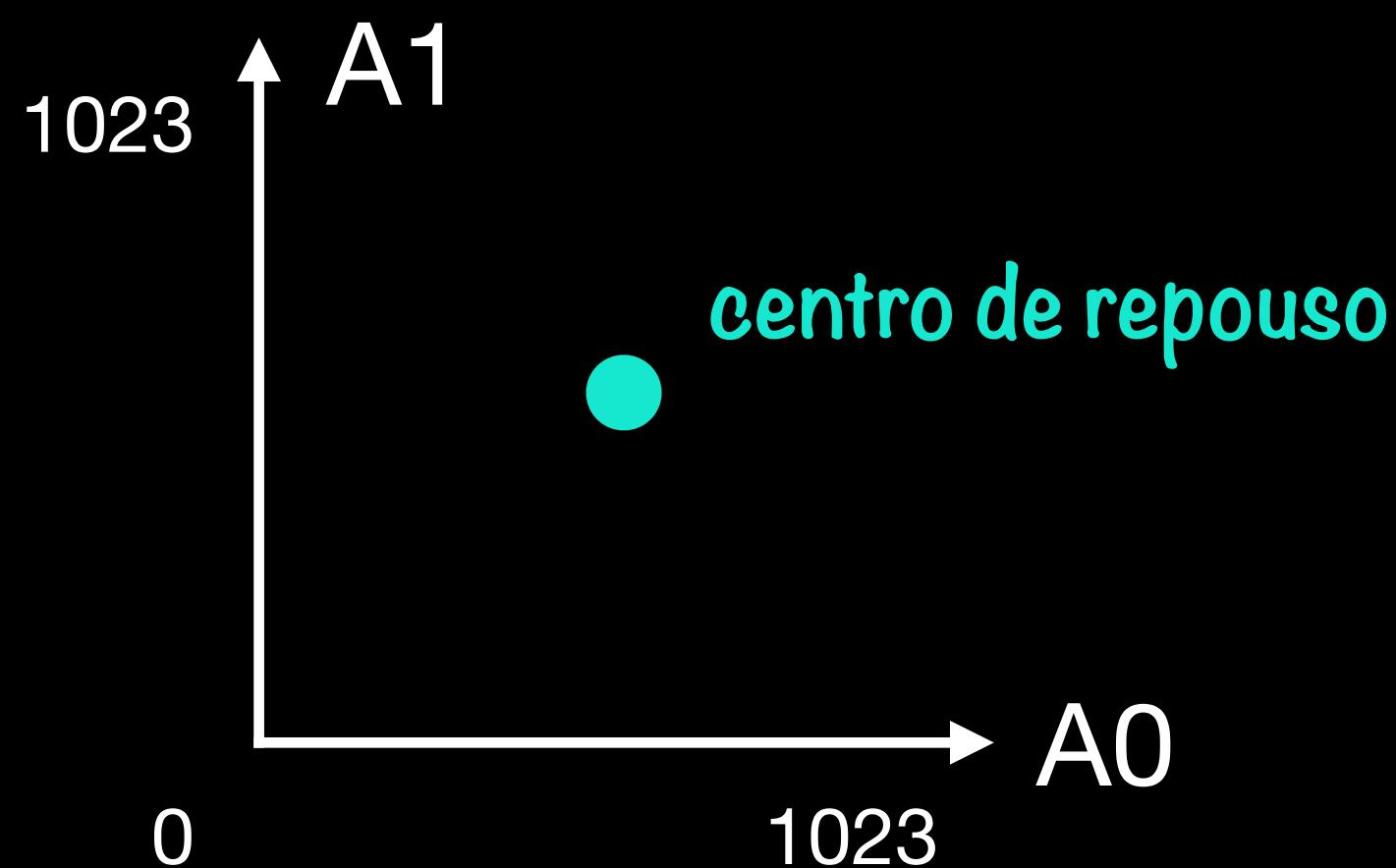


botões digitais

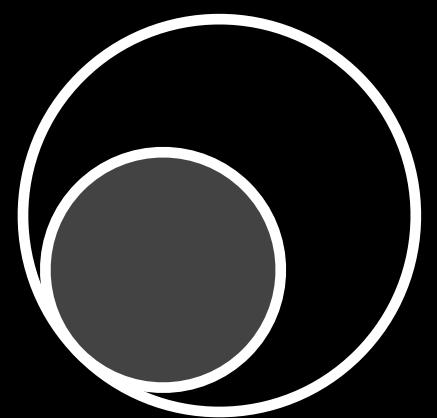
2
5
3
4

7 6
botões digitais

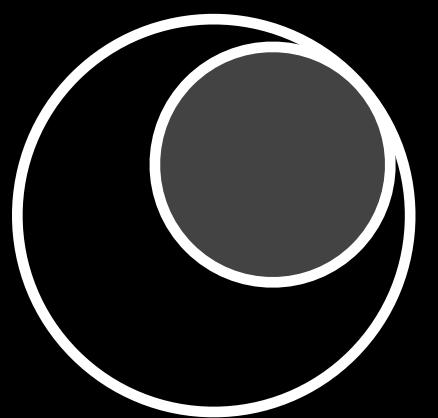
Pino Usados pelo Shield Joystick



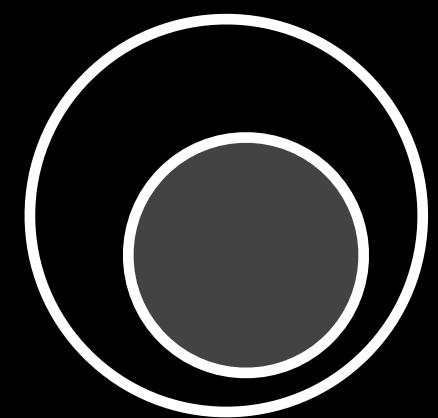
$A_0 = 512$
 $A_1 = 512$



$A_0 = 0$
 $A_1 = 0$



$A_0 = 1023$
 $A_1 = 1023$

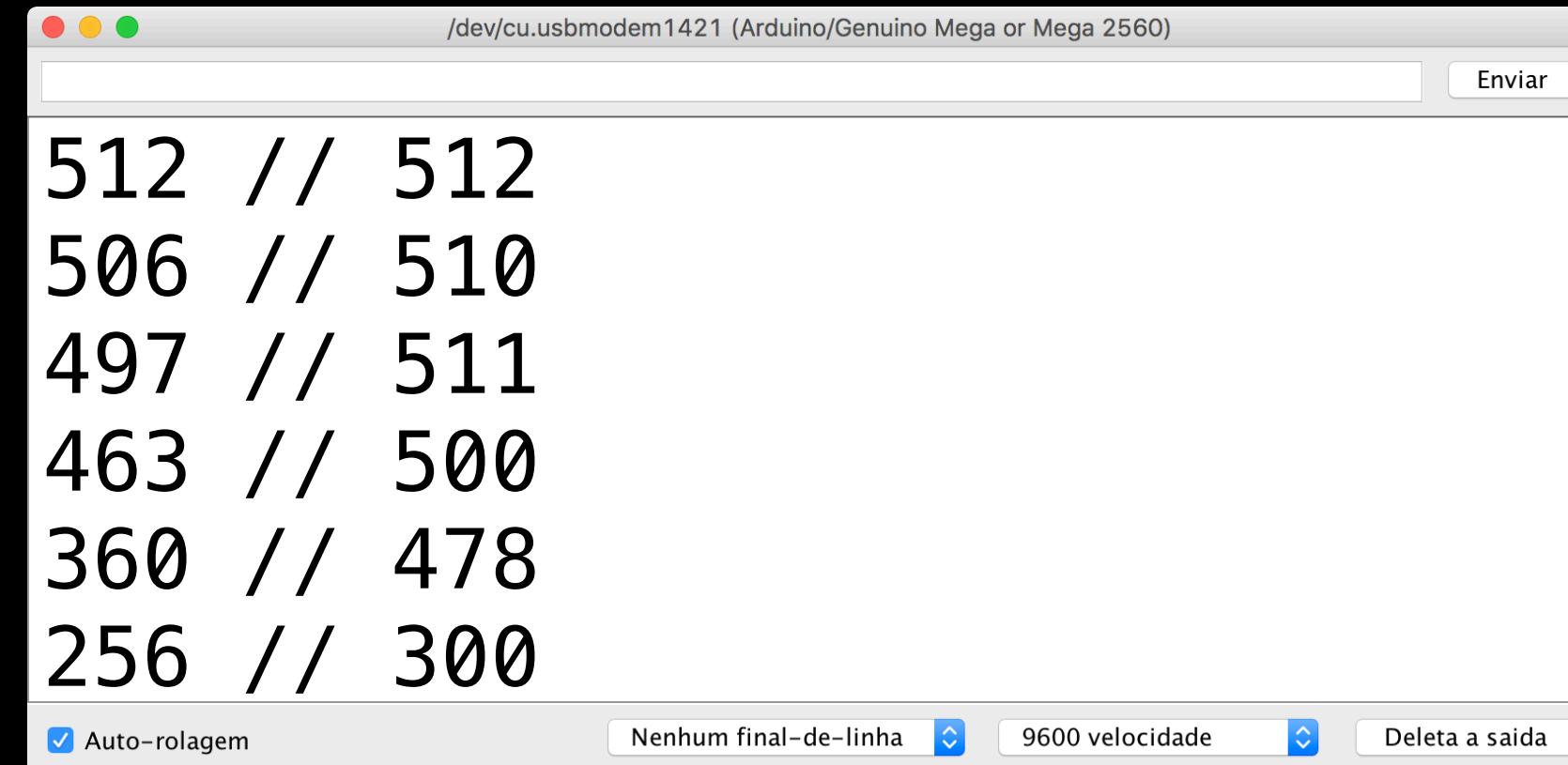


$A_0 = 745$
 $A_1 = 353$

Controle Analógico de Dois Eixos

```
int eixoX = A0;
int eixoY = A1;
void setup () {
    pinMode(eixoX, INPUT);
    pinMode(eixoY, INPUT);
}

void loop () {
    Serial.print( analogRead(eixoX) );
    Serial.print( " // " );
    Serial.println( analogRead(eixoY) );
    delay(200);
}
```

