NIVELAMENTO DE ROBÓTICA

DIA - 3

COMBU MAKER

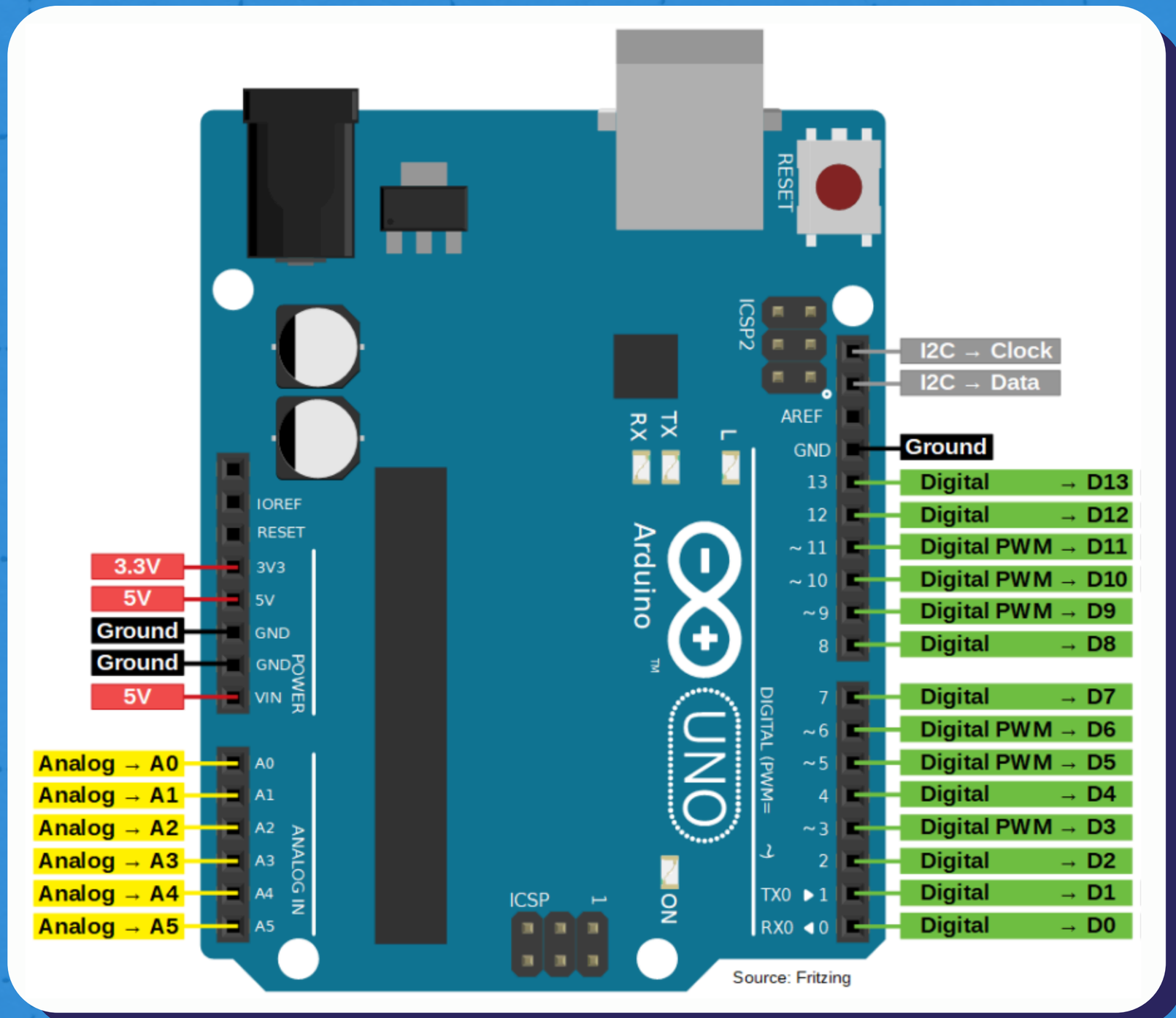
Siga nosso Instagram

@combumaker

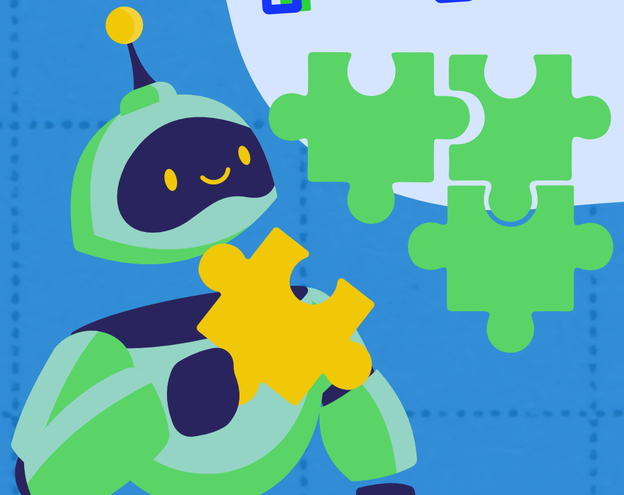
COMBU MAKER

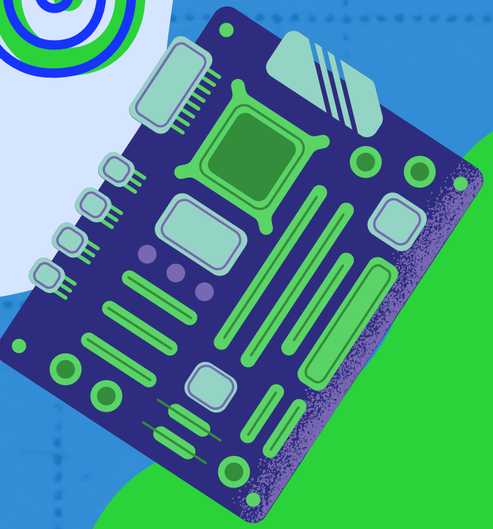
ARDUINO:

COMBU MAKER



COMPARAÇÃO: PYTHON E ARDUINO





COMBU MAKER

PYTHON

- É uma linguagem de programação conhecida por sua sintaxe simples e legibilidade;
- É uma **linguagem de tipagem dinâmica**, o que significa que a sintaxe não exige que o tipo de dado seja informado de forma explícita;
- Possui uma maior flexibilidade e é mais rápida para desenvolver um algoritmo, porém é mais demorada para a execução do código.
- As variáveis podem ser reatribuídas para armazenar diferentes tipos de dados.

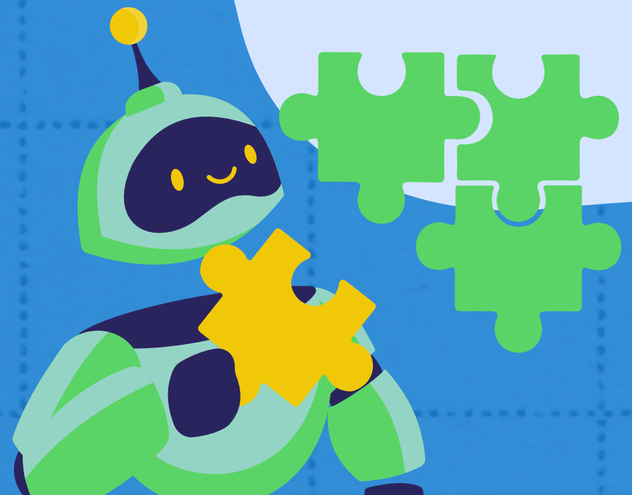
COMBU MAKER

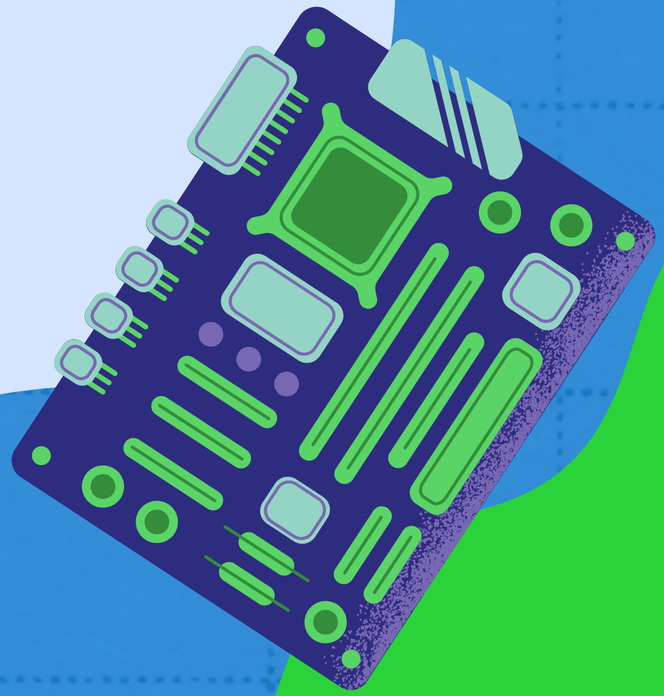
ARDUINO

- Utiliza uma sintaxe semelhante à linguagem C/C++, com estruturas de controle de fluxo familiares, como loops, condicionais e funções;
- É uma **linguagem de tipagem estática**, ou seja o programador precisa explicitar o tipo de dado usado no código;
- Possui um melhor performance durante a execução do código, mas pode ser mais demorada e complexa para desenvolver um algorítimo.

COMBU MAKER

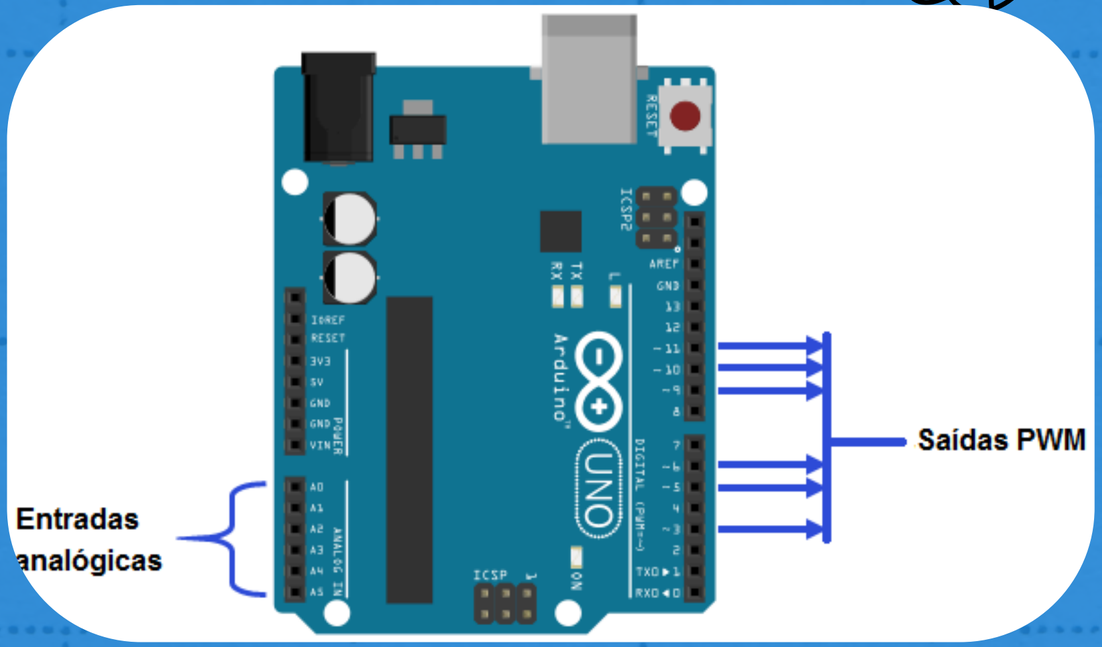
SINAL PWM





COMBU MAKER

SINAL PWM



O que é?

