Projeto 08

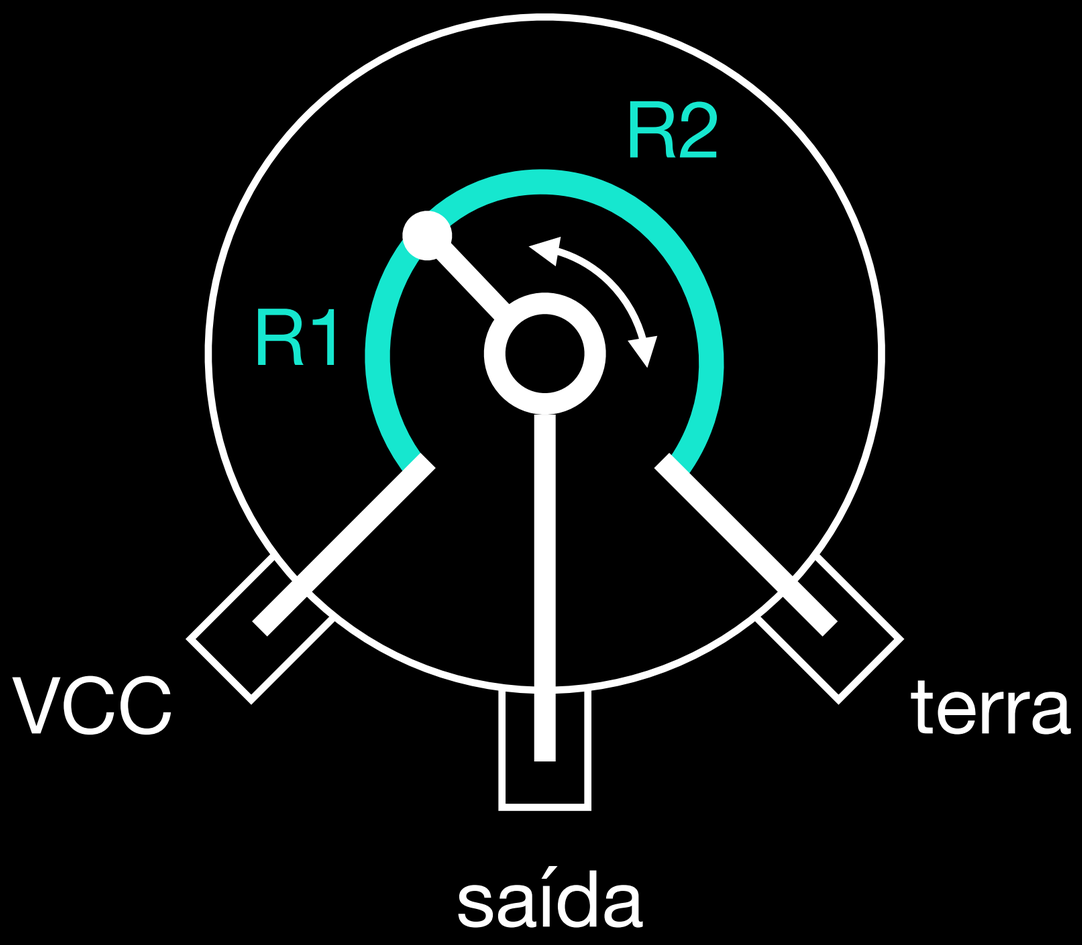
Controle Analógico – Teoria

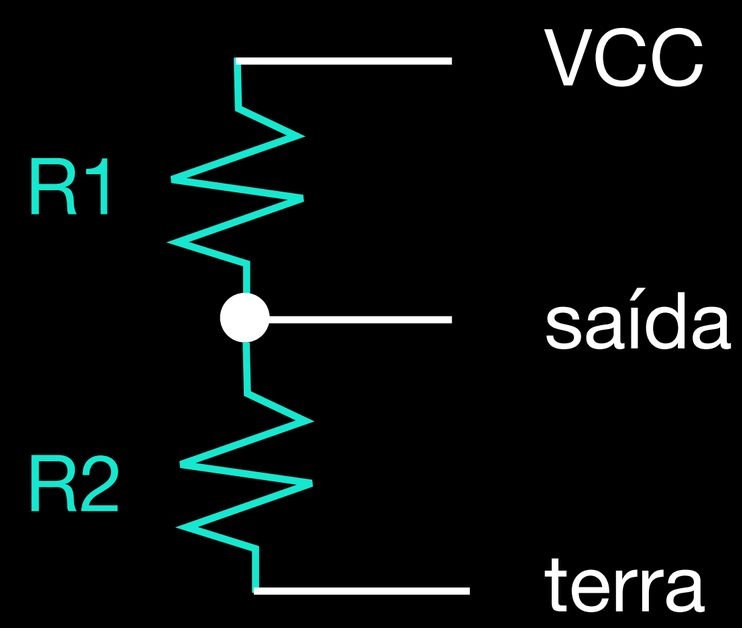
Jan K. S. – janks@puc-rio.br

ENG1419 – Programação de Microcontroladores

Hardware

Resistor Variável (Potenciômetro)



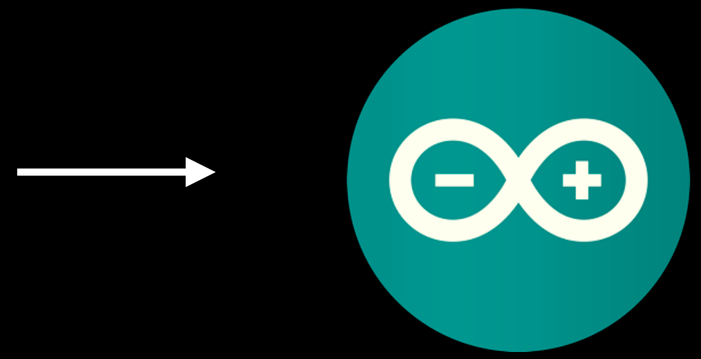


$$\text{saída} = \text{VCC}\ \frac{\text{R2}}{\text{R1} + \text{R2}}$$

saída varia entre 0 e VCC

Funcionamento Interno do Resistor Variável

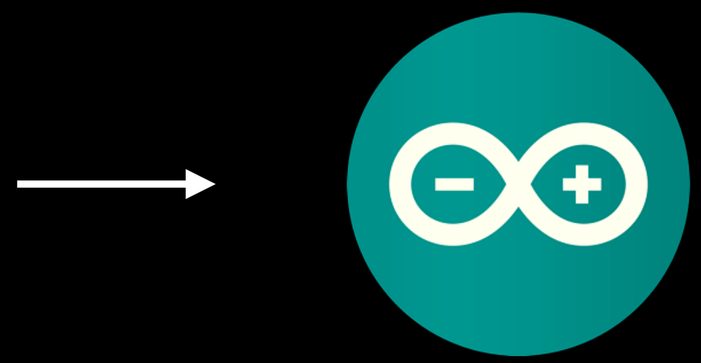
entrada digital
(0V ou 5V)



`LOW` ou `HIGH`

`digitalRead(pino)`

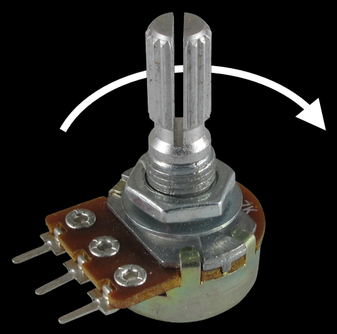
entrada analógica
(entre 0V e 5V)