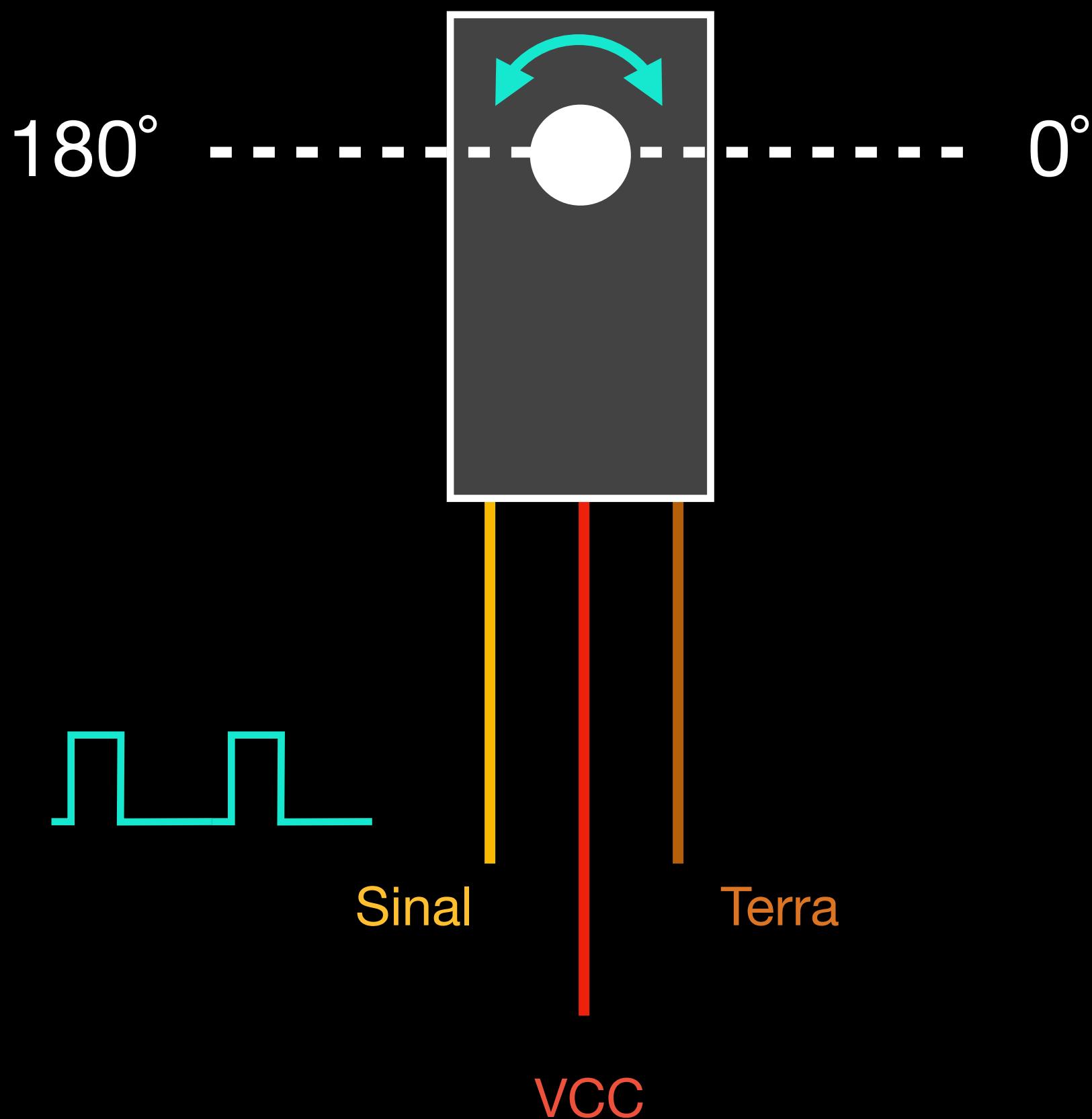


A screenshot of a terminal window titled "/dev/cu.usbmodem1421 (Arduino/Genuino Mega or Mega 2560)". The window displays a series of analog reading values from an Arduino sketch. The values are printed in pairs, separated by a double slash ("//"). The first value is 512 // 512, followed by 506 // 510, 497 // 511, 463 // 500, 360 // 478, and 256 // 300. The terminal interface includes standard controls like 'Enviar' (Send), 'Auto-rolagem' (Autoscroll), and a speed dropdown set to 9600 velocidade (9600 speed).

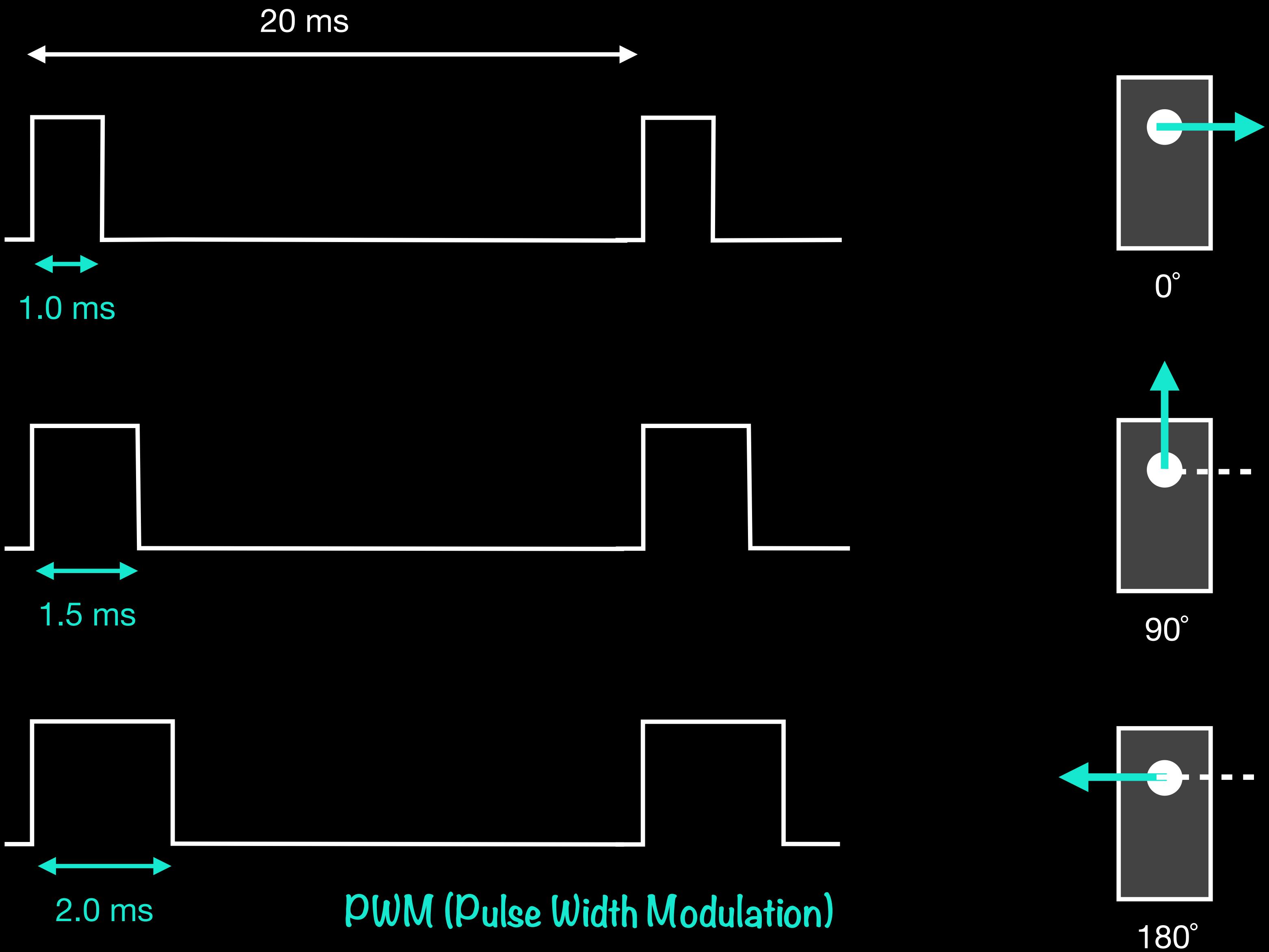
512	//	512
506	//	510
497	//	511
463	//	500
360	//	478
256	//	300