- Pode ser traduzida como "Modulação por Largura de Pulso".

- No Arduino, você pode usar saídas PWM em alguns dos pinos digitais para gerar sinais PWM.

- Permite um controle preciso da potência aplicada a uma carga, minimizando o desperdício de energia.

COMBU MAKER

ANALOG WRITE()

- A função **analogWrite()** é usada para controlar a frequência do pulso de energia que vai para um pino digital;

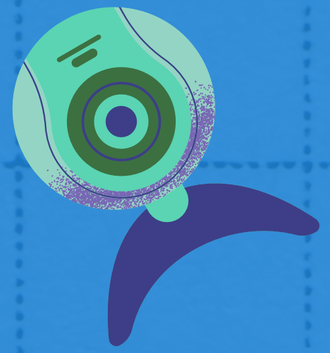
- A função **analogWrite()** funciona criando um sinal PWM no pino digital especificado;

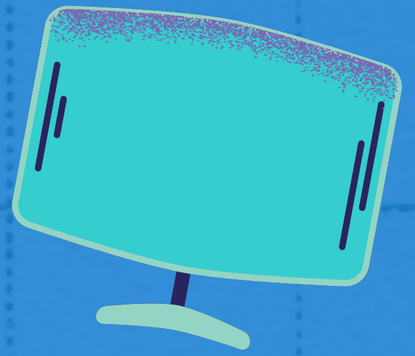
- O ciclo de trabalho é especificado como um valor entre 0 e 255, sendo 0 o sinal está sempre desligado, enquanto um valor de 255 significa que o sinal está sempre ligado na frequência máxima;

- Valores intermediários definem o ciclo de trabalho proporcionalmente.

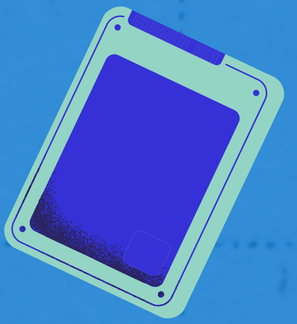
COMBU MAKER

EXEMPLO



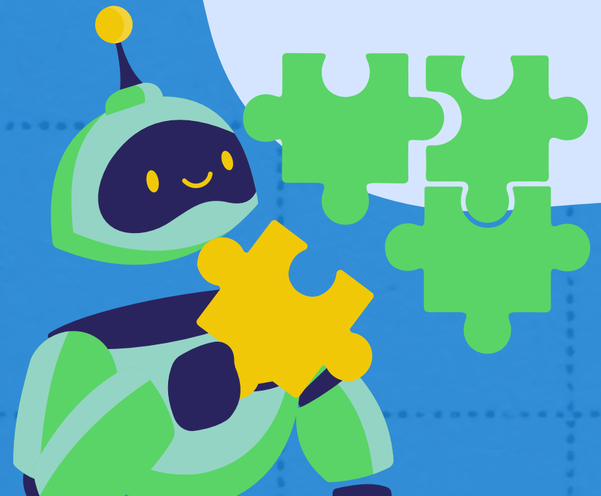


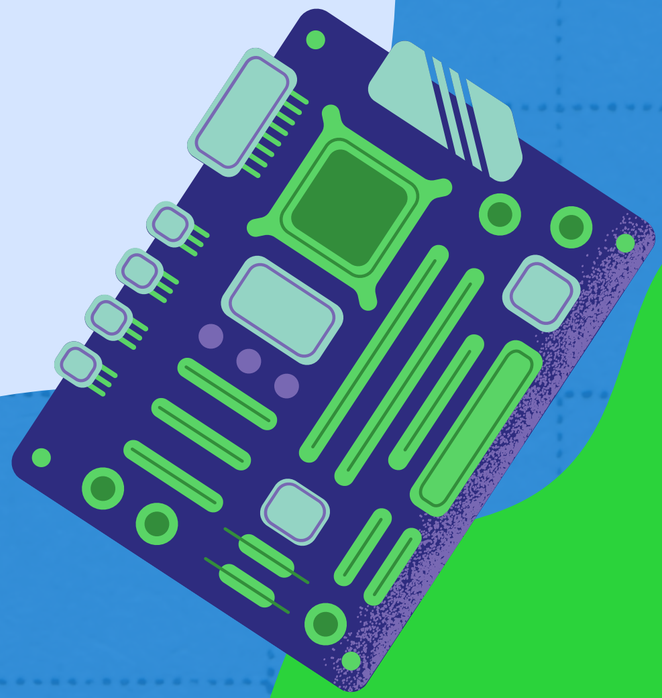
```
1  analogWrite(ledPin, 132);
2  delay(25);
```



COMBU MAKER

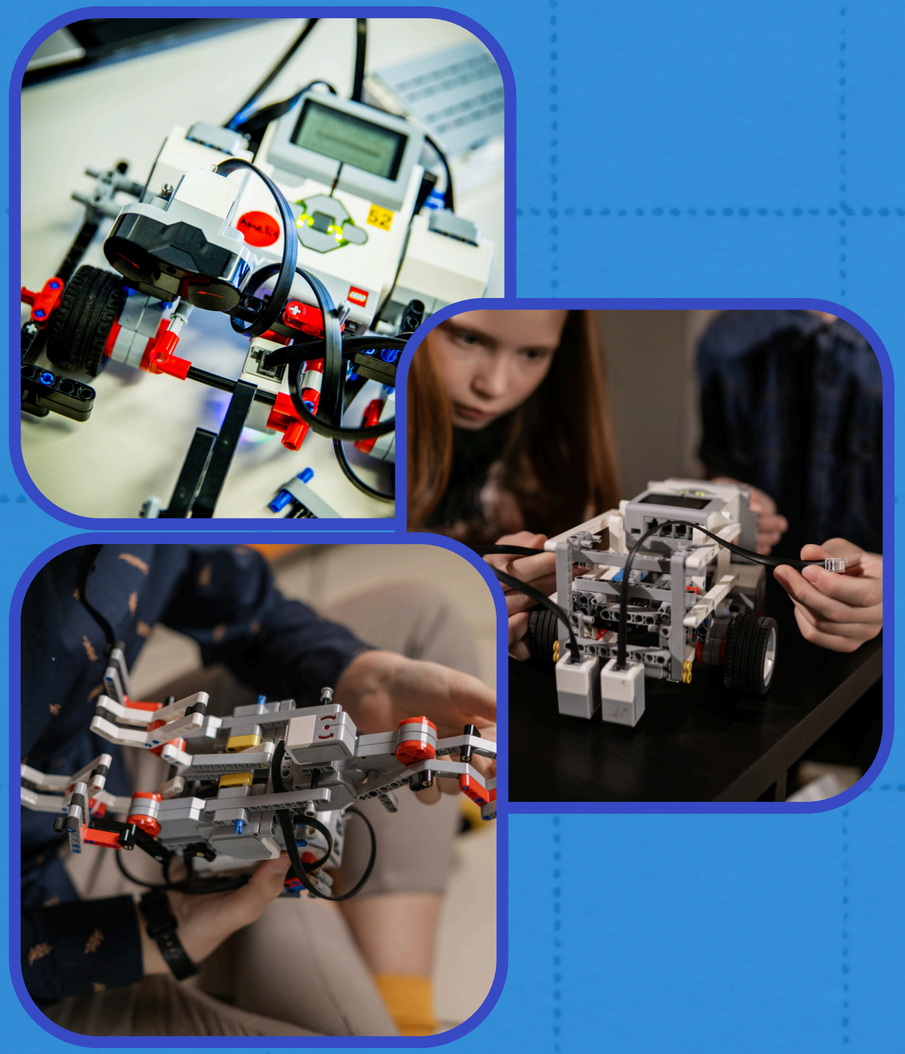
BUZZER





COMBU MAKER

BUZZER



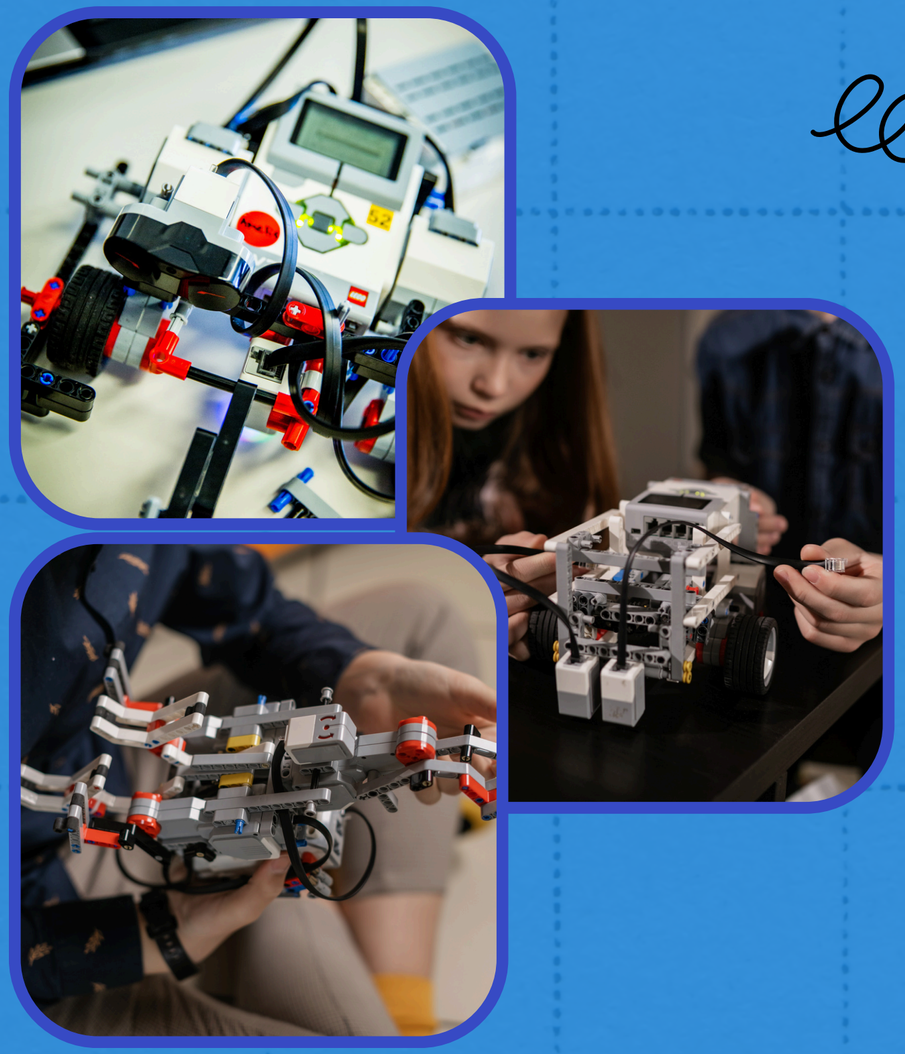
O que é?