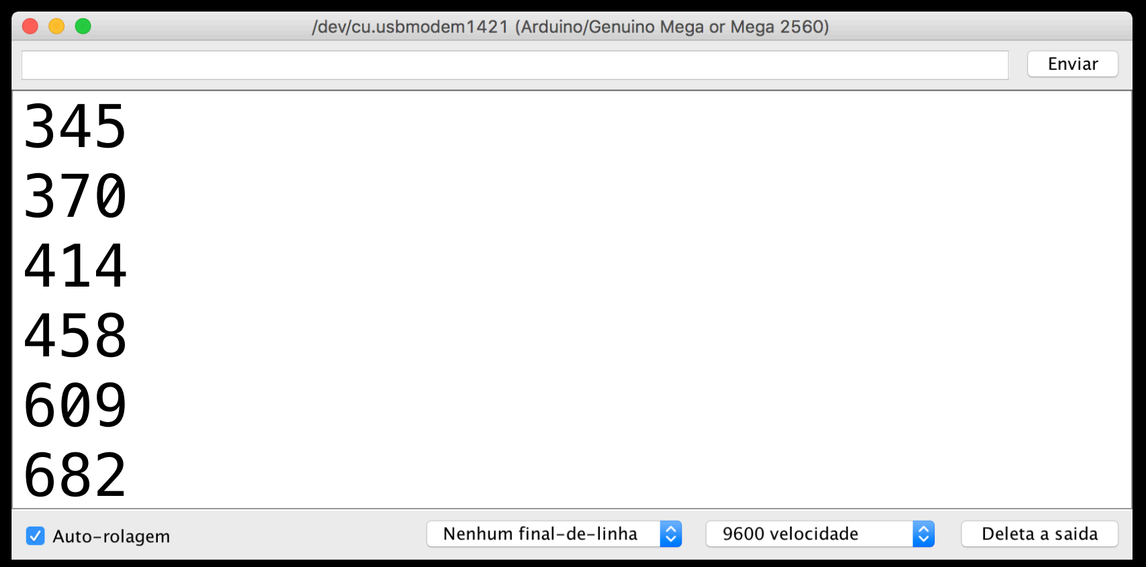


inteiro entre
0 e 1023

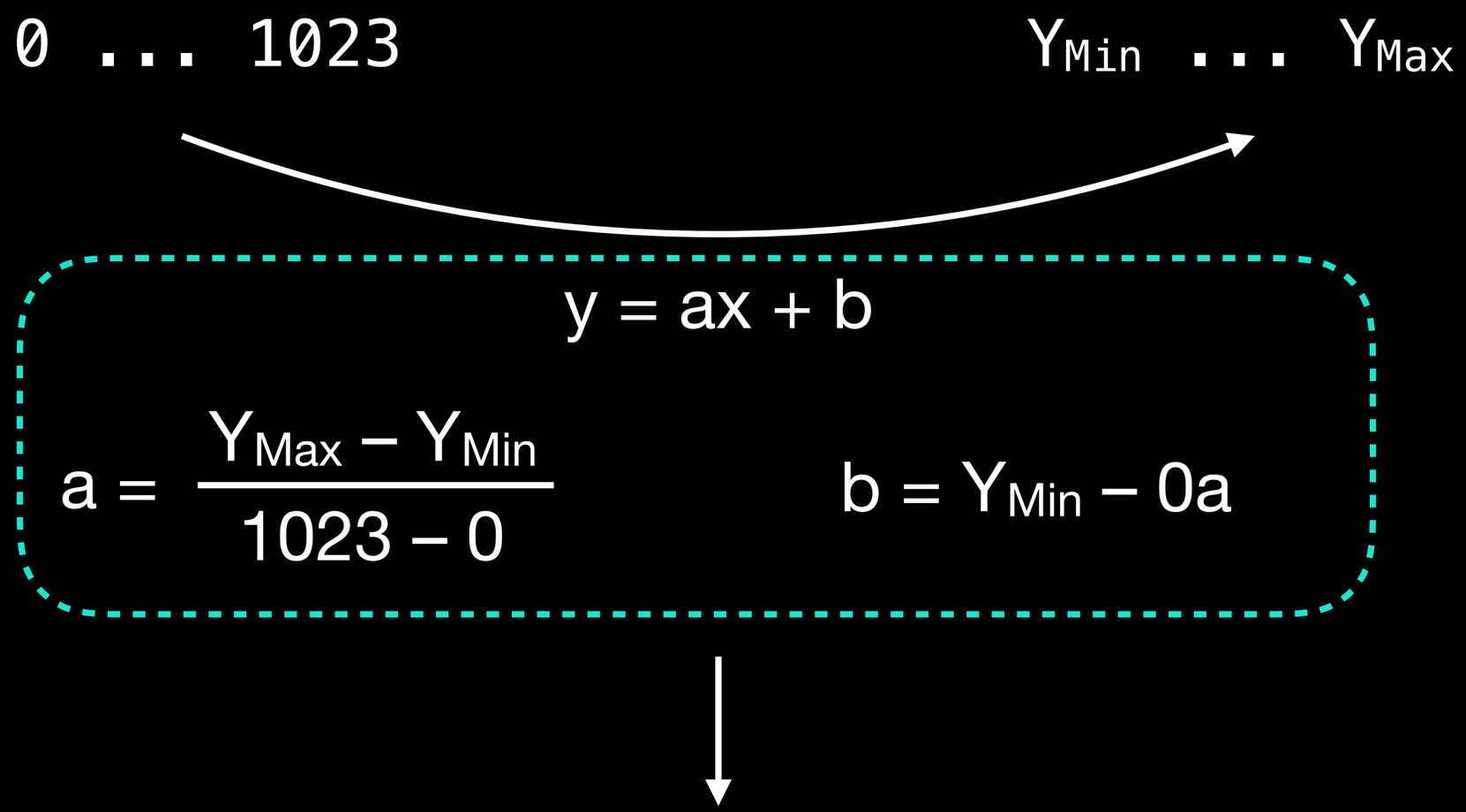
`analogRead(pino)`

```ino
int potenciometro = A5;
void setup () {
    Serial.begin(9600);
    pinMode(potenciometro, INPUT);
}

void loop () {
    int valorAnalogico = analogRead(potenciometro);
    Serial.println(valorAnalogico);
    delay(200);
}
```





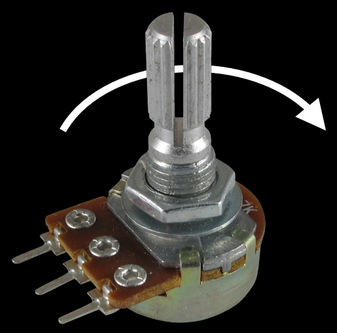
Leitura de Valores Analógicos

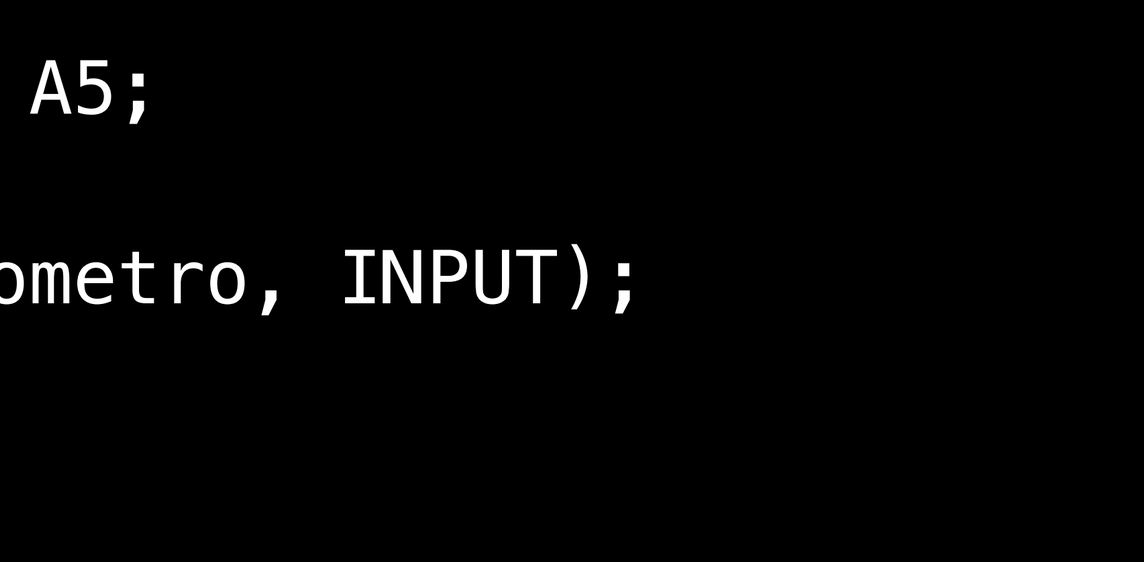



```
int y = map(valorLido, 0, 1023, yMinimo, yMaximo);
```