Servomotor



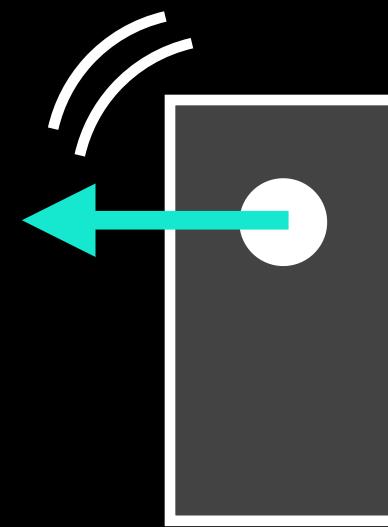
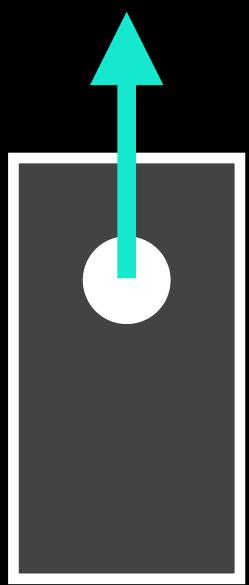
Controle de Giro do Servo



Controle do Ângulo do Servo via Entrada com Pulso

```
#include <Servo.h>

int pinoDoServo = 12;
Servo servo;
void setup () {
    servo.attach(pinoDoServo); // liga o servo
    servo.write(45); // ajusta e trava ângulo em 45°
    delay(500);
    servo.write(90);
    delay(500);
    servo.detach(); // desliga o servo (motor solto)
}
```



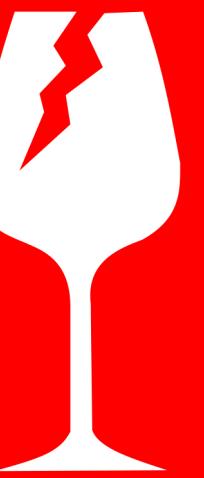
Problema de Movimentos Bruscos do Servo

```
#include <Servo.h>

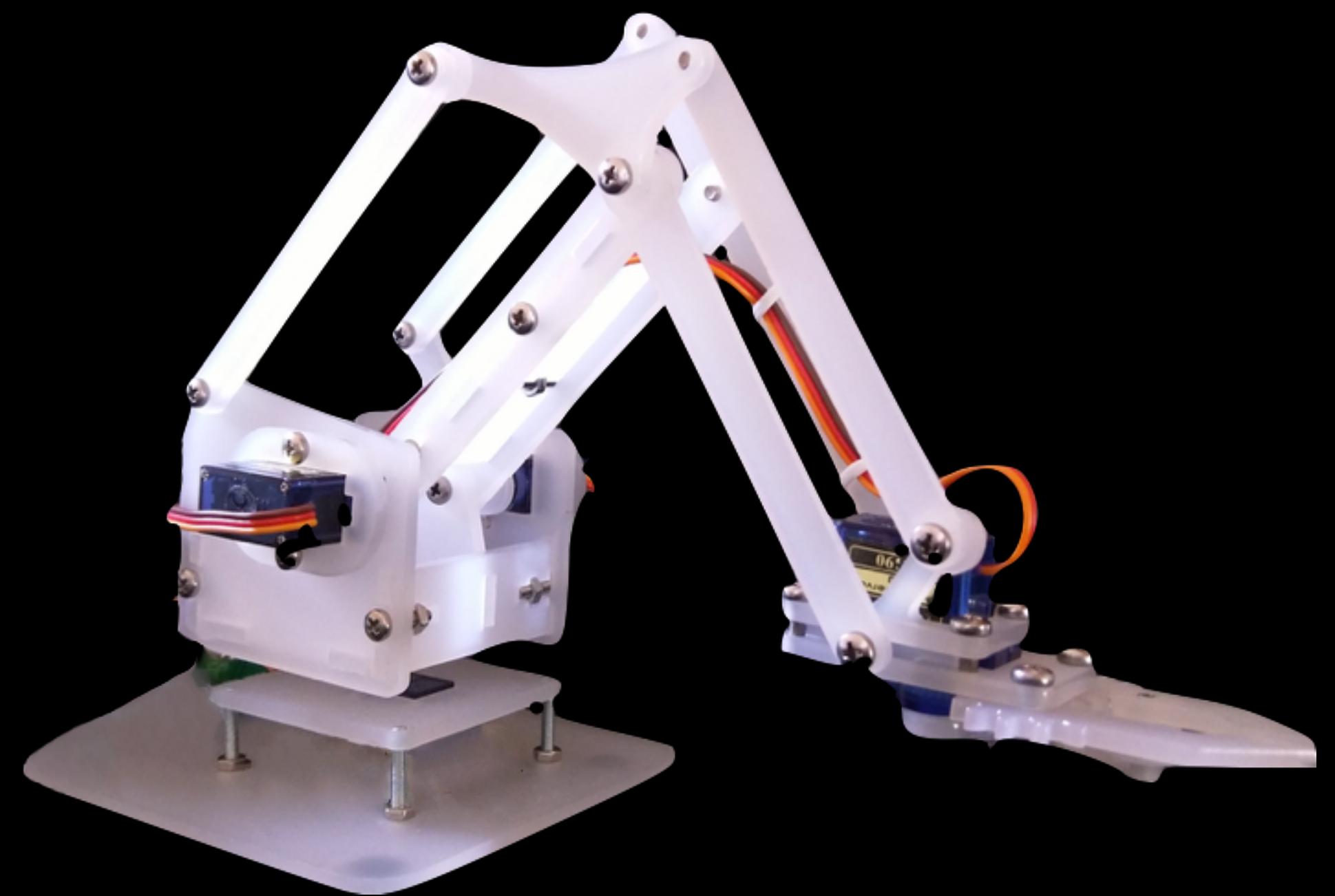
int pinoDoServo = 12;
Servo servo;
void setup () {
    servo.attach(pinoDoServo);
}

void loop () {
    for (int angulo = 0; angulo <= 180; angulo++) {
        servo.write(angulo);
        delay(15);
    }
    for (int angulo = 180; angulo >= 0; angulo--) {
        servo.write(angulo);
        delay(15);
    }
}
```