- Buzzer é um dispositivo utilizado em eletrônica para produzir som ou emitir um sinal sonoro.

- Para usar um buzzer basta atribuí-lo à uma porta PWM (as que possuem sinal “~”), junto com um sinal GND para o aterramento.

COMBU MAKER

BUZZER

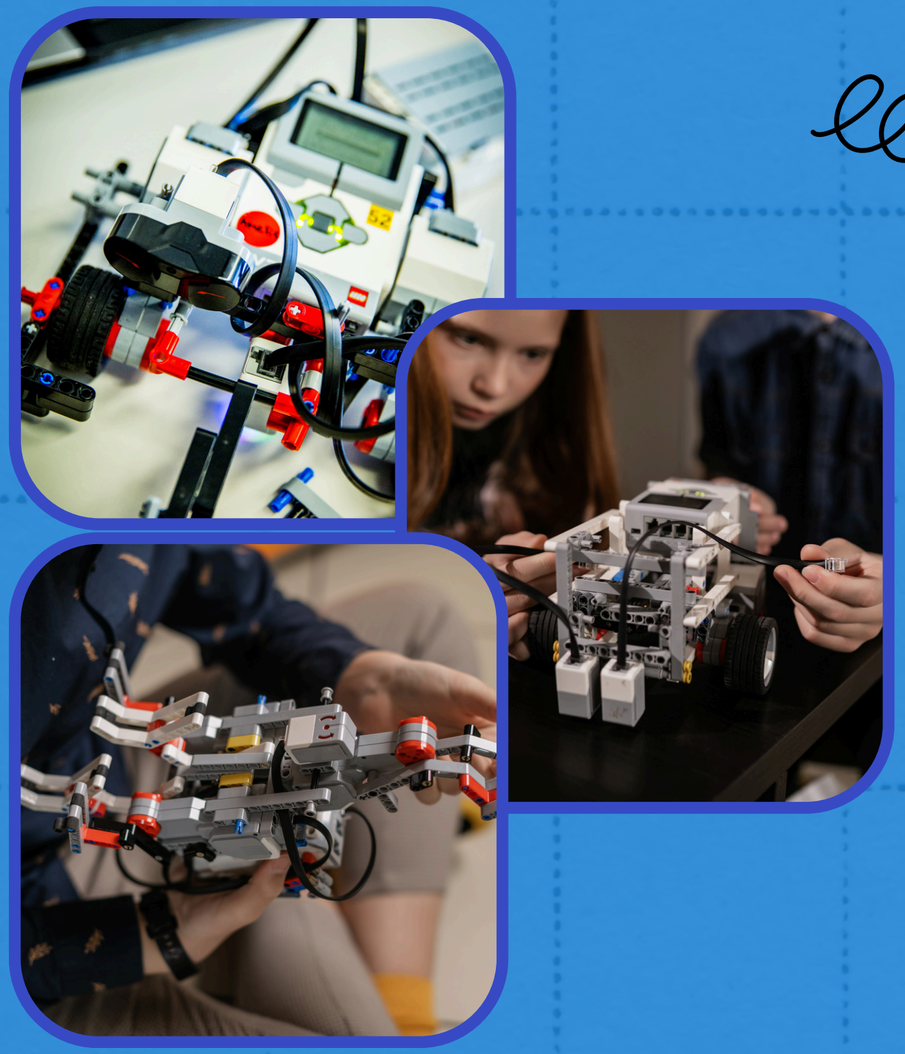


Programação:

- Diferentemente dos LEDs, que possuem maneiras diferentes de funcionamento diferentes dependendo da porta que é colocado, os buzzers que usaremos não precisam fazer o uso da função **analogWrite()**, pois só funcionam com sinais PWM, e possui duas funções principais.

COMBU MAKER

BUZZER

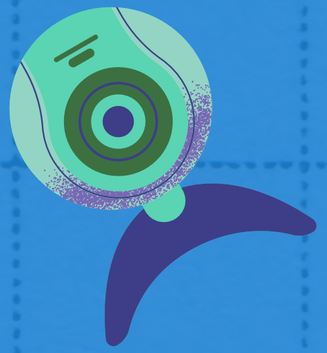


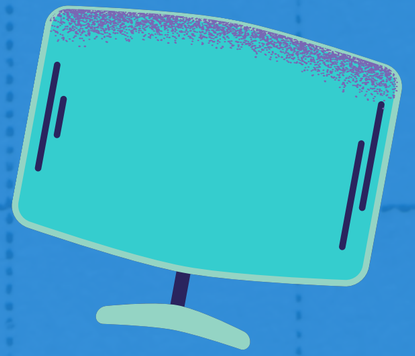
tone():
- É a função responsável por atribuir o pin que se contra o buzzer e a frequência do som que ele irá emitir.

noTone():
Essa função recebe somente um argumento que é o pin do buzzer e éresponsável por parar o som que ele está emitindo .

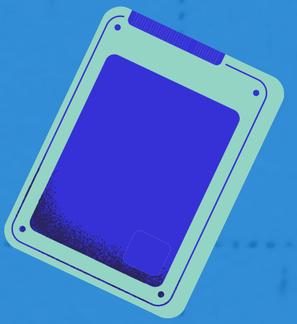
COMBU MAKER

EXEMPLO



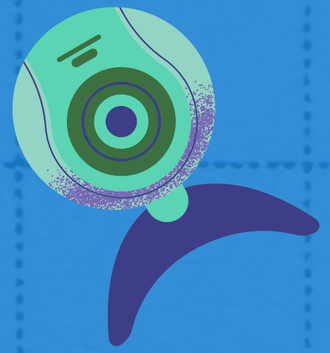


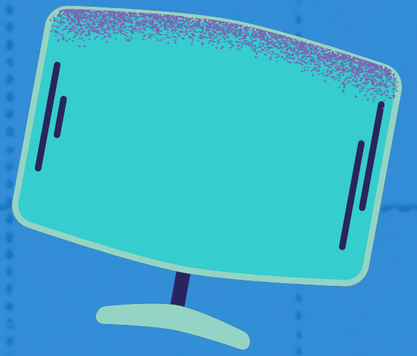
```
1  tone(buzzerPin, 1000);
2  delay(1000);
```



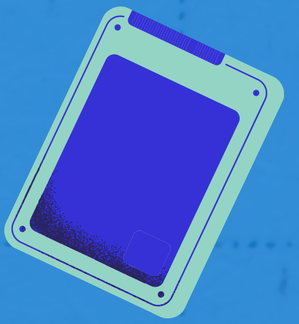
COMBU MAKER

EXEMPLO



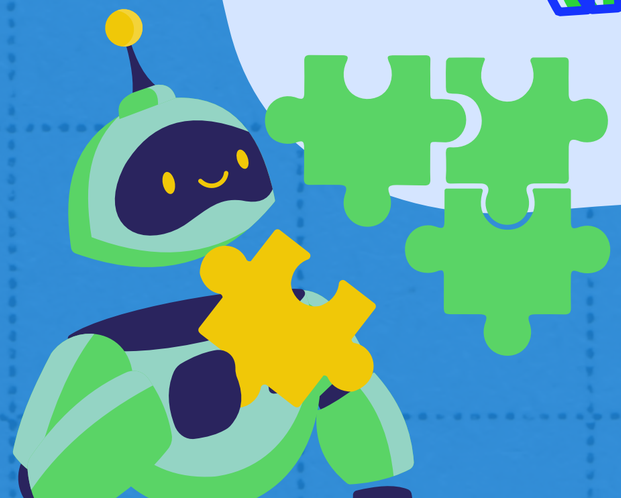


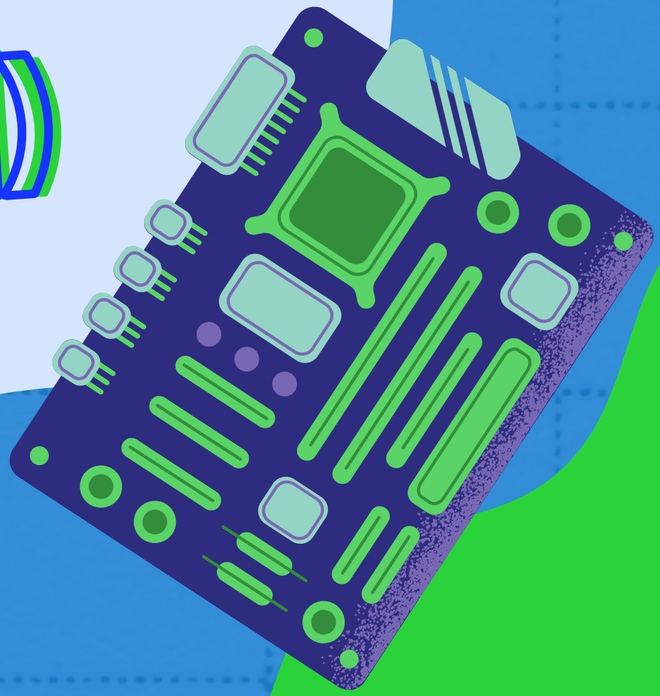
```
1  noTone(buzzerPin);
2  delay(1000);
```



COMBU MAKER

BOTÃO (PUSH BUTTON)





COMBU MAKER

BOTÃO(PUSH BUTTON)

O que é:

- É um dos componentes eletrônicos mais utilizados para prototipagem de projetos.

- Esta chave é um tipo de interruptor pulsador (conduz somente quando está pressionado).

COMBU MAKER

BOTÃO(PUSH BUTTON)

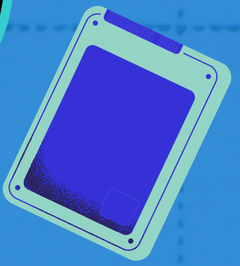
Pull UP e Pull DOWN:

- São usados em eletrônica para garantir que um circuito funcione corretamente e evite problemas indesejados.

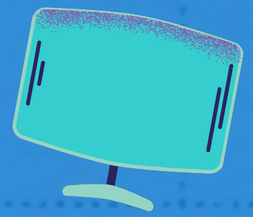
- Ajudam a eliminar “ruídos elétricos” que podem causar leituras erradas no sistema.