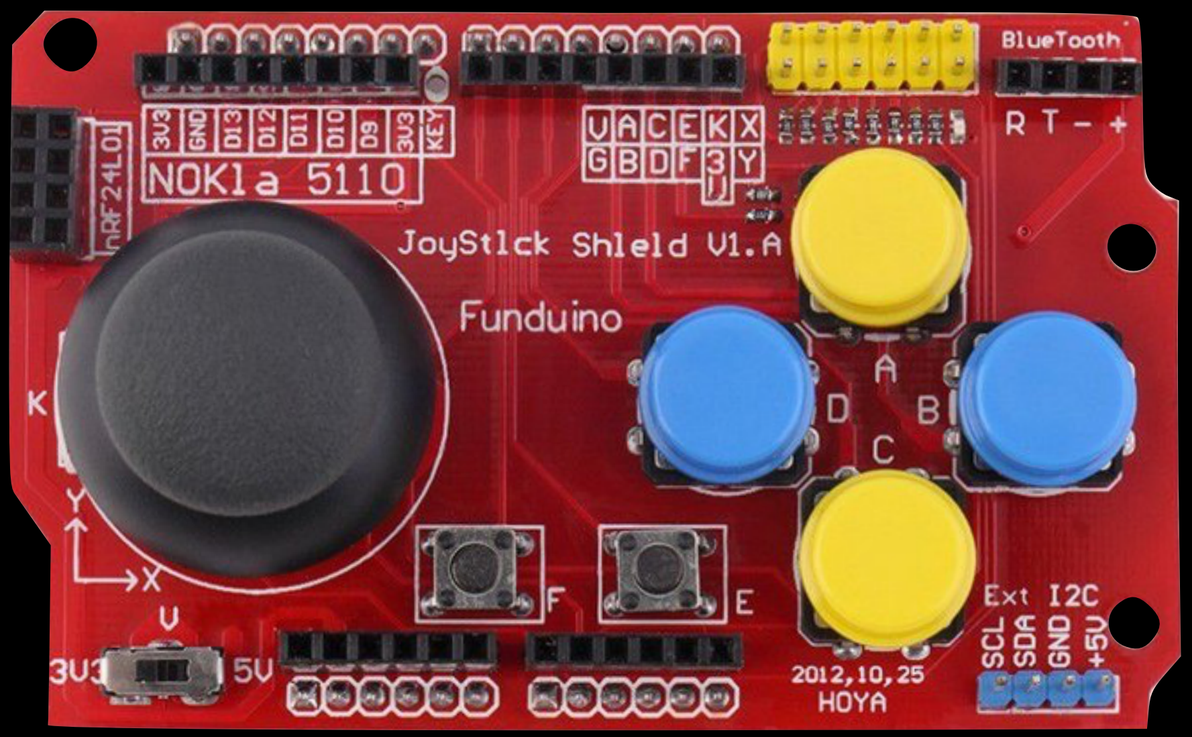
Mapeamento de Valores Analógicos

```ino
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

A1

A0

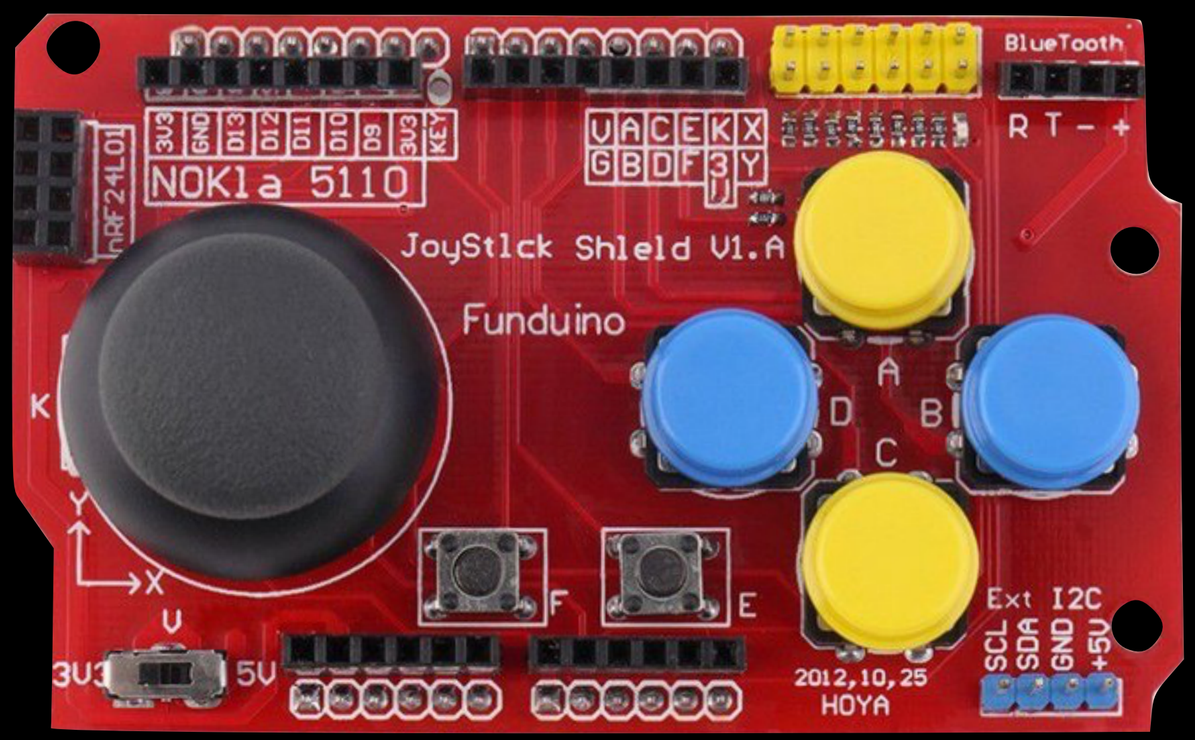
botões digitais

2

5

3

4

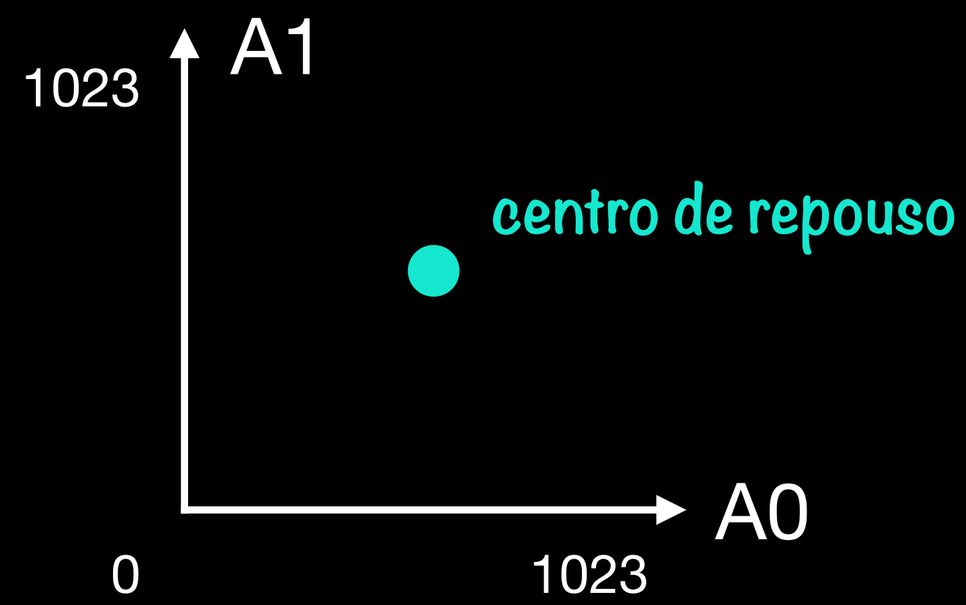


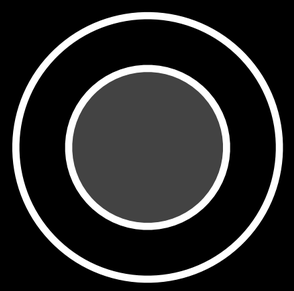
7

6

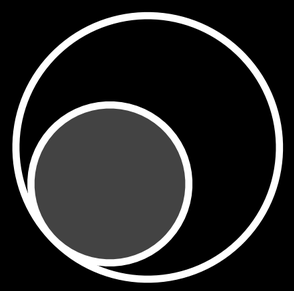
botões digitais

Pino Usados pelo Shield Joystick

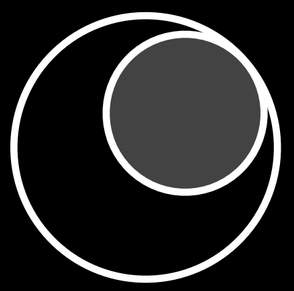




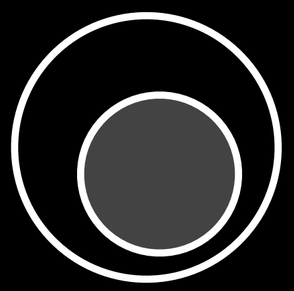
A0 = 512
A1 = 512



A0 = 0
A1 = 0



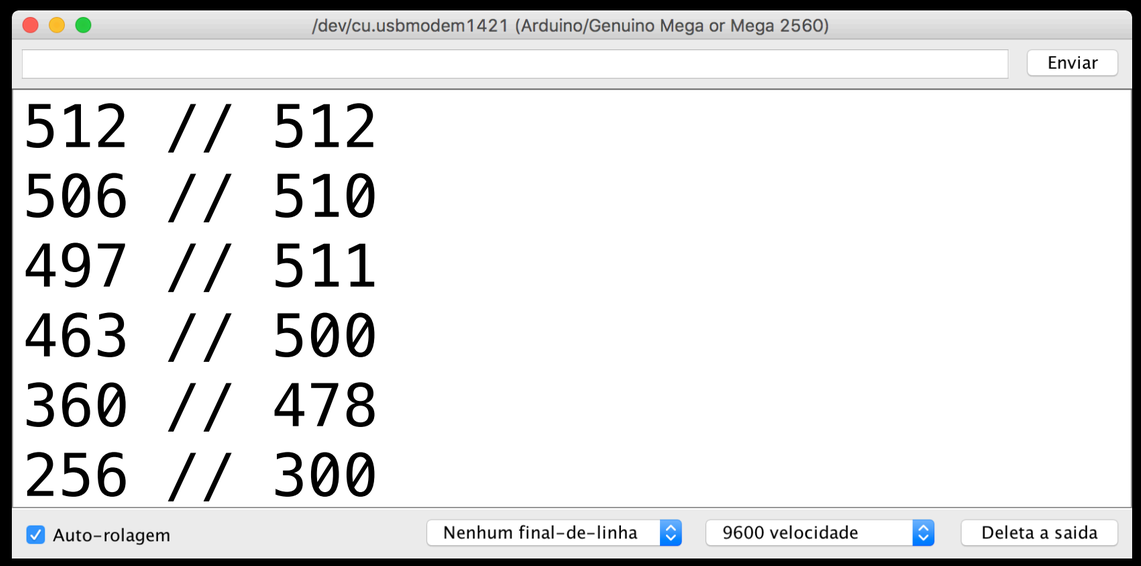
A0 = 1023
A1 = 1023



A0 = 745
A1 = 353

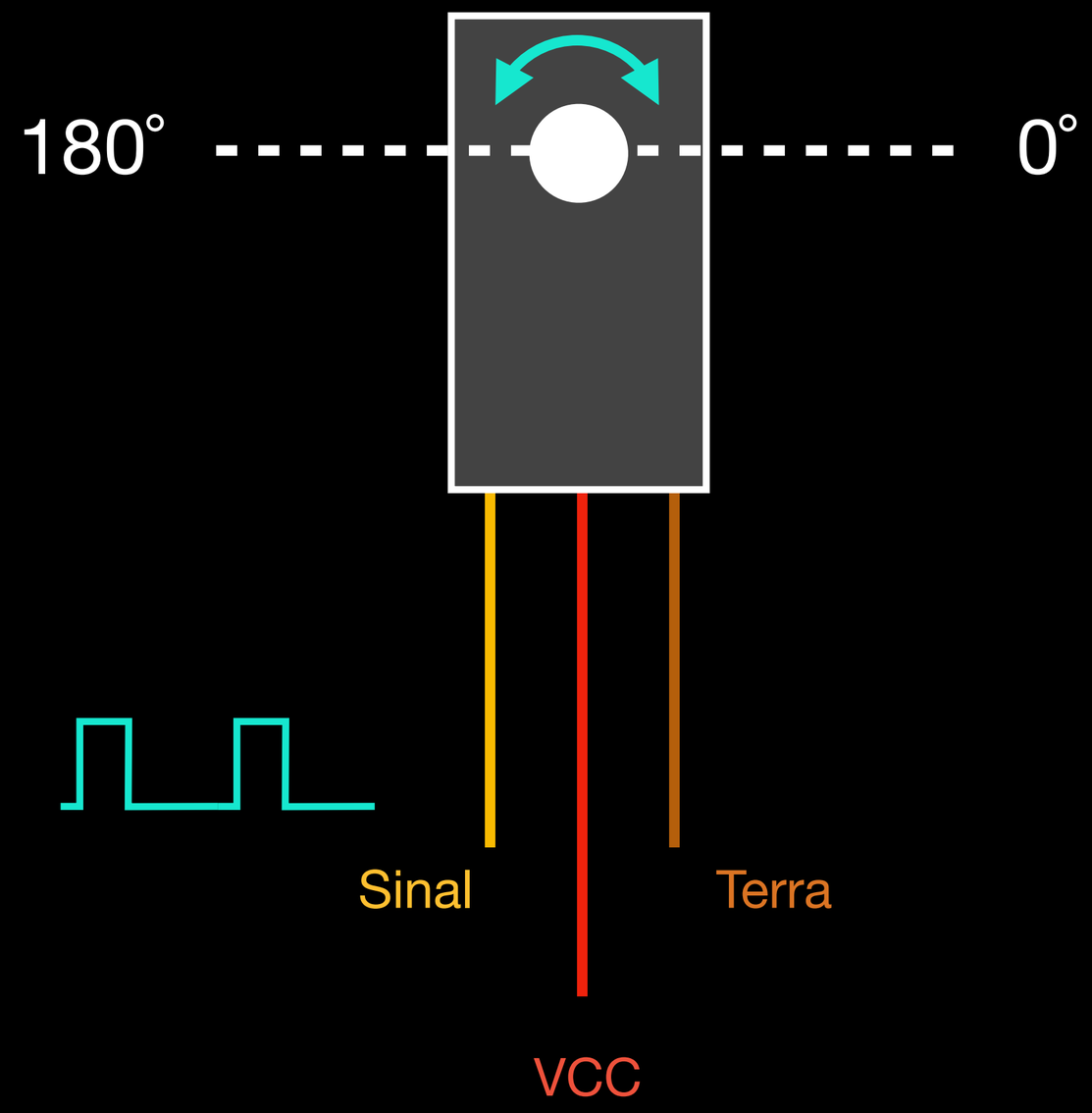
Controle Analógico de Dois Eixos