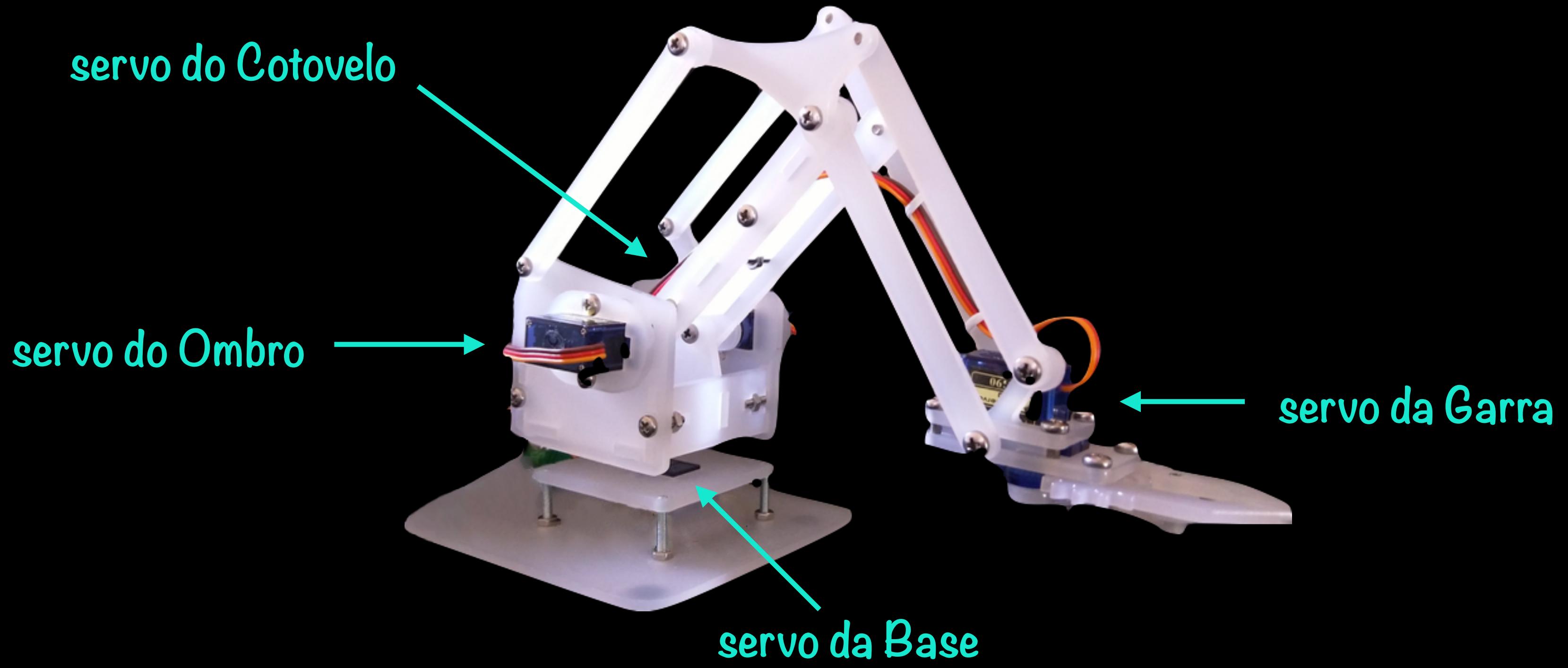
CUIDADO



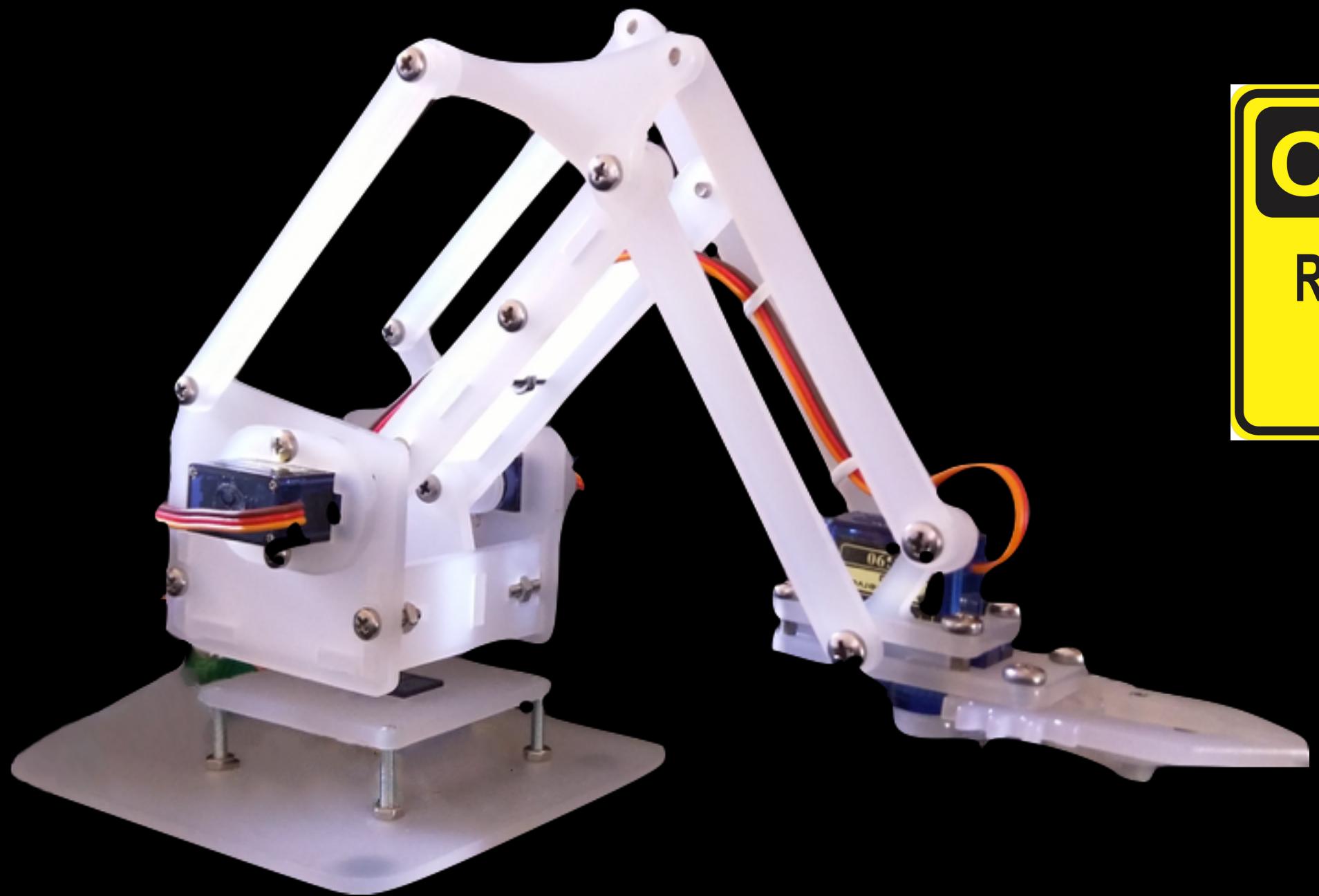
FRÁGIL



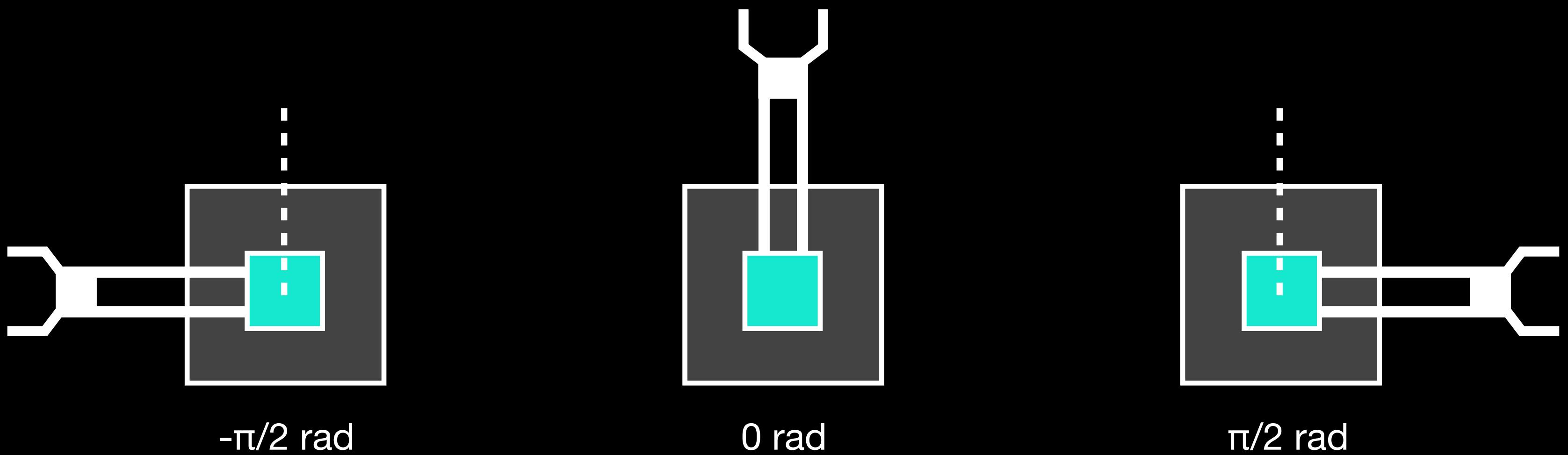
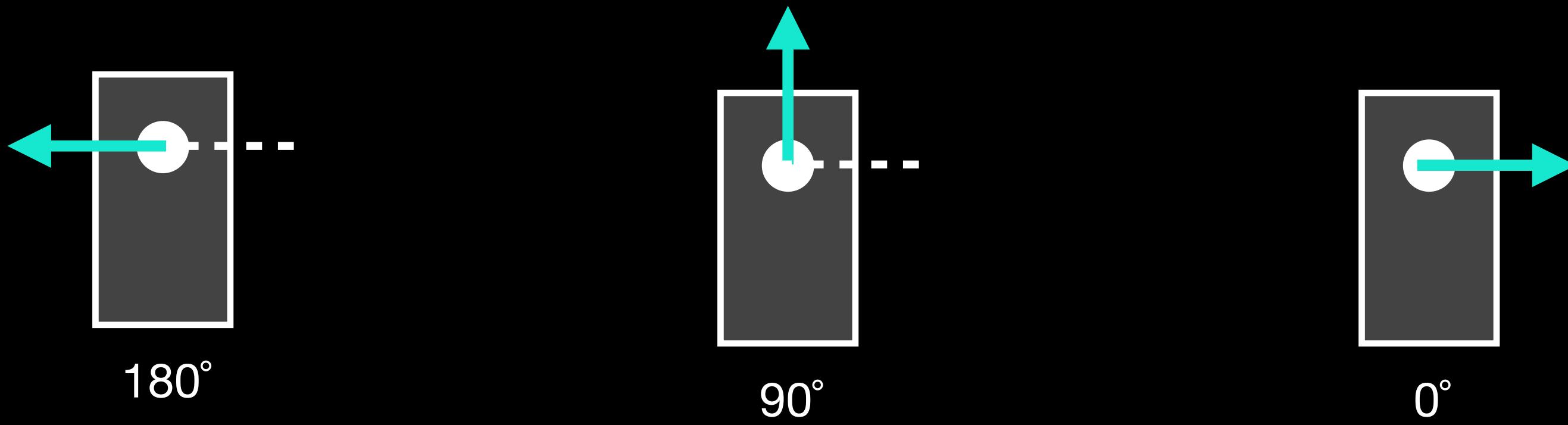
Braço Mecânico (meArm)