COMBU MAKER

BOTÃO(PUSH BUTTON)

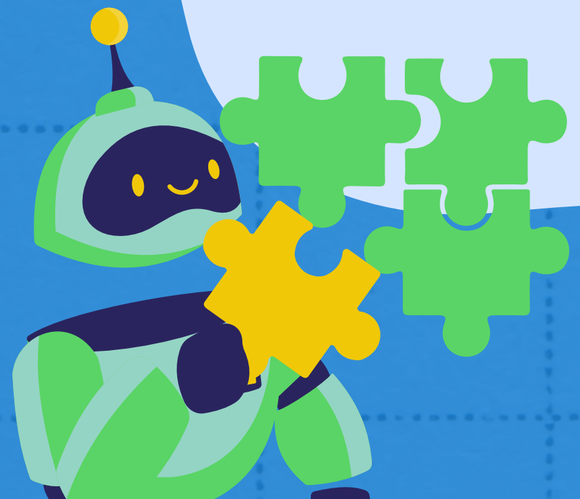


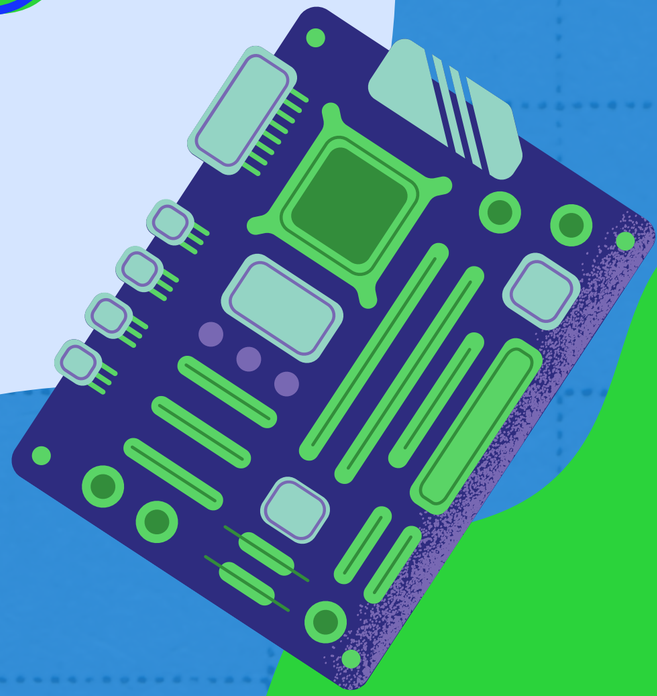
```
1  void setup() {
2    pinMode(buttonPin, INPUT_PULLUP);
3    pinMode(ledPin, OUTPUT);
4  }
```



COMBU MAKER

LER ESTADO DO BOTÃO





COMBU MAKER

DIGITAL READ

É necessário fazer o uso da da instrução digital read para ler o estado do botão e assim programar o botão de acordo com os seus estados, HIGH ou LOW

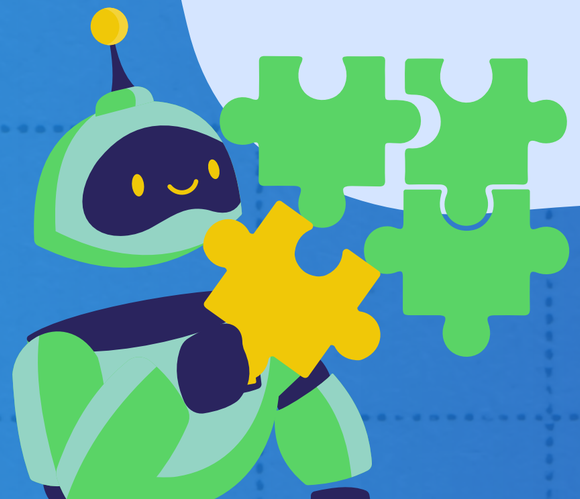
COMBU MAKER

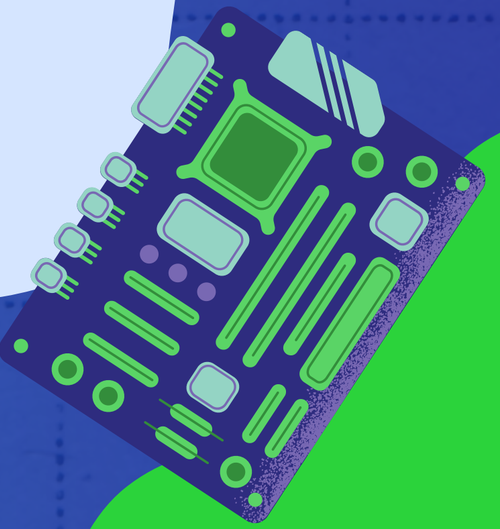
EXEMPLO

```
1
2  void loop() {
3    int buttonState = digitalRead(buttonPin);
4  }
```

COMBU MAKER

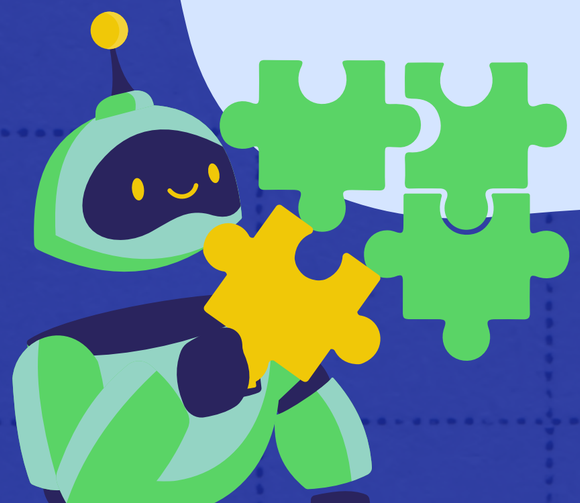
AULA 3

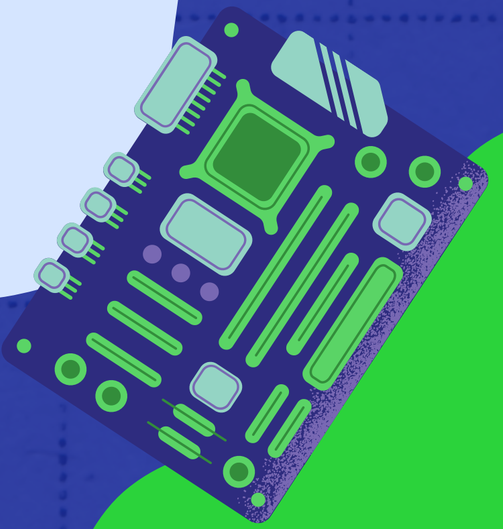




COMBU MAKER

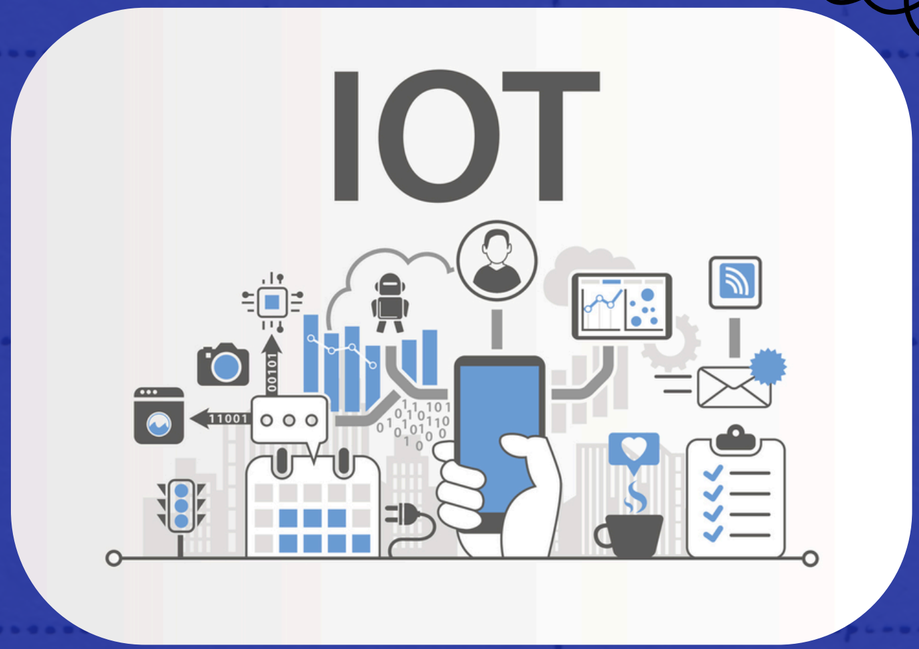
IOT





COMBU MAKER

IOT

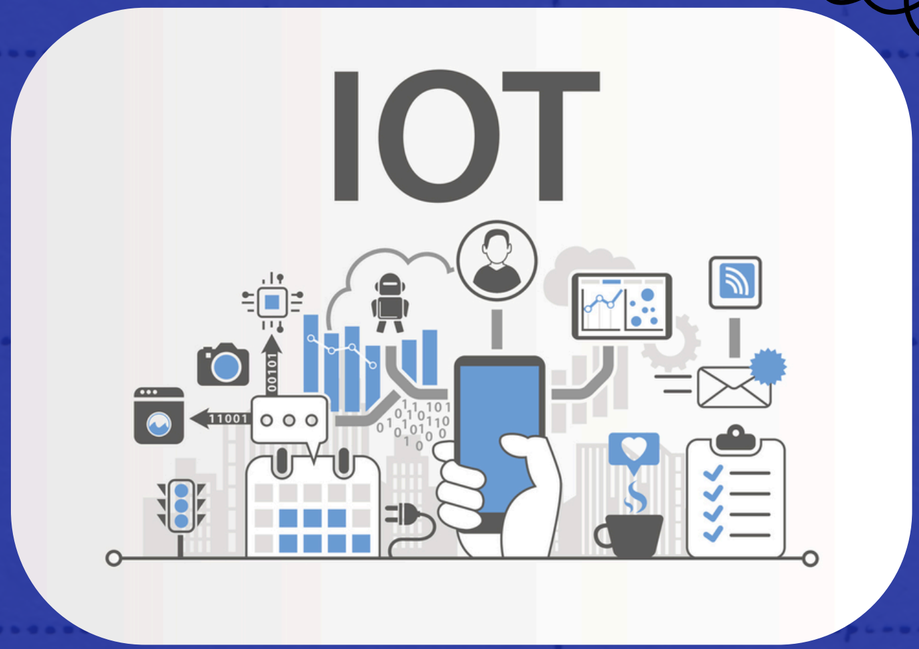


O que é?

A Internet das Coisas (IoT) é uma rede de objetos físicos, como eletrodomésticos, carros, sensores e outros dispositivos, que são conectados à internet e podem se comunicar e trocar dados entre si e com sistemas externos.

COMBU MAKER

IOT

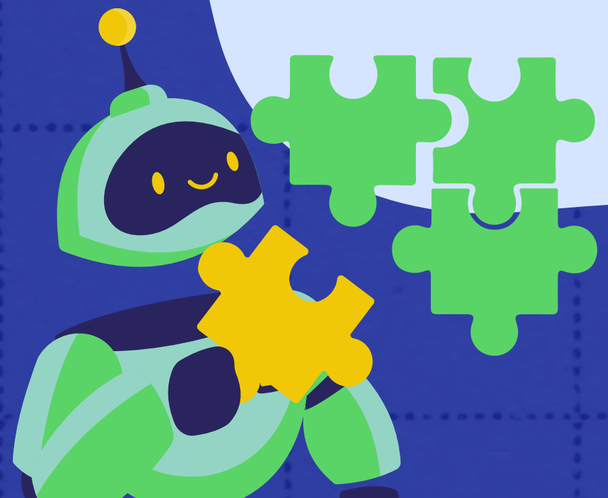


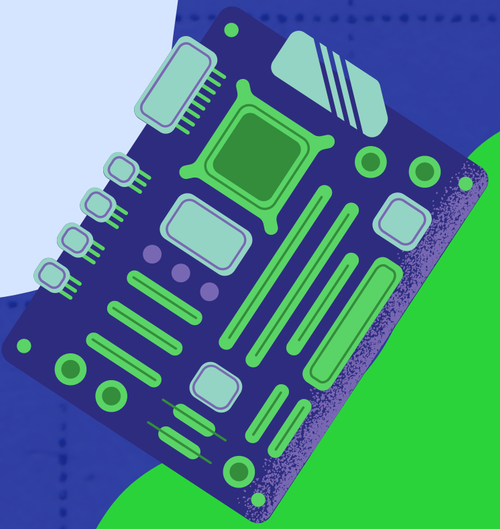
Principais características:

- Conectividade: Os dispositivos IoT se conectam à internet através de redes Wi-Fi, Bluetooth, Ethernet ou celular.
- Comunicação: Os dispositivos IoT podem se comunicar entre si e com sistemas externos para compartilhar dados e realizar tarefas.
- Análise de dados: Os dados coletados pelos dispositivos IoT podem ser analisados para gerar insights e tomar decisões.