```
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);
}
```



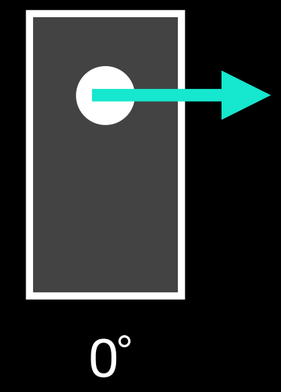
Leitura de Valores Analógicos

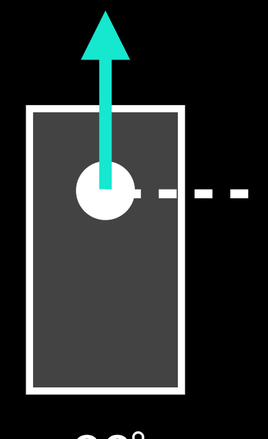
Servomotor

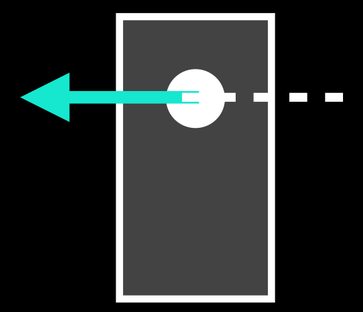


Controle de Giro do Servo

PWM (Pulse Width Modulation)



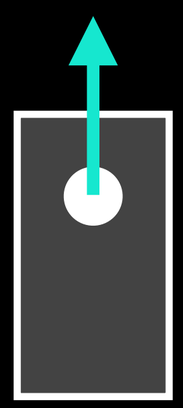


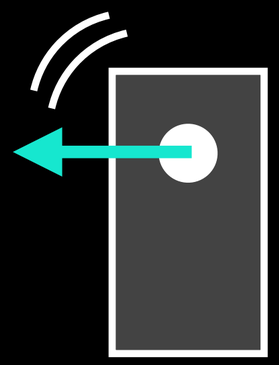


Controle do Ângulo do Servo via Entrada com Pulsos