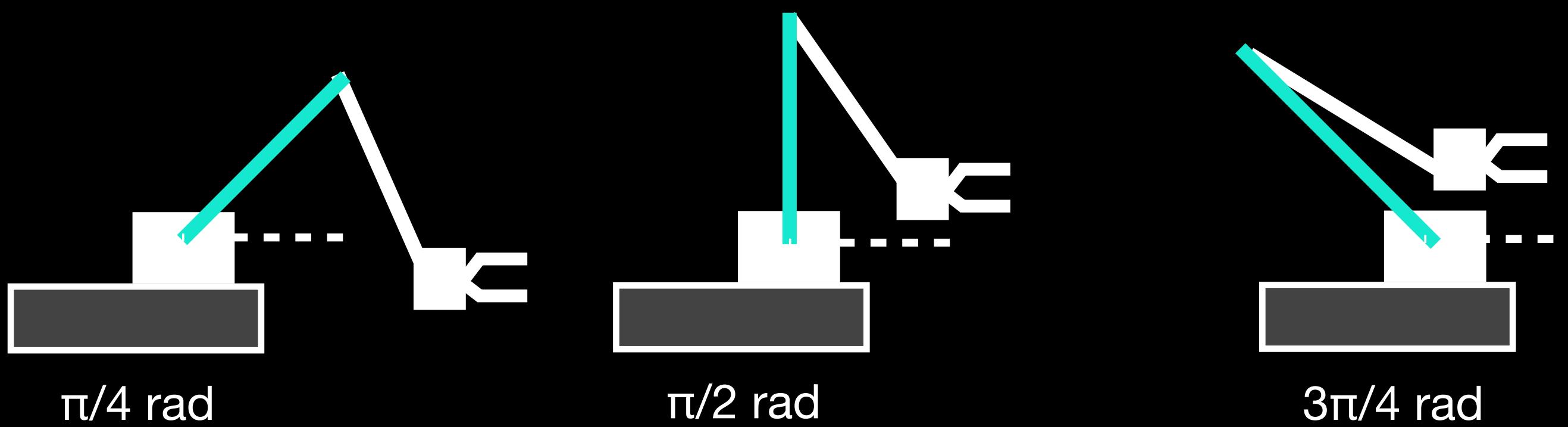
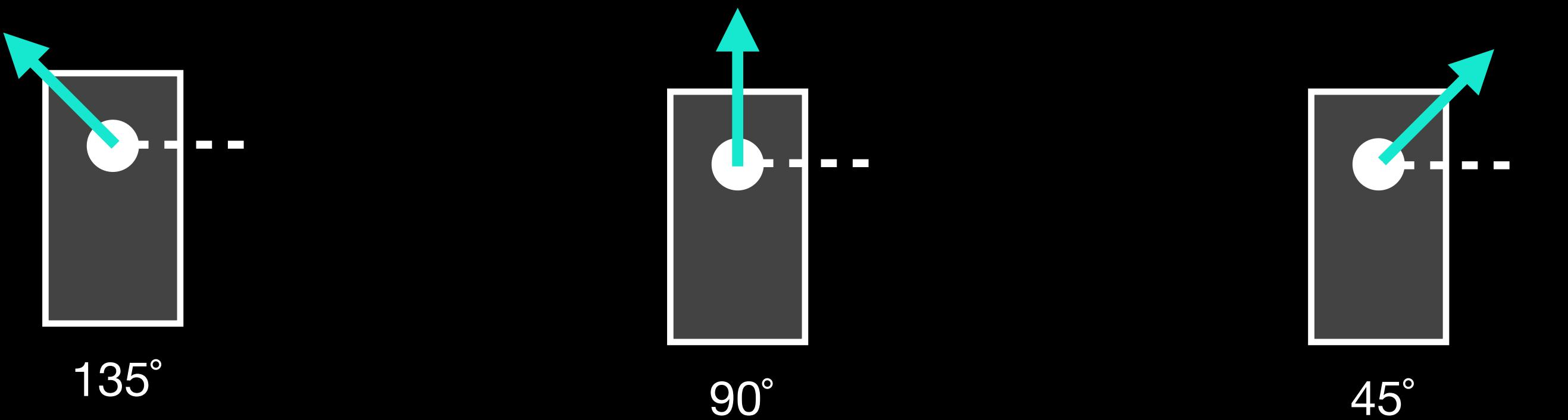
Servos do Braço Mecânico