COMBU MAKER

DEMONSTRAÇÃO

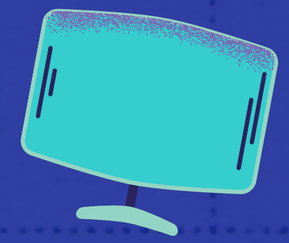




COMBU MAKER

VARIÁVEIS DE AMBIENTE

```
#ifndef ENV_H
#define ENV_H

//Variáveis MQTT
const char* MQTT_BROKER =  "test.mosquitto.org";
const char* TOPIC =  "/LAB/DOOR";
const char* MQTT_USER_NAME = "";
const char* MQTT_PASSWORD = "";
const int MQTT_PORT = 1883;

//Variáveis WiFI
const char* SSID = "nome_do_wifi";
const char* WIFI_PASSWORD = "senha_do_wifi";

#endif //ENV_H
```



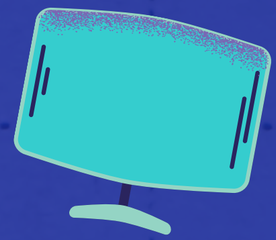
COMBU MAKER

CONEXÃO WIFI

```ino
#include <ESP8266WiFi.h>
#include "../env.h"

#ifndef WIFI_H
#define WIFI_H

void ConnectWifi(){
    WiFi.begin(SSID, WIFI_PASSWORD);

    Serial.print("\n Conectando ao Wifi ");

  while (WiFi.status() != WL_CONNECTED )
  {
    delay(500);
    Serial.print(".");
  }

  Serial.println("\n Wifi Conectado \n");
  Serial.println(WiFi.localIP());

  return;
}



#endif
```

COMBU MAKER

CÓDIGO MQTT

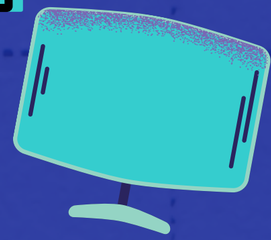


```
#include "../env.h"
#include <ESP8266WiFi.h>
#include <PubSubClient.h>
#include <Arduino.h>

#ifndef MQTT_H
#define MQTT_H


class MQTT
{
public:
    String clientId;
    WiFiClient espClient;
    bool mqttStatus;
    PubSubClient* client = new PubSubClient(espClient);

    bool connectMQTT();
    static void callback(char* topic, byte* payload, unsigned int length);

    MQTT(String);
    ~MQTT();

};

MQTT::MQTT(String clientId)
:clientId(clientId)
{

}

MQTT::~MQTT()
{
}
```

COMBU MAKER

CÓDIGO MQTT

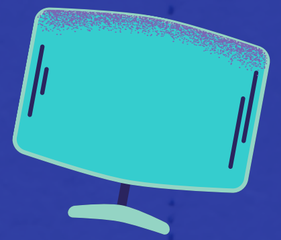
```
bool MQTT::connectMQTT(){
  byte tentativa = 0;
  client->setServer(MQTT_BROKER, MQTT_PORT);
  client->setCallback(callback);

  do
  {

    clientId += String(WiFi.macAddress());

    if(client->connect(clientId.c_str(), MQTT_USER_NAME, MQTT_PASSWORD)){
      Serial.println("Exito na conexão:");
      Serial.printf("Cliente %s conectado ao broker\n", clientId.c_str());
      mqttStatus = true;
    } else {
      Serial.println("Falha na conexão com o broker");
      Serial.print("conexão: ");
      Serial.print(client->state());
      Serial.printf("\nTentativa: %i", tentativa);
      delay(1000);
    }
    tentativa++;
  } while (!client->connected() && tentativa < 5);

  if(tentativa < 5){
    client->publish(TOPIC, "{B6 A1 D8 E7,Tiago Oliveira,202204940020,2024021610s1257}");
    client->subscribe(TOPIC);
    return 1;
  } else{
      Serial.println("Não conectado");
      return 0;
  }
}
```



COMBU MAKER

CÓDIGO MQTT

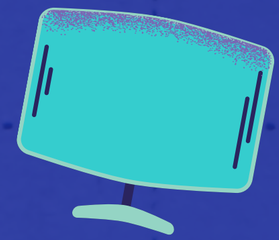
```
1  void MQTT::callback(char* topic, byte* payload, unsigned int length){
2    Serial.print("Nova mensagem no tópico: ");
3    Serial.println(topic);
4    Serial.println("Mensagem:");
5    for (unsigned int i = 0 ; i < length; i++){
6      Serial.print((char) payload[i]);
7    }
8    Serial.println("\n -----------------------------");
9  };
10
11
12 #endif
```



COMBU MAKER

MAIN.CPP

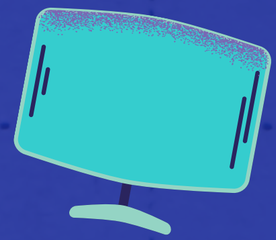
```cpp
#include "./classes/MQTT.h"
#include "./classes/WIFI.h"
#include <Arduino.h>

MQTT mqtt("T-aaago");

void setup(){
  Serial.begin(115200);
  ConnectWifi();
  mqtt.connectMQTT();
}

void loop(){
 if(mqtt.mqttStatus){
    mqtt.client->loop();
 }
}
```



COMBU MAKER

```ino
#include "./classes/MQTT.h"
#include "./classes/WIFI.h"
#include <Arduino.h>

#define pinBotao1 12

void enviaPacote();

MQTT mqtt("T-aaago");

void setup(){
  pinMode(pinBotao1, INPUT_PULLUP);
  Serial.begin(115200);
  ConnectWifi();
  mqtt.connectMQTT();
}

void loop(){
 if(mqtt.mqttStatus){
    mqtt.client->loop();
    enviaPacote();
 }
}

void enviaPacote() {
static bool estadoBotao1 = HIGH;
static bool estadoBotao1Ant = HIGH;
static unsigned long debounceBotao1;

  estadoBotao1 = digitalRead(pinBotao1);
  if (  (millis() - debounceBotao1) > 30 ) {  //Elimina efeito Bouncing
     if (!estadoBotao1 && estadoBotao1Ant) {

        //Botao Apertado
        mqtt.client->publish(TOPIC, "1");
        Serial.println("Botao1 APERTADO. Payload enviado.");

        debounceBotao1 = millis();
     } else if (estadoBotao1 && !estadoBotao1Ant) {

        //Botao Solto
        mqtt.client->publish(TOPIC, "0");
        Serial.println("Botao1 SOLTO. Payload enviado.");

        debounceBotao1 = millis();
     }

  }
  estadoBotao1Ant = estadoBotao1;
}
```

ENVIAR OS DADOS DO ESP8266 PARA O ESP-12



COMBU MAKER

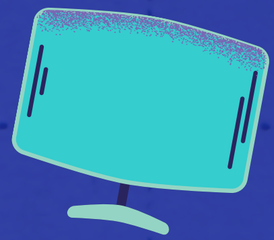
```ino
#include "./classes/MQTT.h"
#include "./classes/WIFI.h"
#include <Arduino.h>



MQTT mqtt("Listener");

void setup(){
  pinMode(D6, OUTPUT);
  Serial.begin(115200);
  ConnectWifi();
  mqtt.connectMQTT();
}

void loop(){
 if(mqtt.mqttStatus){
    mqtt.client->loop();

 }
}

```

RECEBE A MENSAGEM (ESP-12)

COMBU MAKER

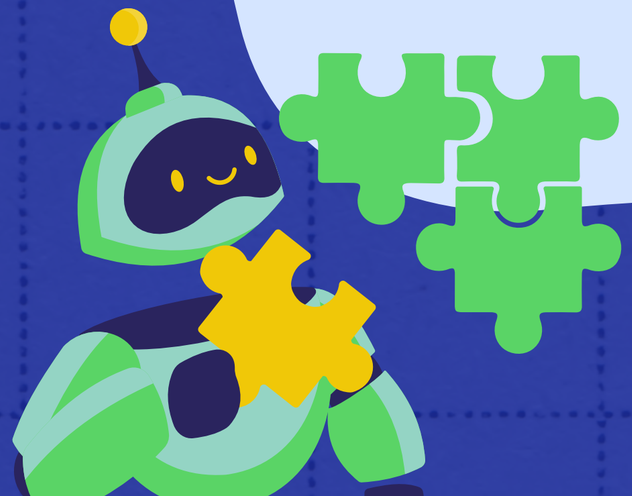
CALLBACK DO ESP-12 É DIFERENTE

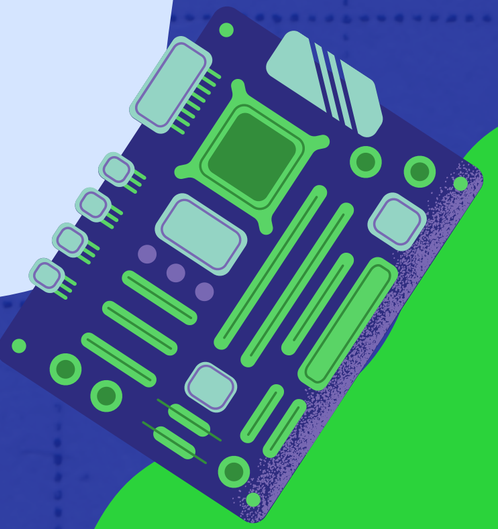
```
1  void MQTT::callback(char* topic, byte* payload, unsigned int length){
2    String buttonValue;
3
4    Serial.print("Nova mensagem no tópico: ");
5    Serial.println(topic);
6    Serial.println("Mensagem:");
7    for (unsigned int i = 0 ; i < length; i++){
8      char c = (char) payload[i];
9      Serial.print(c);
10     buttonValue += c;
11   }
12   Serial.println("\n -----------------------------");
13
14   if (buttonValue == "1"){
15     digitalWrite(D6, HIGH);
16   }
17   if (buttonValue == "0"){
18     digitalWrite(D6, LOW);
19   }
20 };
```



COMBU MAKER

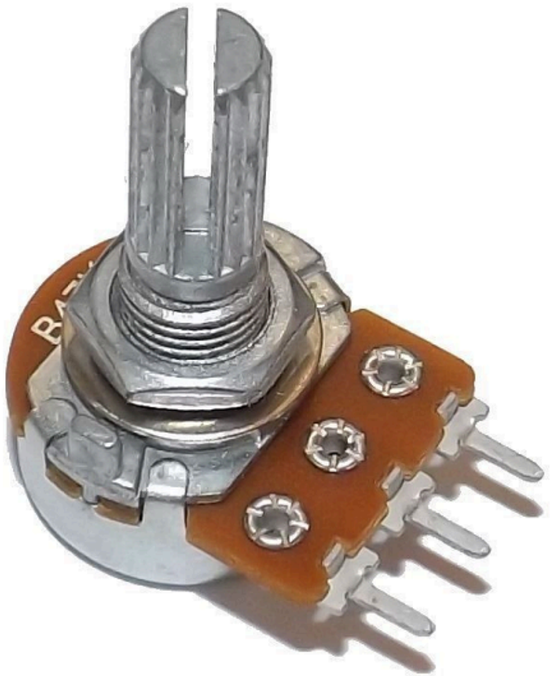
POTENCIOMETRO





COMBU MAKER

POTENCIOMETRO



O que é?