```ino
#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)
}
```

Movimentação do Servo

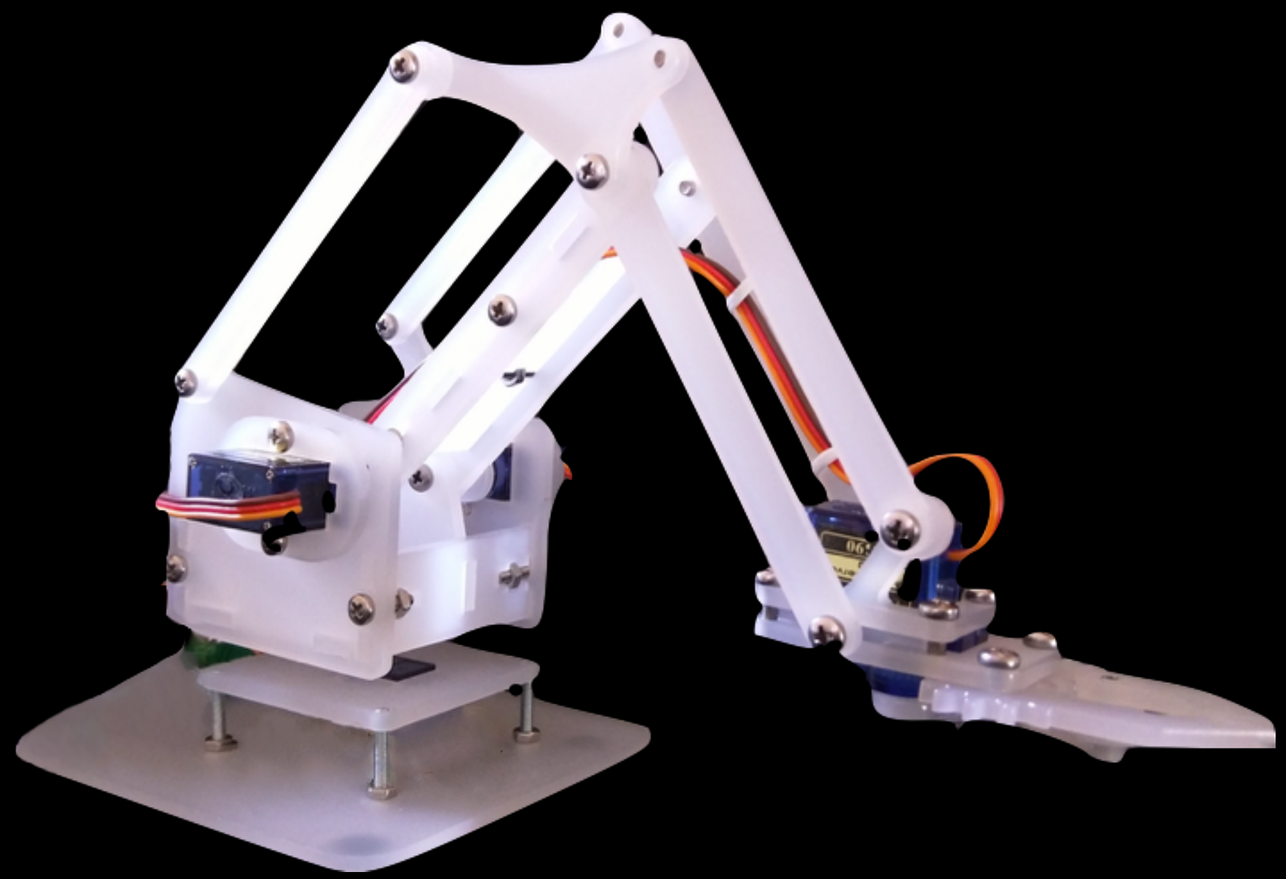




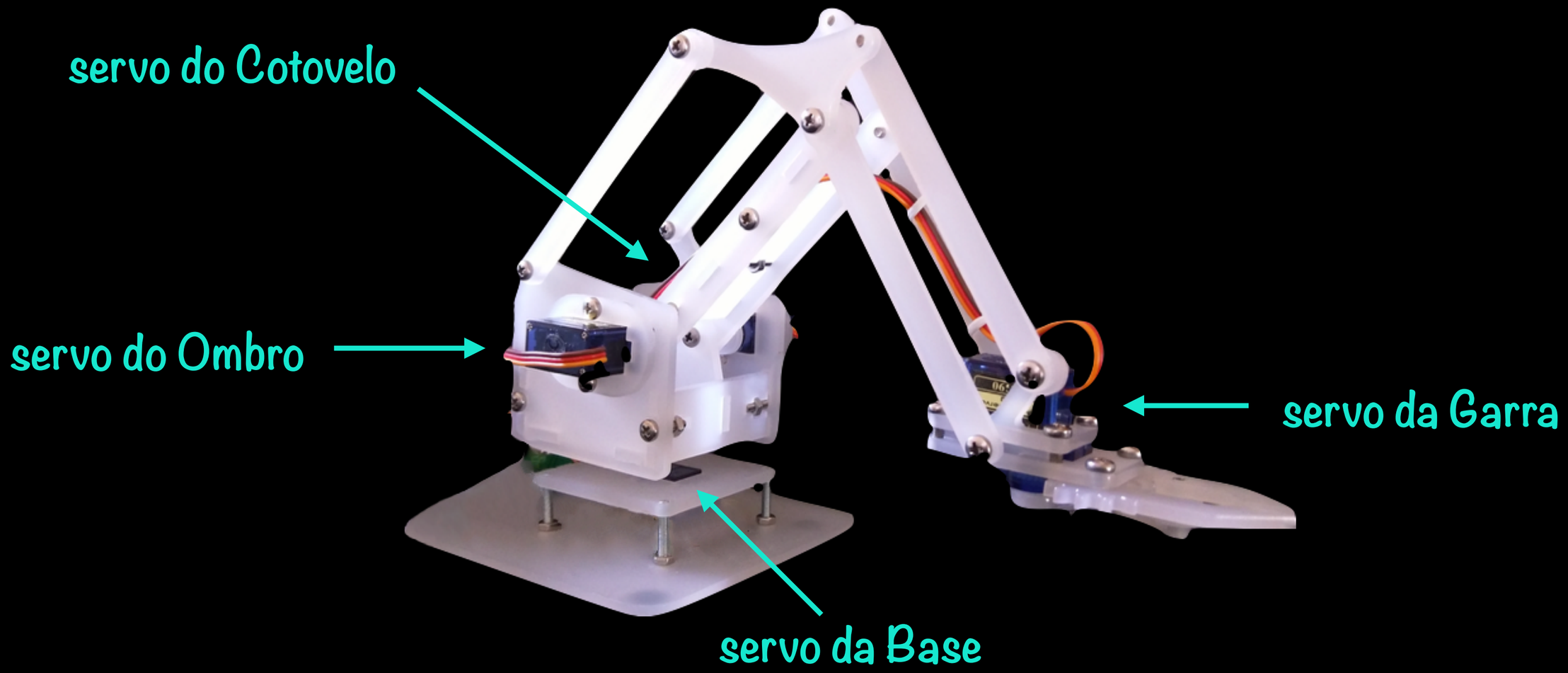
Problema de Movimentos Bruscos do Servo