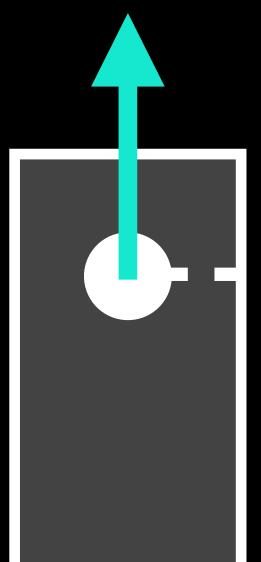
Fonte Variável para Alimentar os Servos



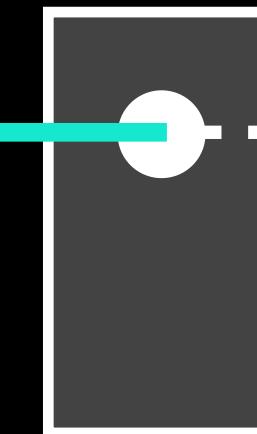
Controle do Ângulo da Base via Servo



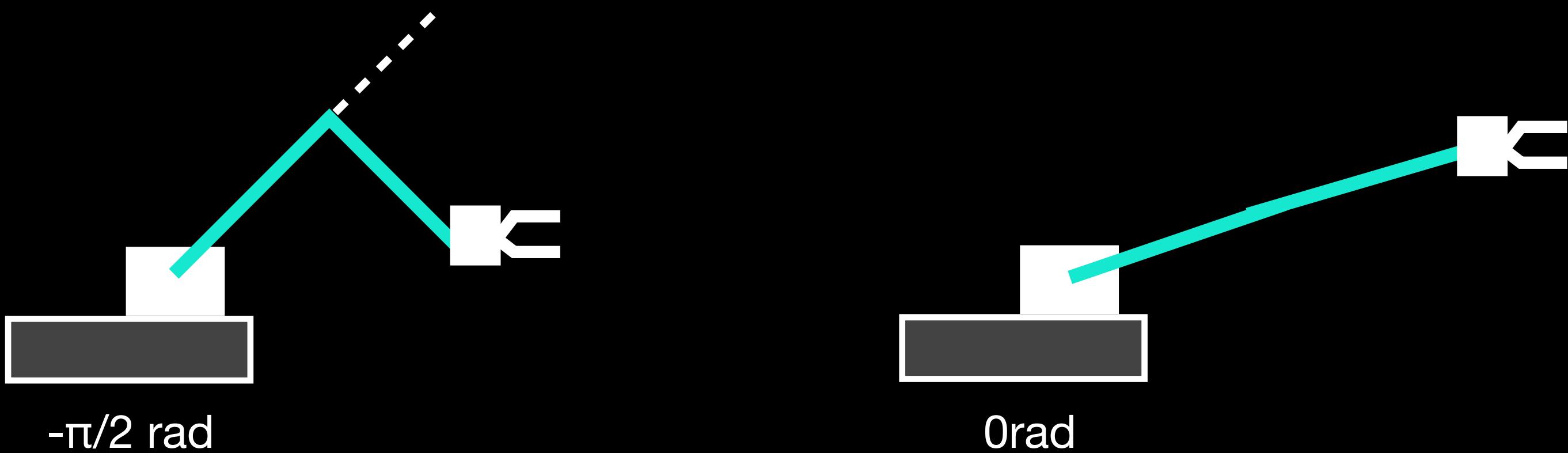
Controle do Ângulo do Ombro via Servo



90°



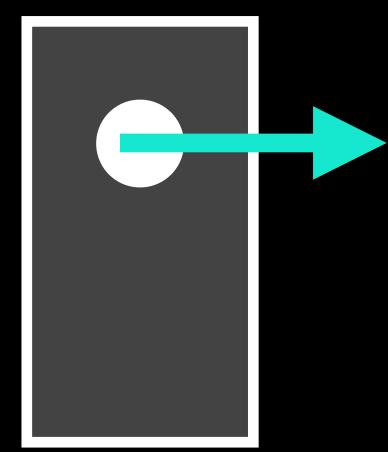
180°



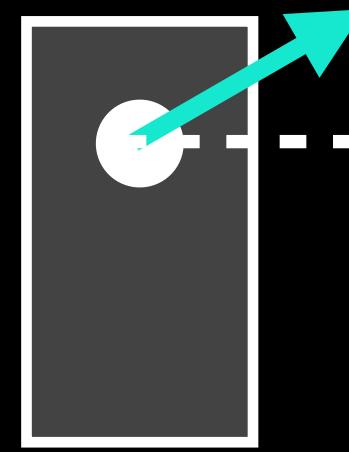
$-\pi/2$ rad

0 rad

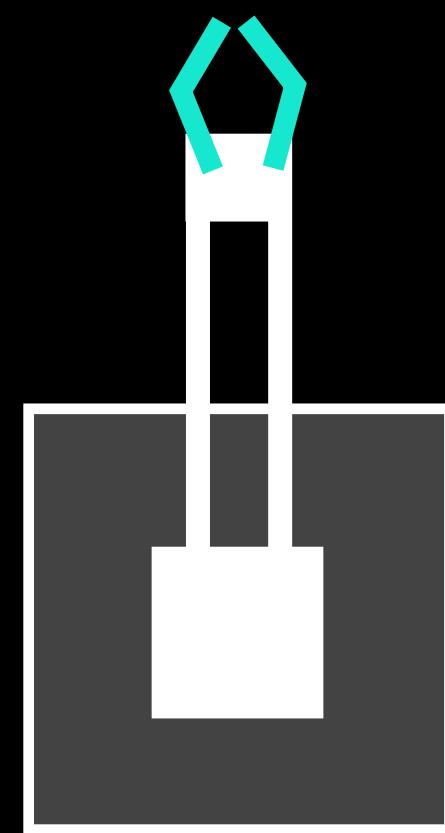
Controle do Ângulo do Ombro via Servo



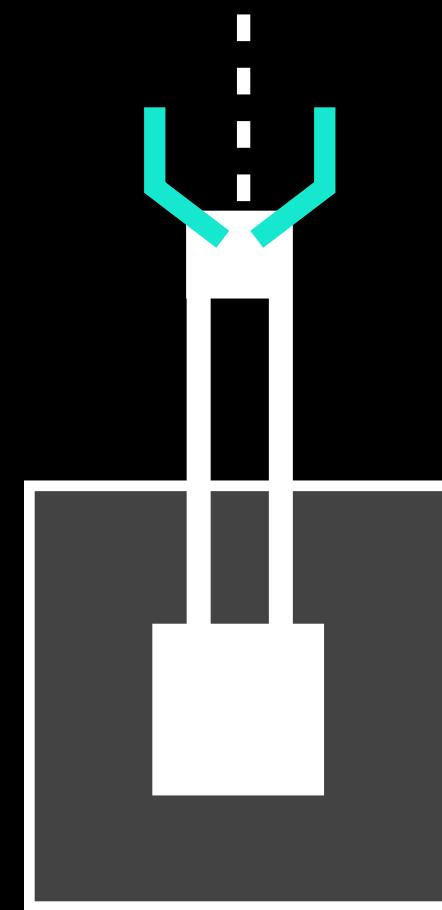
0°



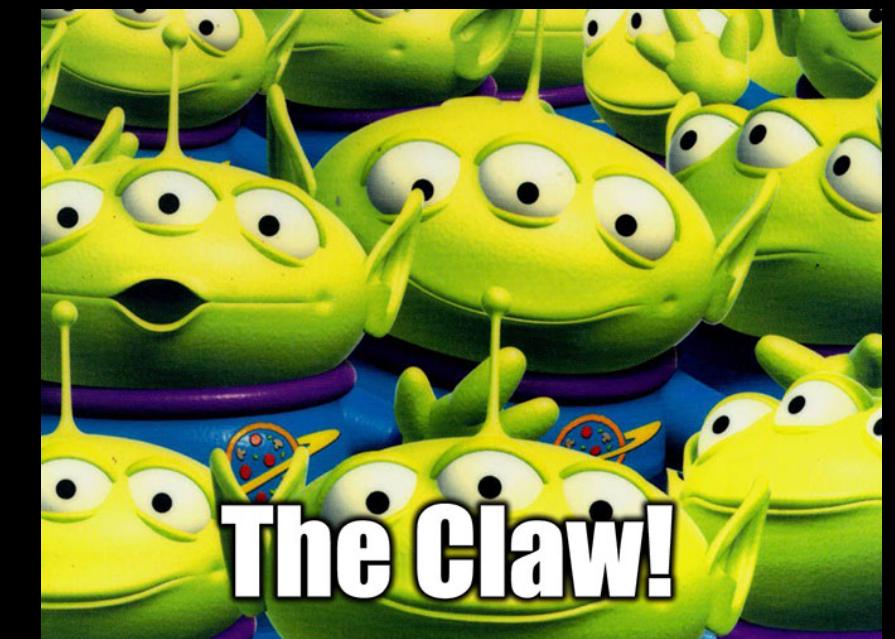
30°



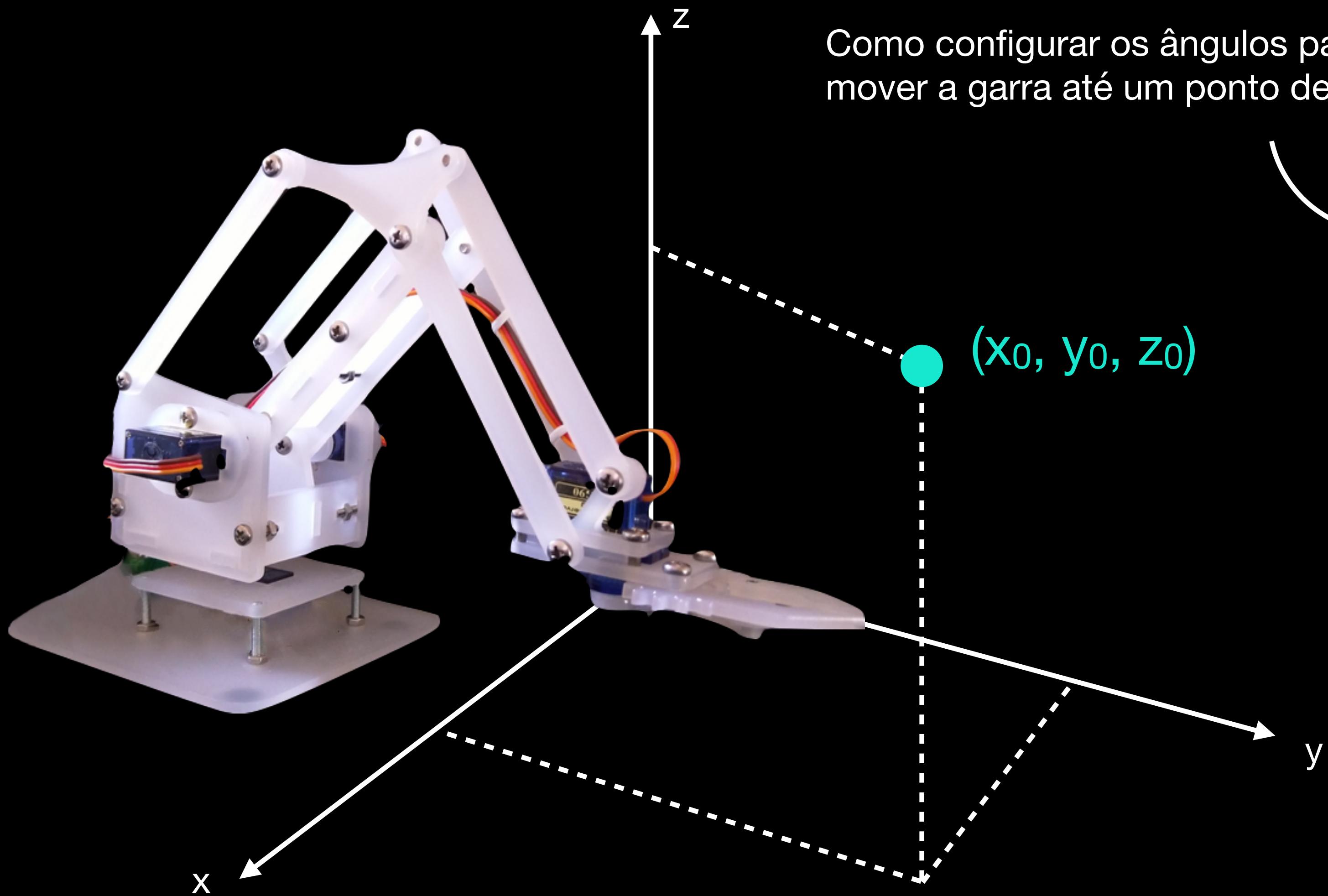
0 rad



$\pi/2$ rad



Controle do Ângulo da Garra via Servo



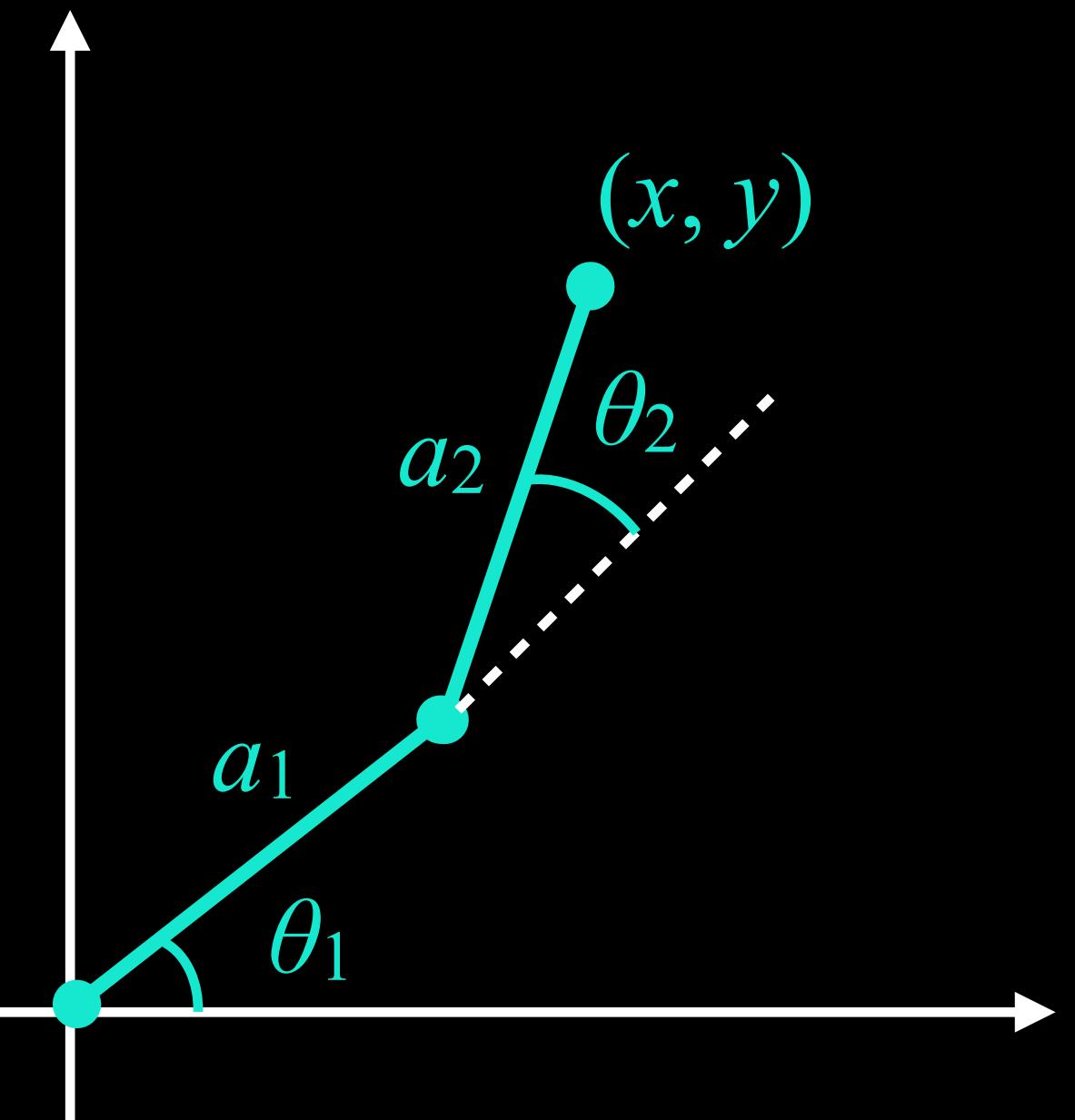
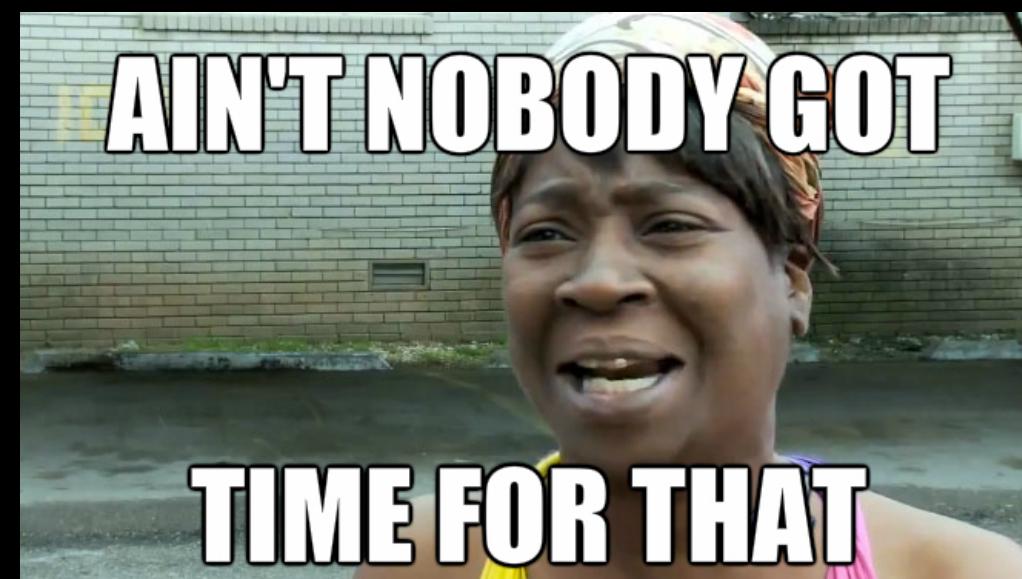
Como configurar os ângulos para mover a garra até um ponto desejado?



Sistema de Coordenadas do Braço Mecânico

$$x^2 + y^2 = a_1^2 + a_2^2 - 2a_1a_2 \cos(\pi - \theta_2)$$

$$\theta_2 = \arccos\left(\frac{x^2 + y^2 - a_1^2 - a_2^2}{2a_1 + 2a_2}\right)$$



$$\tan \alpha = \frac{y}{x} \quad \tan \beta = -\frac{a_2 \sin \theta_2}{a_1 + a_2 \cos \theta_2}$$

$$\tan \theta_1 = \tan(\alpha - \beta) = \dots = \frac{y(a_1 + a_2 \cos \theta_2) - x a_2 \sin \theta_2}{x(a_1 + a_2 \cos \theta_2) + y a_2 \sin \theta_2}$$

A screenshot of a GitHub README.md page for the meArm library. The page is displayed in a web browser window with a light gray header bar containing standard OS X-style icons (red, yellow, green circles, back, forward, search, etc.) and the URL "GitHub, Inc. github.com/yorkhackspace/meArm". The main content area has a white background and features a dark gray header with the text "meArm" and a small icon. Below this is a horizontal line and a brief description: "Inverse