Um potenciômetro é um componente eletrônico que tem a capacidade de variar sua resistência elétrica quando um botão ou alavanca é girado. Isso significa que ele pode fornecer uma resistência variável entre seus terminais. Essa resistência variável é utilizada para controlar a quantidade de corrente que flui em um circuito.

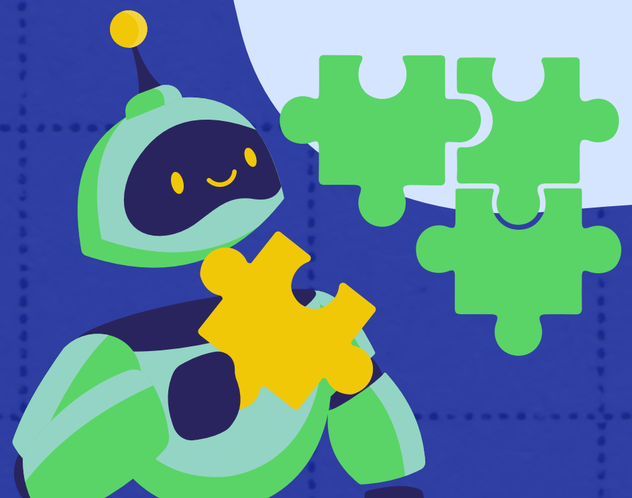
COMBU MAKER

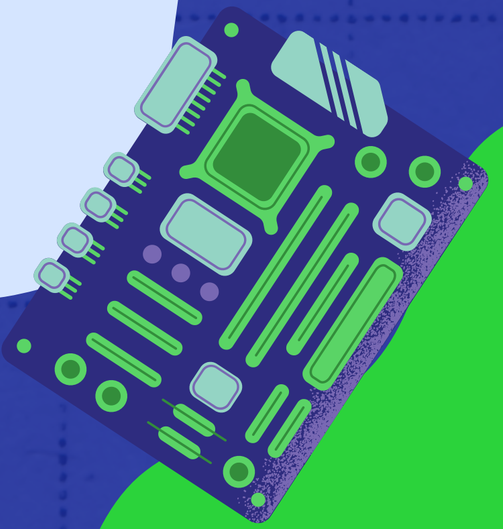
POTENCIOMETRO

Em Arduino, o potenciômetro é comumente usado como um dispositivo de entrada analógica. Ele é ligado a um dos pinos analógicos do Arduino e fornece uma voltagem analógica proporcional à posição do botão ou alavanca. Isso permite que você ajuste parâmetros de saída do programa, como a velocidade de um motor, a intensidade de uma luz ou a posição de um servo motor, de forma contínua e precisa.

COMBU MAKER

PROGRAMAÇÃO





COMBU MAKER

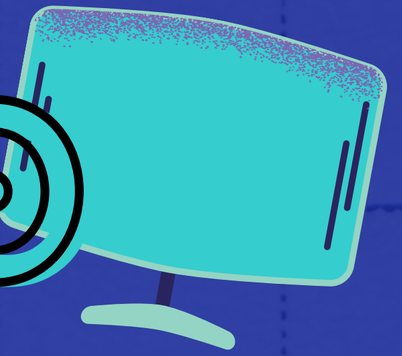
POTENCIÔMETRO

Para usar o potenciômetro, deve-se conectá-lo em uma porta analógica para fazer a leitura e no **VCC**, além do **GND**.

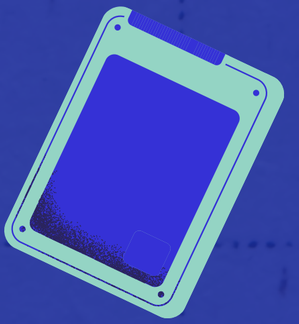
Além disso, é necessário declará-lo como INPUT no **pinMode()**

COMBU MAKER

POTENCIOMETRO



```
1   pinMode(pinBuzzer, OUTPUT);
2    pinMode(pinPot, INPUT);
```



COMBU MAKER

POTENCIOMETRO

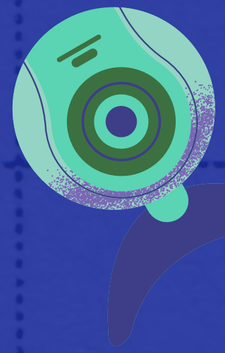
Também deve-se usar o **Serial.begin** para proporcionar a leitura.

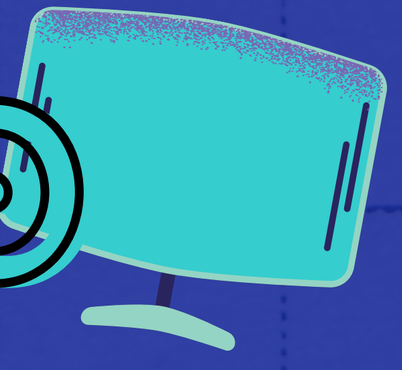
É preciso usar **analogRead()** para realizar a leitura

Pode usar a função **map()** no loop para a conversão dos valores do potenciometro nos valores que são necessários

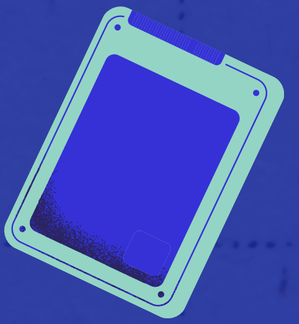
COMBU MAKER

POTENCIOMETRO





```
1   int potValue = analogRead(pinPot);
2   int frequency = map(potValue, 0, 1023, 0, 2500);
```