```
#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);
    }
}
```

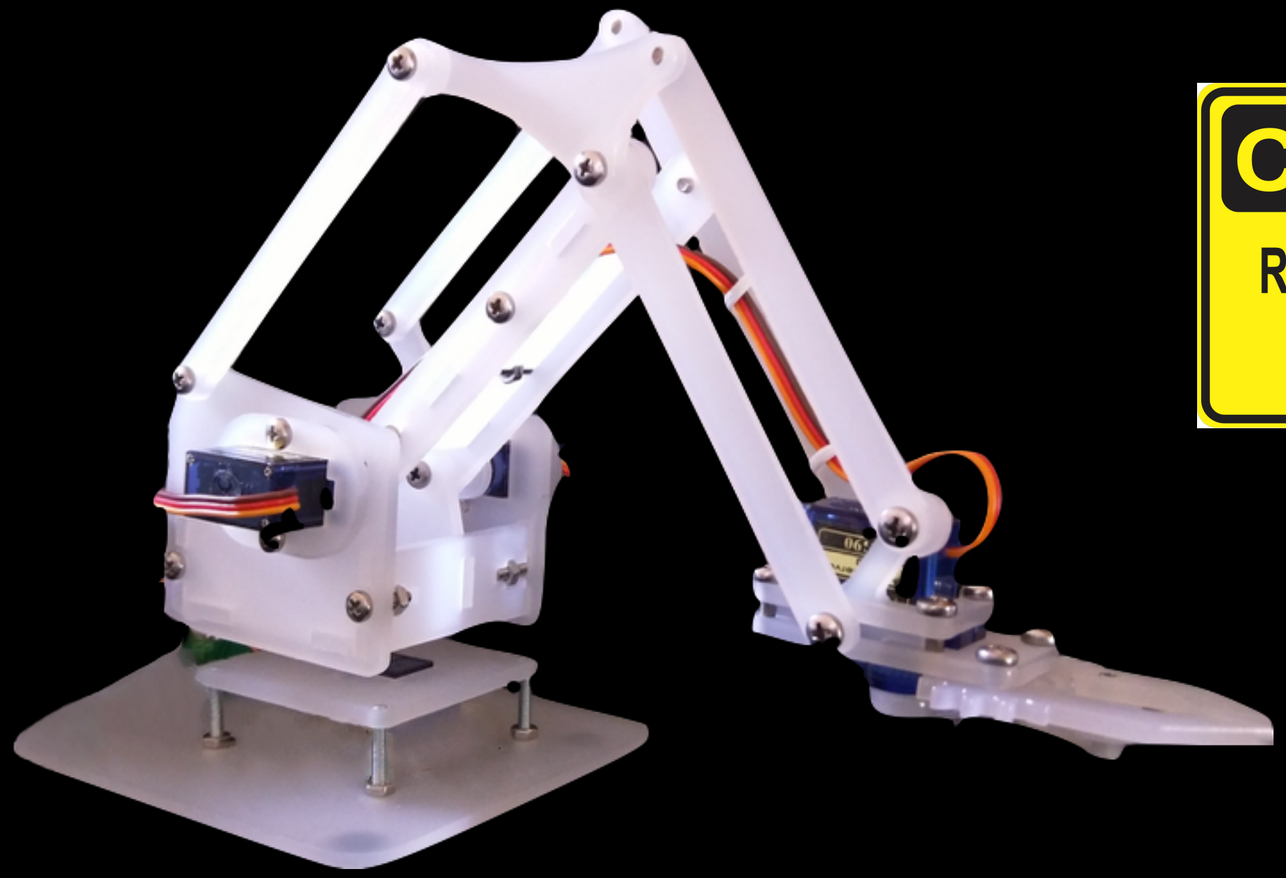
Movimentação Suave do Servo



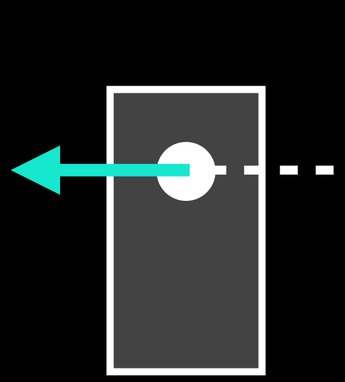
Braço Mecânico (meArm)



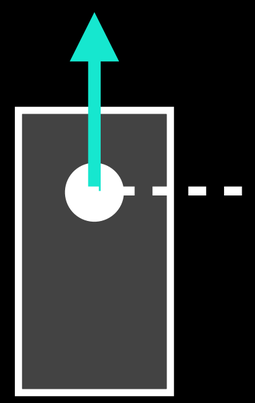
Servos do Braço Mecânico



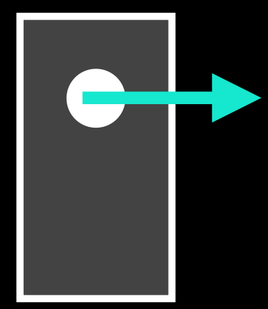
Fonte Variável para Alimentar os Servos



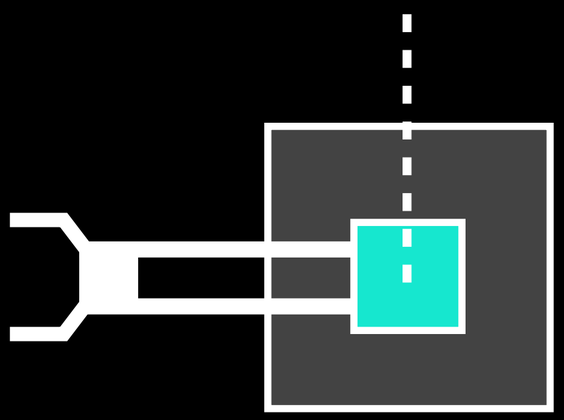
180°



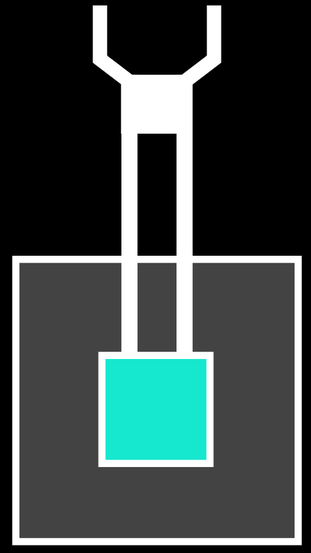
90°



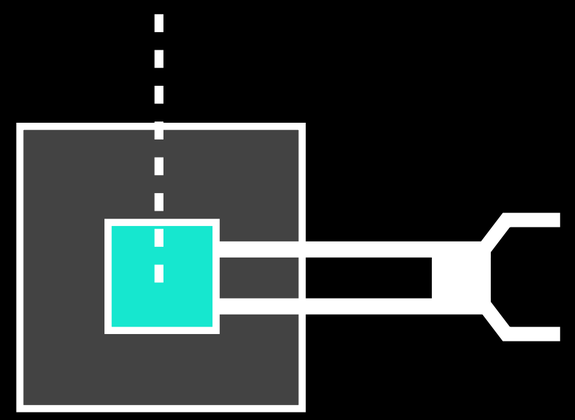
0°



-π/2 rad

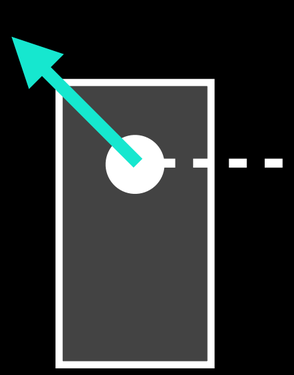


0 rad

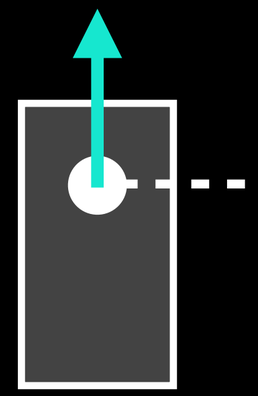


π/2 rad

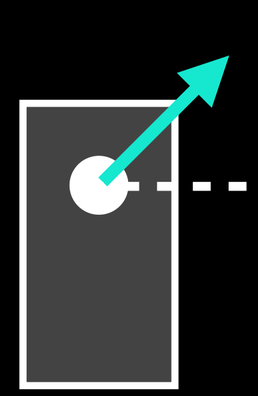
Controle do Ângulo da Base via Servo



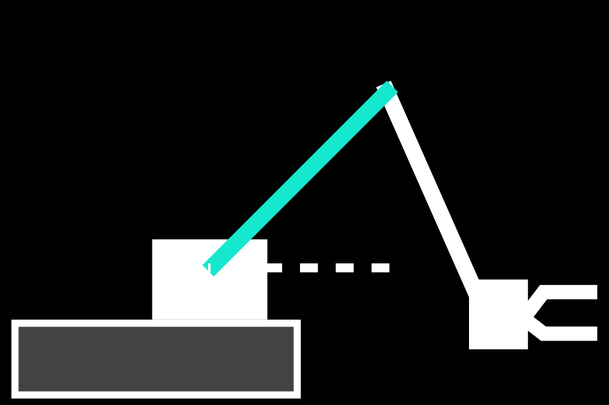
135°



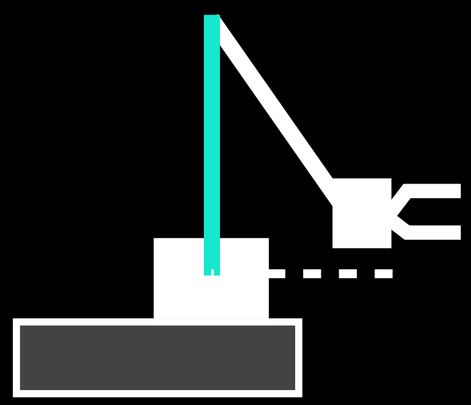
90°



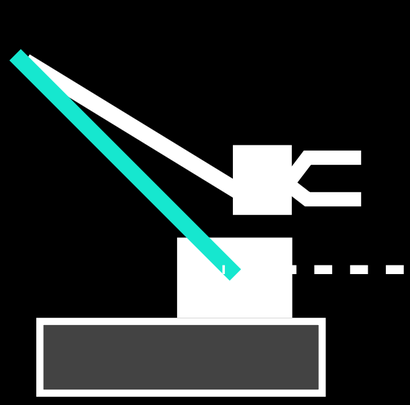
45°



π/4 rad

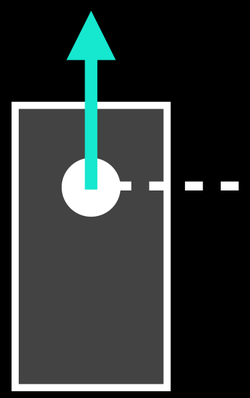


π/2 rad

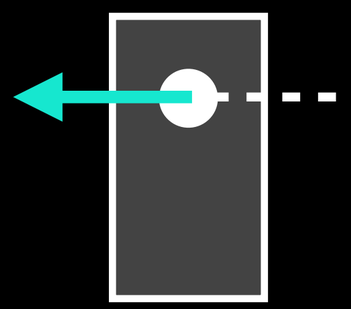


3π/4 rad

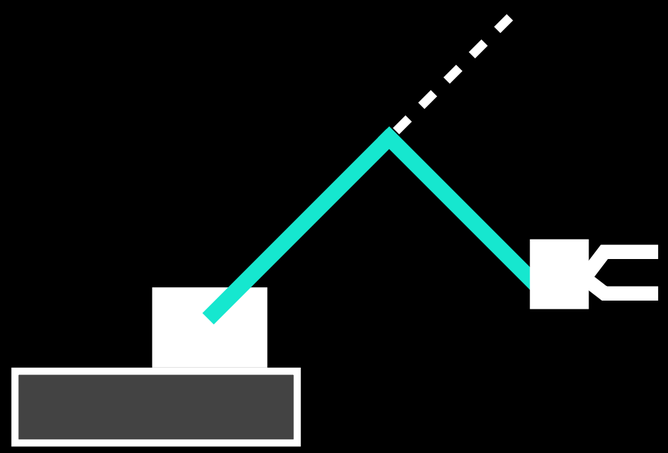
Controle do Ângulo do Ombro via Servo



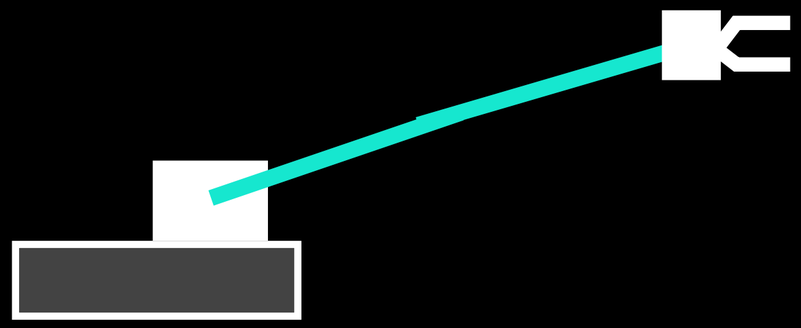
90°



180°

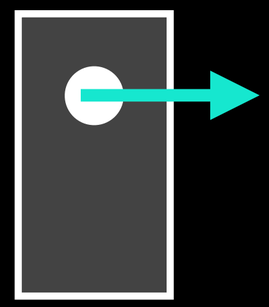


-π/2 rad

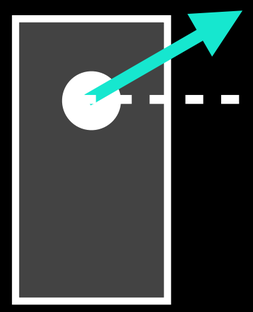


0rad

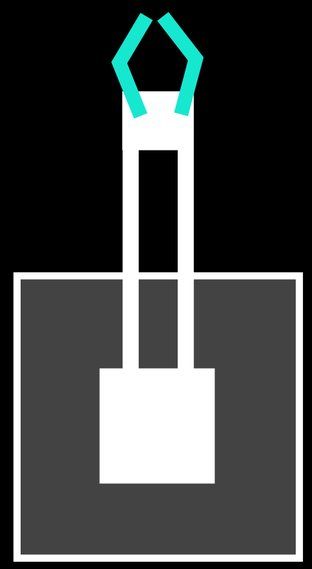
Controle do Ângulo do Ombro via Servo



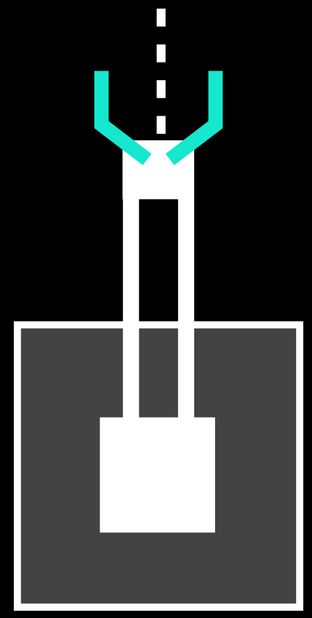
0°



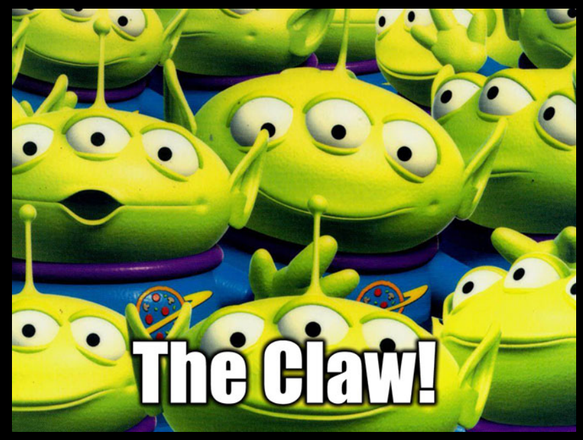
30°



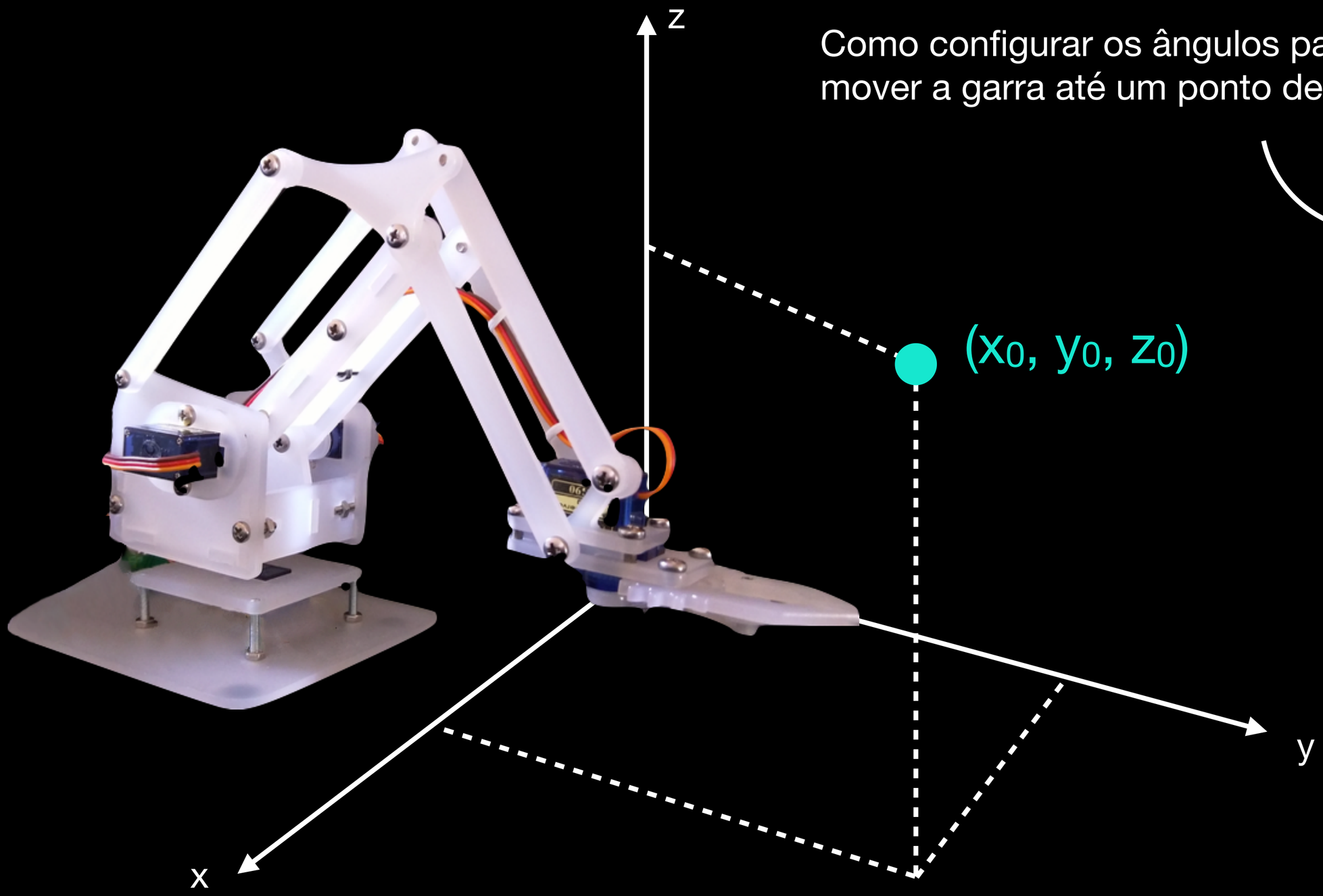
0 rad



π/2 rad



Controle do Ângulo da Garra via Servo

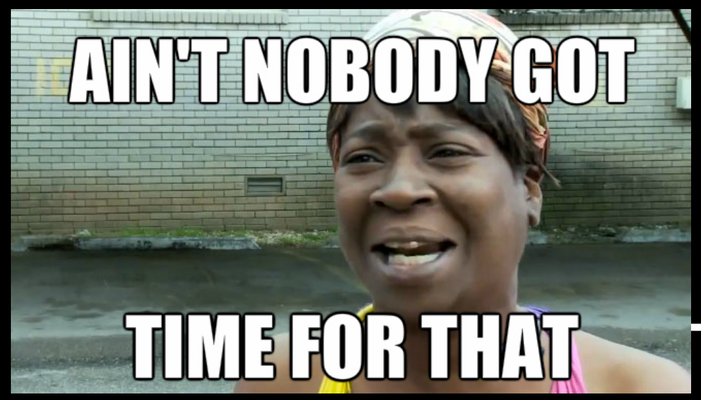


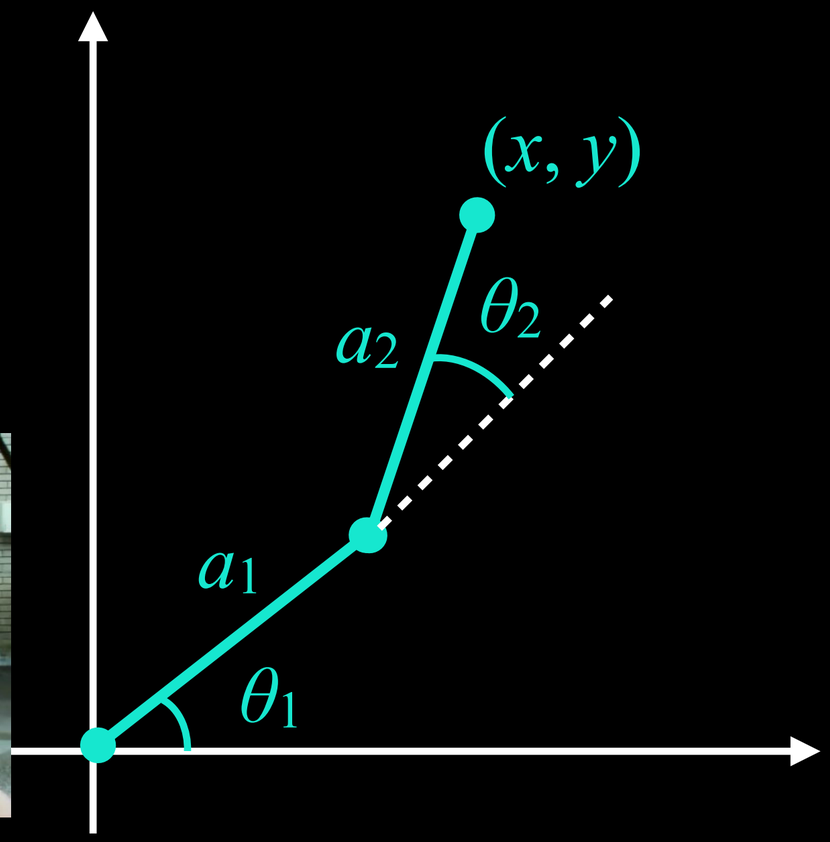
Como configurar os ângulos para
mover a garra até um ponto desejado?

Sistema de Coordenadas do Braço Mecânico

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

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

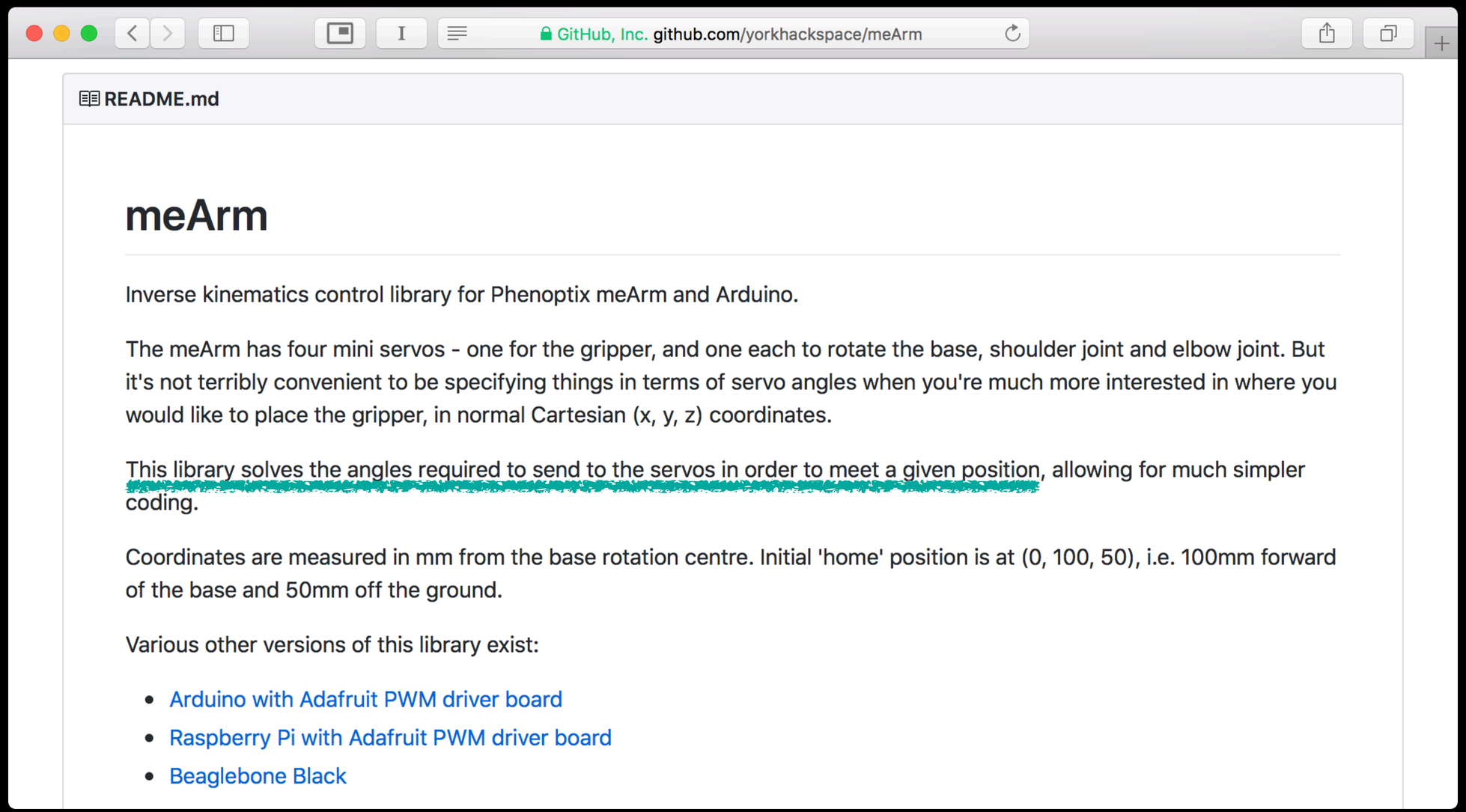




$$\tan\alpha = \frac{y}{x} \qquad \tan\beta = \frac{a_2\sin\theta_2}{a_1 + a_2\cos\theta_2}$$

$$\tan\theta_1 = \tan(\alpha - \beta) = \ldots = \frac{y(a_1 + a_2\cos\theta_2) - xa_2\sin\theta_2}{x(a_1 + a_2\cos\theta_2) + ya_2\sin\theta_2}$$

Cinemática Inversa



Biblioteca meArm para Arduino