kinematics control library for Phenoptix meArm and Arduino." The text continues with an explanation of the library's purpose: "The meArm has four mini servos - one for the gripper, and one each to rotate the base, shoulder joint and elbow joint. But it's not terribly convenient to be specifying things in terms of servo angles when you're much more interested in where you would like to place the gripper, in normal Cartesian (x, y, z) coordinates." A blue underline highlights the sentence "This library solves the angles required to send to the servos in order to meet a given position, allowing for much simpler coding." Further down, there is information about coordinate measurements and a note about the initial 'home' position. At the bottom, there is a section titled "Various other versions of this library exist:" followed by a bulleted list of three items: "Arduino with Adafruit PWM driver board", "Raspberry Pi with Adafruit PWM driver board", and "Beaglebone Black".

```
#include <meArm.h>

int base = 12, ombro = 11, cotovelo = 10, garra = 9;
meArm braco(
    180, 0, -pi/2, pi/2,      // ângulos da base
    135, 45, pi/4, 3*pi/4,   // ângulos do ombro
    180, 90, 0, -pi/2,       // ângulos do cotovelo
    30, 0, pi/2, 0           // ângulos da garra
);
void setup () {
    braco.begin(base, ombro, cotovelo, garra);

    // move rapidamente para x, y, z
    braco.goDirectlyTo(10, 10, 10);
    // move suavemente para x, y, z
    braco.gotoPoint(50, 60, 20);

    braco.openGripper();
    delay(500);
    braco.closeGripper();
}
```

```
float coordenadaAtualX = braco.getX();
float coordenadaAtualY = braco.getY();
float coordenadaAtualZ = braco.getZ();

braco.end(); // desliga os servos
```