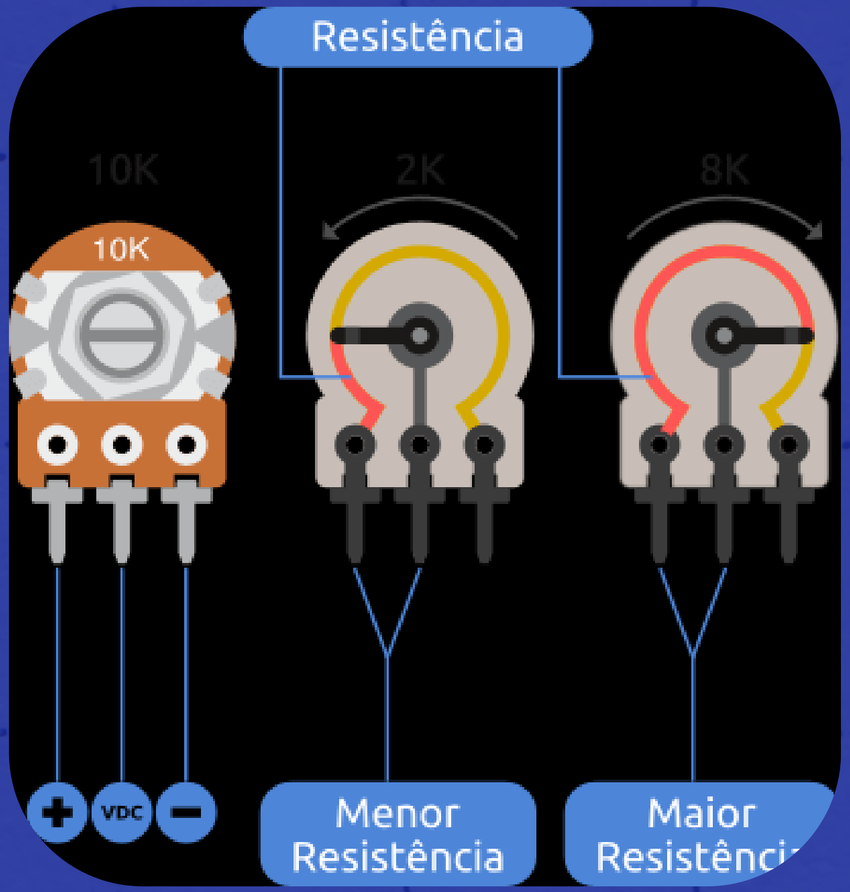
COMBU MAKER

POTENCIOMETRO

```
1   Serial.begin(9600);
```

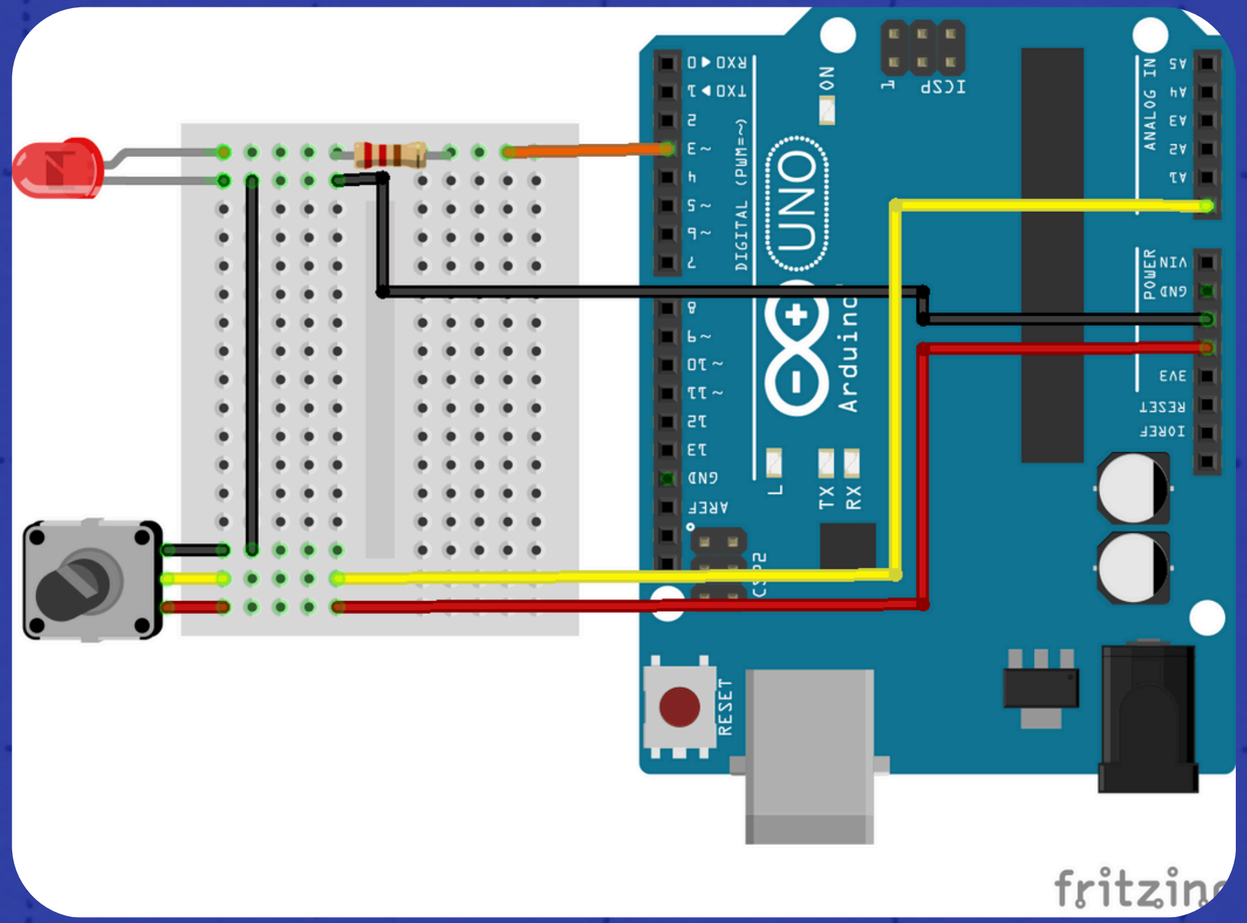
COMBU MAKER

POTENCIOMETRO



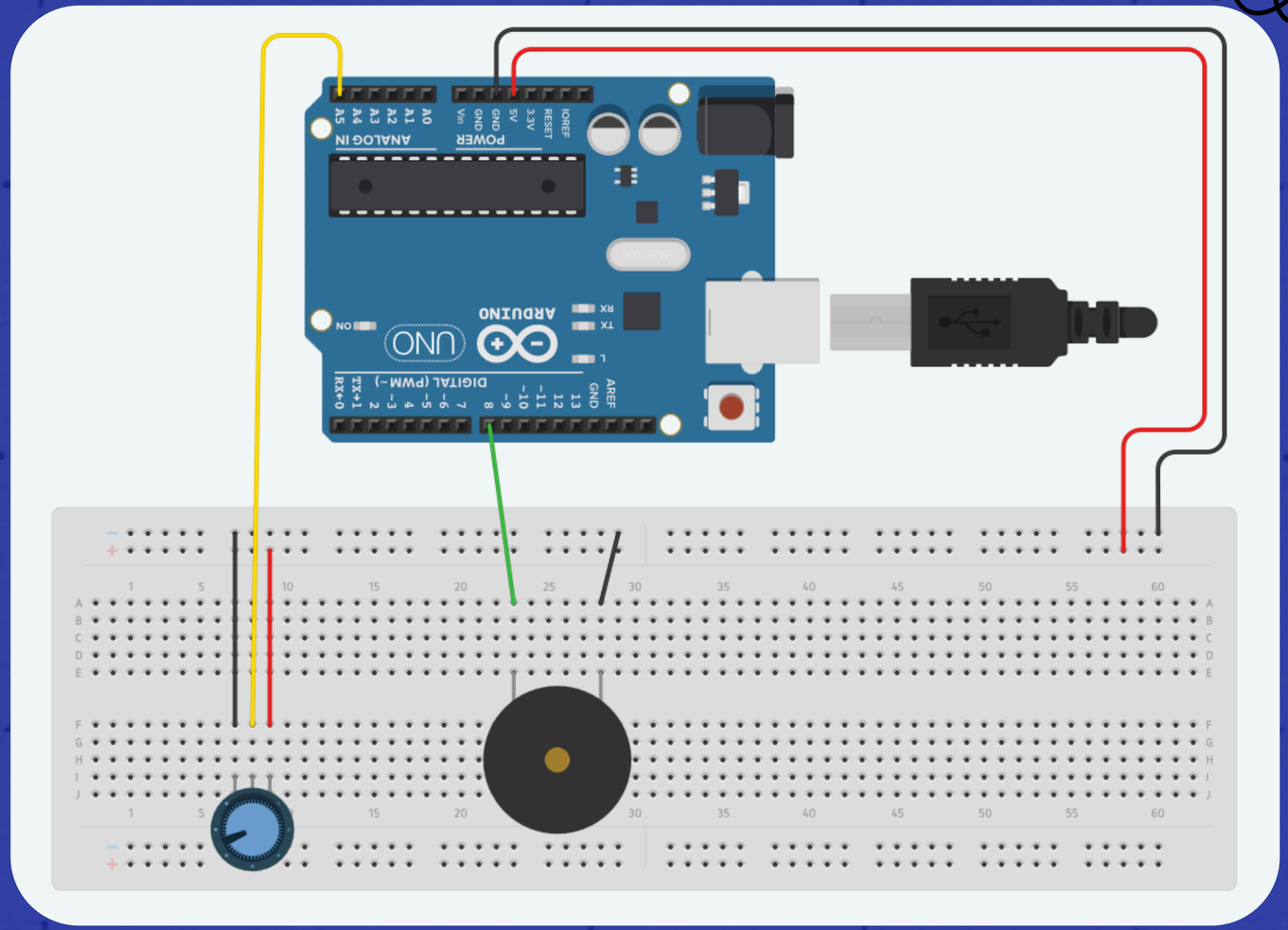
COMBU MAKER

POTENCIOMETRO



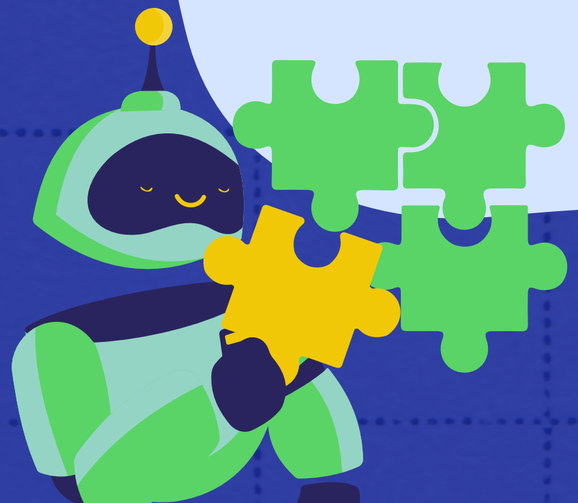
COMBU MAKER

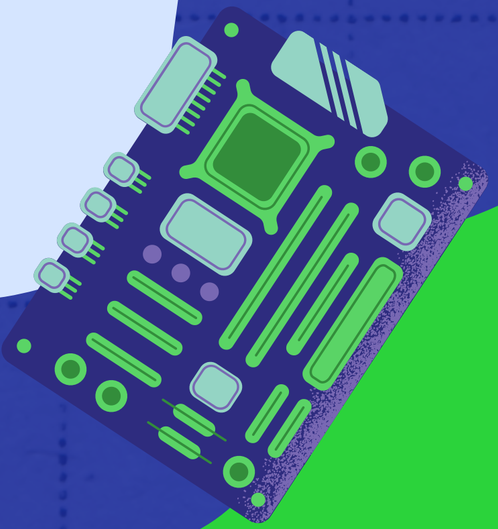
POTENCIOMETRO



COMBU MAKER

DESAFIOS





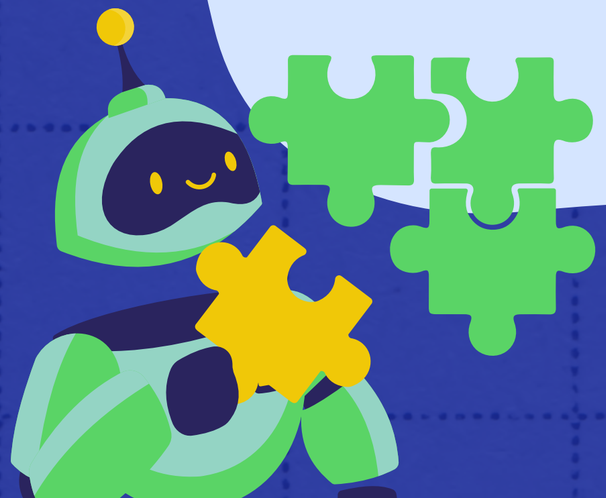
COMBU MAKER

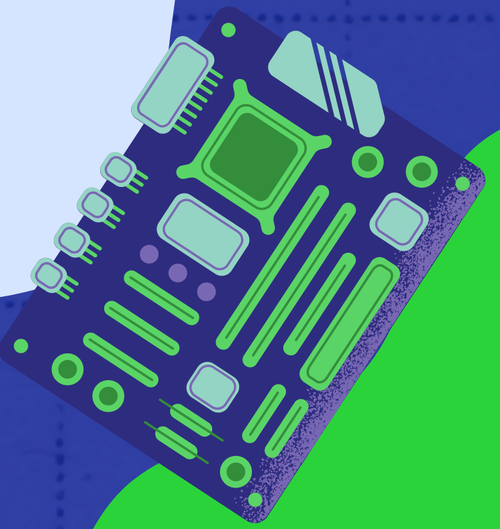
DESAFIO

1. Faça um led aumentar e decrescer sua intensidade com o potenciômetro;

2. Use um potenciômetro para alterar a intensidade de um led e um buzzer.

SERVO MOTOR





COMBU MAKER

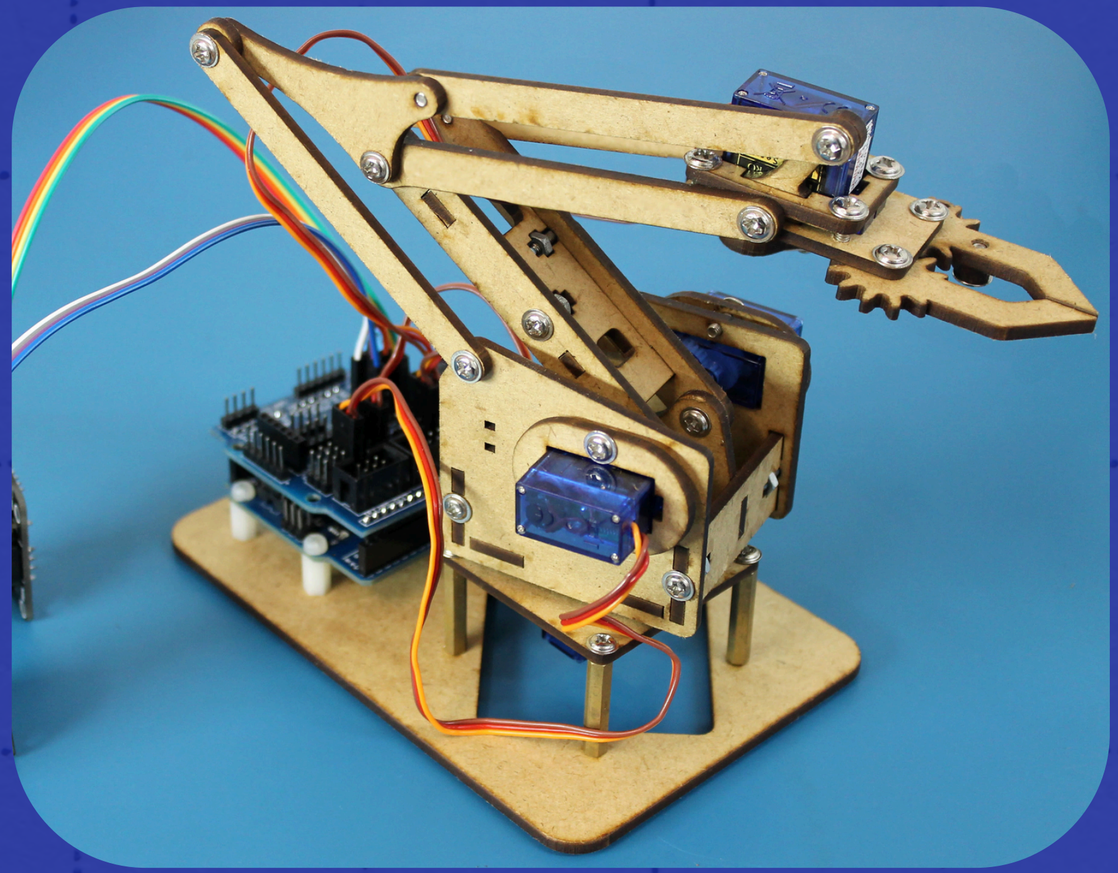
SERVO MOTOR

O que é?