```ino
#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();
}
```

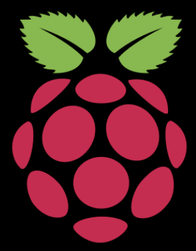
Movimentação do Braço

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

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

Outras Funções do Braço

"Software"



musicas.txt

não tem nem
sistema operacional

imagem.jpeg

ajustes.json

mas...

Armazenamento de Dados

www.arduino.cc/en/Reference/EEPROM

SIGN IN

Reference Language | Libraries | Comparison | Changes

EEPROM Library

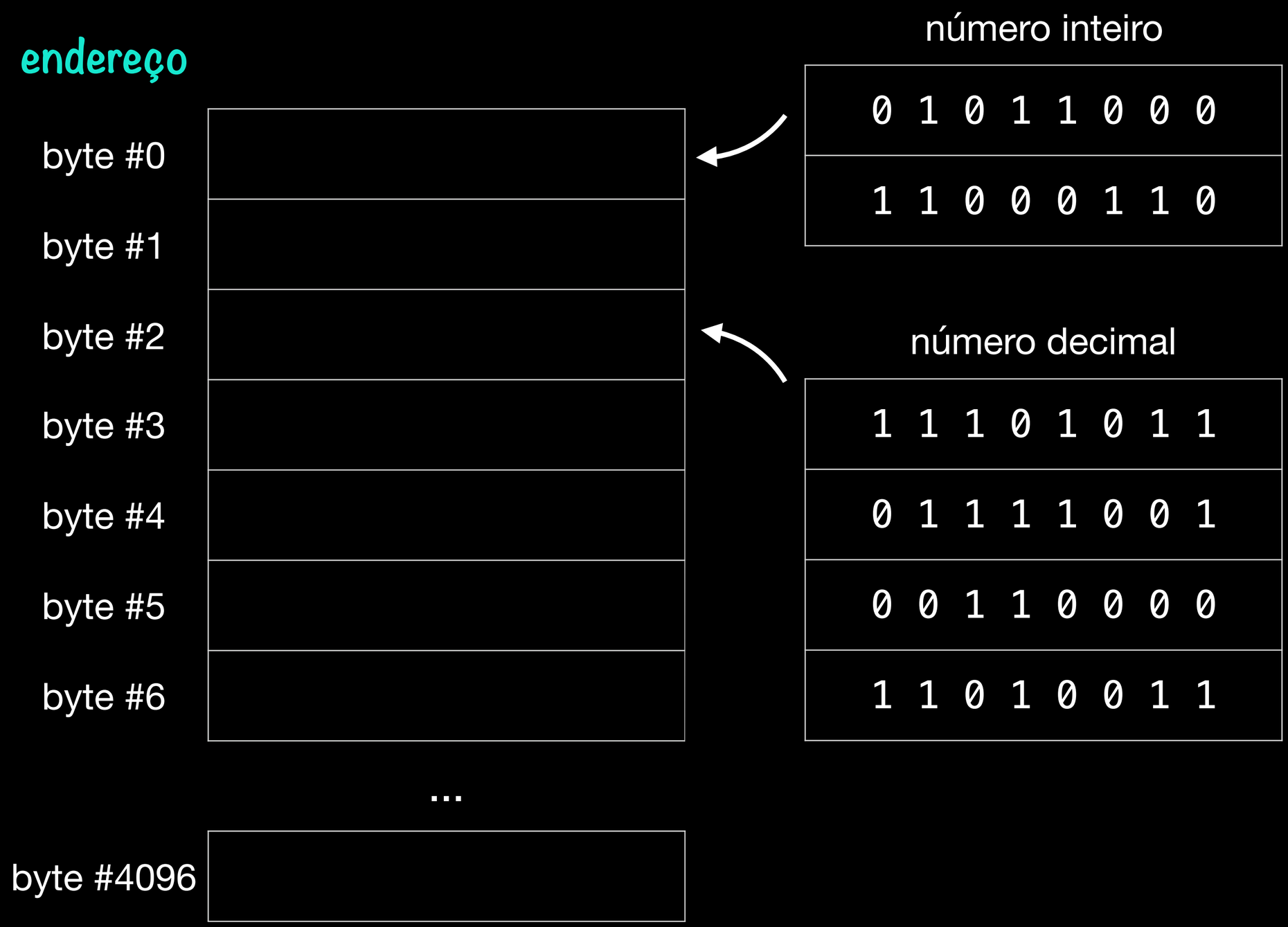
The microcontroller on the Arduino and Genuino AVR based board has EEPROM: memory whose values are kept when the board is turned off (like a tiny hard drive). This library enables you to read and write those bytes.

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[]

Biblioteca de Acesso à Memória EEPROM



Funcionamento Interno da Memória EEPROM