"Software"

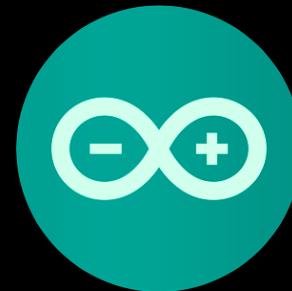
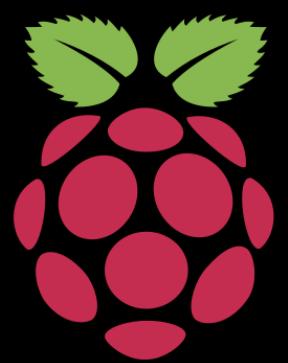
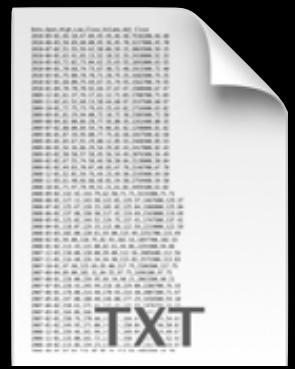
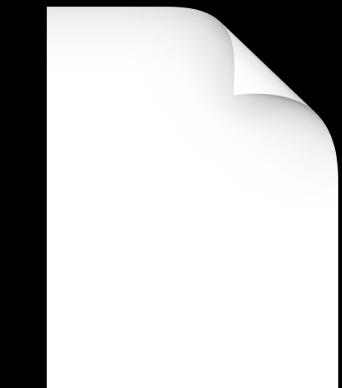


imagem.jpeg



musicas.txt



ajustes.json

não tem nem
sistema operacional

mas...



Armazenamento de Dados

The screenshot shows a web browser window displaying the Arduino Reference EEPROM page. The URL in the address bar is www.arduino.cc/en/Reference/EEPROM. The page has a teal header with the Arduino logo, a search icon, a shopping cart icon, a 'SIGN IN' button, and a menu icon. Below the header, there are navigation links: 'Reference' (highlighted), 'Language | Libraries | Comparison | Changes'. The main title is 'EEPROM Library'. The content describes the EEPROM library and its functions.

The microcontroller on the Arduino and Genuino AVR based board has EEPROM:
memory whose values are kept when the board is turned off (like a tiny hard drive).
This library enables you to read and write those bytes.

The supported micro-controllers on the various Arduino and Genuino boards have different amounts of EEPROM: 1024 bytes on the ATmega328P, 512 bytes on the ATmega168 and ATmega8, 4 KB (4096 bytes) on the ATmega1280 and ATmega2560.
The Arduino and Genuino 101 boards have an emulated EEPROM space of 1024 bytes.

Functions

- [read\(\)](#)
- [write\(\)](#)
- [update\(\)](#)
- [get\(\)](#)
- [put\(\)](#)
- [EEPROM\[\]](#)

endereço

byte #0

byte #1

byte #2

byte #3

byte #4

byte #5

byte #6

...

byte #4096

número inteiro

0	1	0	1	1	0	0	0
1	1	0	0	0	1	1	0

número decimal

1	1	1	0	1	0	1	1
0	1	1	1	1	0	0	1
0	0	1	1	0	0	0	0
1	1	0	1	0	0	1	1

```
#include <EEPROM.h>

int valorInteiro;
int endereco = 0;

void setup () {
    EEPROM.get(endereco, valorInteiro);

    valorInteiro = valorInteiro * 2 + 30;
    EEPROM.put(endereco, valorInteiro);
}
```

```
int inteiro; // inteiro ocupa 2 bytes de memória
int endereco = 0;
EEPROM.get(endereco, inteiro);
EEPROM.put(endereco, inteiro * 2 + 30);

float decimal; // decimal ocupa 4 bytes de memória
int endereco = 0 + 2;
EEPROM.get(endereco, decimal);
EEPROM.put(endereco, decimal/23);

bool listaDeBooleanos[3]; // lista de bool ocupa 3 bytes
int endereco = 0 + 2 + 4;
EEPROM.get(endereco, listaDeBooleanos);
listaDeBooleanos[0] = true;
listaDeBooleanos[1] = !listaDeBooleanos[1];
EEPROM.put(endereco, listaDeBooleanos);
```

Note

An EEPROM write takes 3.3 ms to complete. The EEPROM memory has a specified **life of 100,000 write/erase cycles**, so you may need to be careful about how often you write to it.

```
int minhaVariavel = 2;  
  
void loop () {  
    minhaVariavel = minhaVariavel + 1.5;  
    EEPROM.put(0, minhaVariavel);  
}
```

Resumo da Ópera

Funcionalidade

Leitura Analógica
[acessar documentação](#)

Servomotor
[acessar documentação](#)

Braço Mecânico
[acessar documentação](#)

EEPROM
[acessar documentação](#)

Comandos

```
int potenciometro = A5;  
pinMode(potenciometro, INPUT);  
int valorAnalogico = analogRead(potenciometro);  
int valorMapeado = map(valorAnalogico, 0, 1023, min, max);  
  
#include <Servo.h>  
Servo servo;  
servo.attach(pino); servo.detach();  
servo.write(anguloEmGraus);  
  
#include <meArm.h>  
int base = 12, ombro = 11, cotovelo = 10, garra = 9;  
meArm braco(  
    180, 0, -pi/2, pi/2, // ângulos da base  
    135, 45, pi/4, 3*pi/4, // ângulos do ombro  
    180, 90, 0, -pi/2, // ângulos do cotovelo  
    30, 0, pi/2, 0 // ângulos da garra  
);  
braco.begin(base, ombro, cotovelo, garra);  
braco.goDirectlyTo(x, y, z); braco.gotoPoint(x, y, z);  
braco.openGripper(); braco.closeGripper();  
braco.getX(); braco.getY(); braco.getZ(); braco.end();  
  
#include <EEPROM.h>  
EEPROM.get(endereco, minhaVariavel);  
EEPROM.put(endereco, minhaVariavel);
```

Funcionalidade

Campainha Passiva
[acessar documentação](#)

Comandos

```
int campainhaPassiva = 5;  
pinMode(campainhaPassiva, OUTPUT);  
int frequencia = 220; int duracaoEmMs = 500;  
tone(campainhaPassiva, frequencia);  
tone(campainhaPassiva, frequencia, duracaoEmMs);  
noTone(campainhaPassiva);
```

Interrupção
[acessar documentação](#)

```
int sensorDeSom = 19;  
pinMode(sensorDeSom, INPUT);  
int origem = digitalPinToInterrupt(sensorDeSom);  
attachInterrupt(origem, minhaFuncao, RISING);
```

Contagem
de Tempo
[acessar documentação](#)

```
unsigned long instanteAnteriorDeDeteccao = 0;  
  
if (millis() > instanteAnteriorDeDeteccao + 10) {  
    instanteAnteriorDeDeteccao = millis();  
}
```

Encoder Rotativo
[acessar documentação](#)

```
#include <RotaryEncoder.h>  
RotaryEncoder encoder(20, 21);  
attachInterrupt(digitalPinToInterrupt(20), funcao, CHANGE);  
attachInterrupt(digitalPinToInterrupt(21), funcao, CHANGE);  
encoder.tick(); int posicao = encoder.getPosition();
```

Funcionalidade

Revisão de C++

Print Serial

Escrita/Leitura acessar documentação

GButton acessar documentação

ShiftDisplay acessar documentação

Timer1 acessar documentação

Comandos

```
int inteiro = 2; float decimal = 4.5; bool booleano = true;  
char texto[] = "Olá"; int listaDeInteiros[] = {1, 2, 3, 4};  
  
if (x > 0 && y > 0) {  
    z = 1;  
}  
else if (x < 0 || y < 0) {  
    z = 2;  
}
```

```
for (int i = 0; i < 5; i++) {  
    Serial.println(i);  
}  
float soma (float x) {  
    return x + 2;  
}
```

```
Serial.begin(9600); Serial.println("Olá"); Serial.println(2);
```

```
int led = 13; pinMode(led, OUTPUT); digitalWrite(led, LOW);  
int campainha = 3; digitalWrite(campainha, HIGH);  
int botao = A1; pinMode(botao, INPUT); digitalRead(botao) == LOW
```

```
#include <GButton.h>  
GButton botao(A1); botao.isPressed(); botao.process();  
botao.setPressHandler(funcao); botao.setReleaseHandler(funcao);
```

```
#include <ShiftDisplay.h>  
ShiftDisplay display(4, 7, 8, COMMON_ANODE, 4, true);  
ShiftDisplay display(4, 7, 8, COMMON_CATHODE, 4, true);  
display.set(1234); display.set(4.21, 2); display.set("Erro");  
display.update(); display.show(1000);
```

```
#include <TimerOne.h>  
Timer1.initialize(1000000); Timer1.attachInterrupt(funcao);
```