```ino
#include <EEPROM.h>

int valorInteiro;
int endereco = 0;

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

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

Leitura e Escrita de uma Variável na EEPROM

```ino
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);
```

```ino
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);
```

Leitura e Escrita de Diferentes Variáveis na EEPROM

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 Opera

Funcionalidade	Comandos

Leitura Analógica
acessar documentação
	`int potenciometro = A5;`
	`pinMode(potenciometro, INPUT);`
	`int valorAnalogico = analogRead(potenciometro);`
	`int valorMapeado = map(valorAnalogico, 0, 1023, min, max);`

Servomotor
acessar documentação
	`#include <Servo.h>`
	`Servo servo;`
	`servo.attach(pino); servo.detach();`
	`servo.write(anguloEmGraus);`

Braço Mecânico
acessar documentação
	`#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();`

EEPROM
acessar documentação
	`#include <EEPROM.h>`
	`EEPROM.get(endereco, minhaVariavel);`
	`EEPROM.put(endereco, minhaVariavel);`

Resumo para o Projeto 08

Funcionalidade	Comandos

Campainha Passiva
acessar documentação
	`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();`

Resumo para o Projeto 07

Funcionalidade	Comandos

Revisão de C++	`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; }`

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

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

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

ShiftDisplay
acessar documentação
	`#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);`

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

Resumo para o Projeto 06