Um servo motor é um atuador eletromecânico que combina um motor elétrico com um sistema de feedback. Isso significa que, além de girar, ele também possui um sensor que informa sua posição atual.

COMBU MAKER

SERVO MOTOR



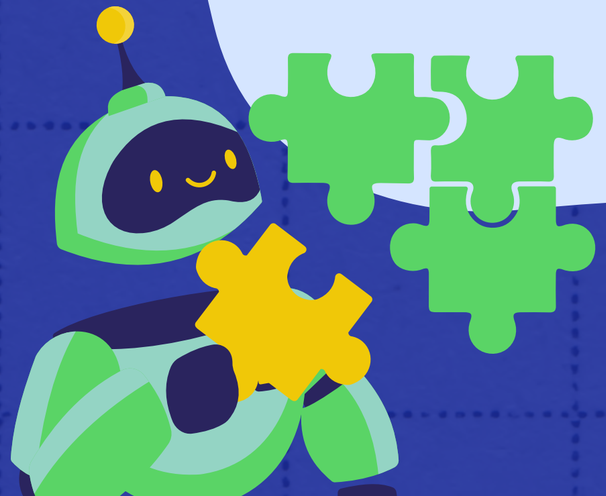
Onde é usado?

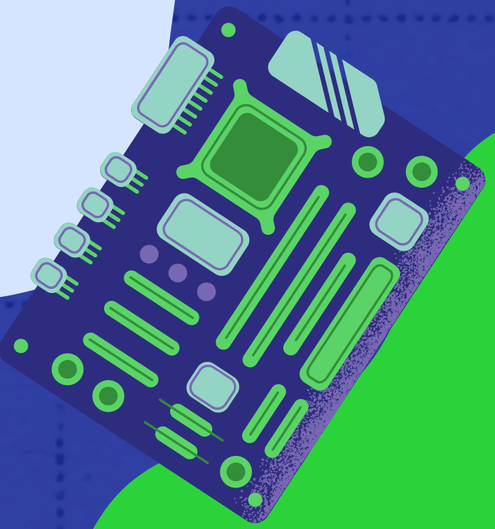
Os servo motores são usados em diversas aplicações, como:

- Robótica: Braços robóticos, drones, carros autônomos, etc.
- Automação industrial: Máquinas de embalagem, linhas de produção, etc.
- Eletrônica: Impressoras 3D, aeromodelos, etc.
- Brinquedos: Carros de controle remoto, robôs de brinquedo, etc.

COMBU MAKER

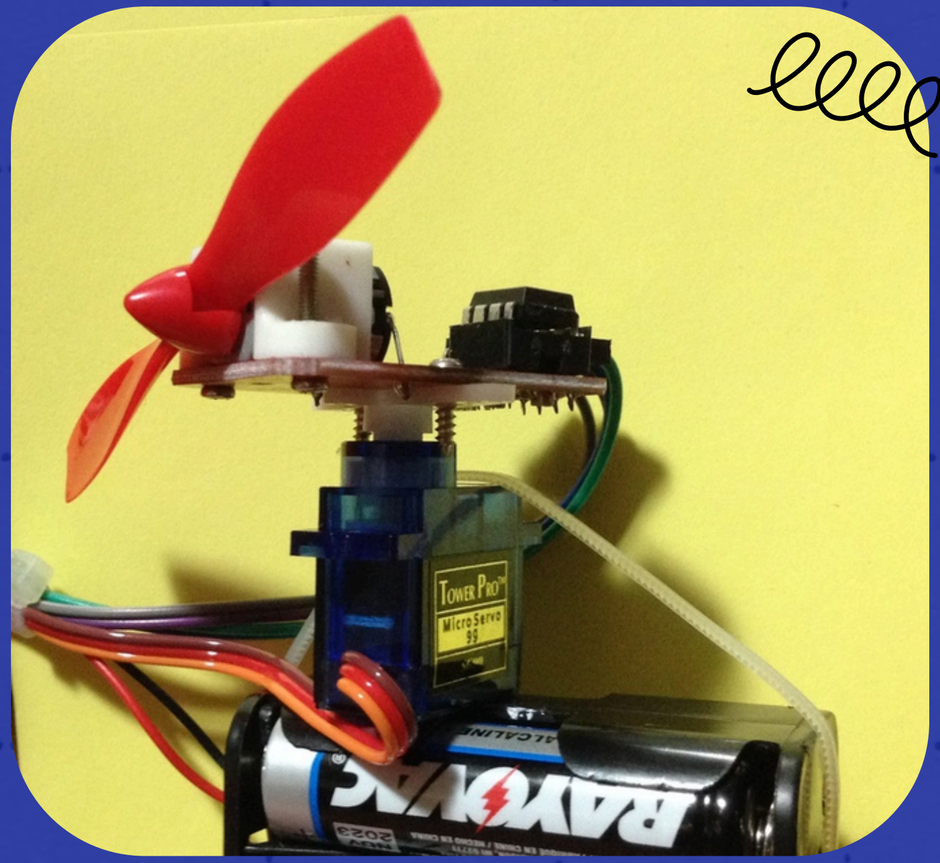
PROGRAMAÇÃO





COMBU MAKER

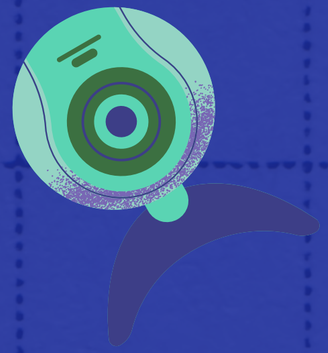
SERVO MOTOR

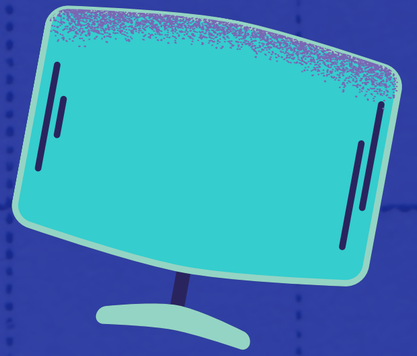


Para programar o servo motor, é necessário usar a biblioteca específica para servo motores que é a Servo.h. Além disso, é necessário declarar o tipo de servo que deve ser utilizado, nesse caso um servoMotor

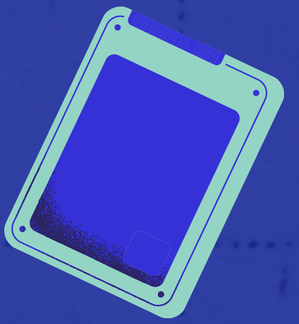
COMBU MAKER

SERVO MOTOR





```arduino
#include <Servo.h>

Servo servoMotor; // Cria um objeto do tipo Servo
```



COMBU MAKER

SERVO MOTOR

Para programar o servo motor, é necessário usar a biblioteca específica para servo motores que é a Servo.h. Além disso, é necessário declarar o tipo de servo que deve ser utilizado, nesse caso um servoMotor

COMBU MAKER

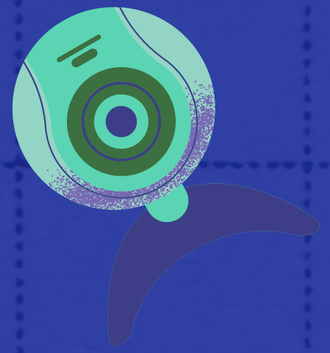
ARDUINO

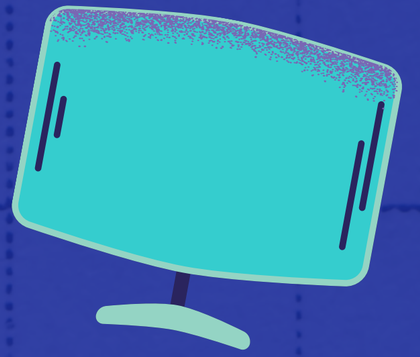
Além disso, no setup deve colocar o função da biblioteca servoMotor.attach que define o pino usado no servo motor.

Também, deve-se fazer o uso dqa função Serial.begin, que permite a comunicação entre o servoMotor e os outros componentes da placa

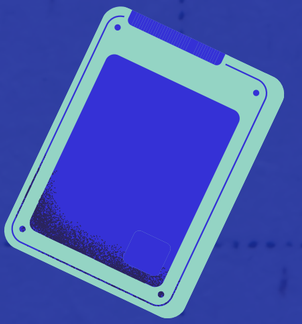
COMBU MAKER

SERVO MOTOR





```
1     servoMotor.attach(9); // Atribui o pino digital 9 ao servo
  motor
2     Serial.begin(9600); // Inicia a comunicação serial
3   }
```



COMBU MAKER

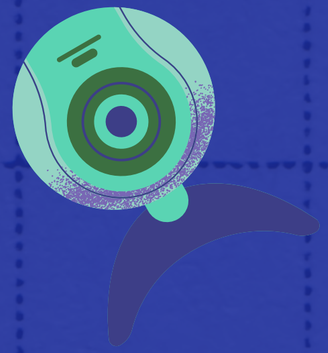
ARDUINO

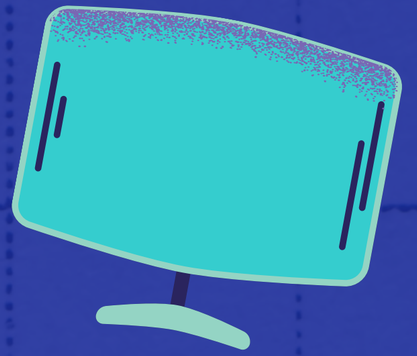
Para provocar um comando de movimento no servo motor, no void loop, deve-se usar a função servoMotor.write para definir a posição do servo motor

Lembrem-se que o servo motor precisa ser conectado a uma entrada pwm

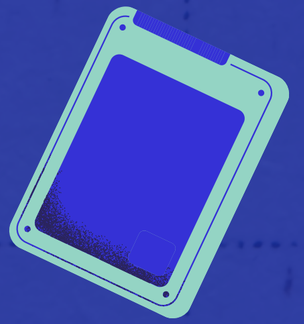
COMBU MAKER

SERVO MOTOR



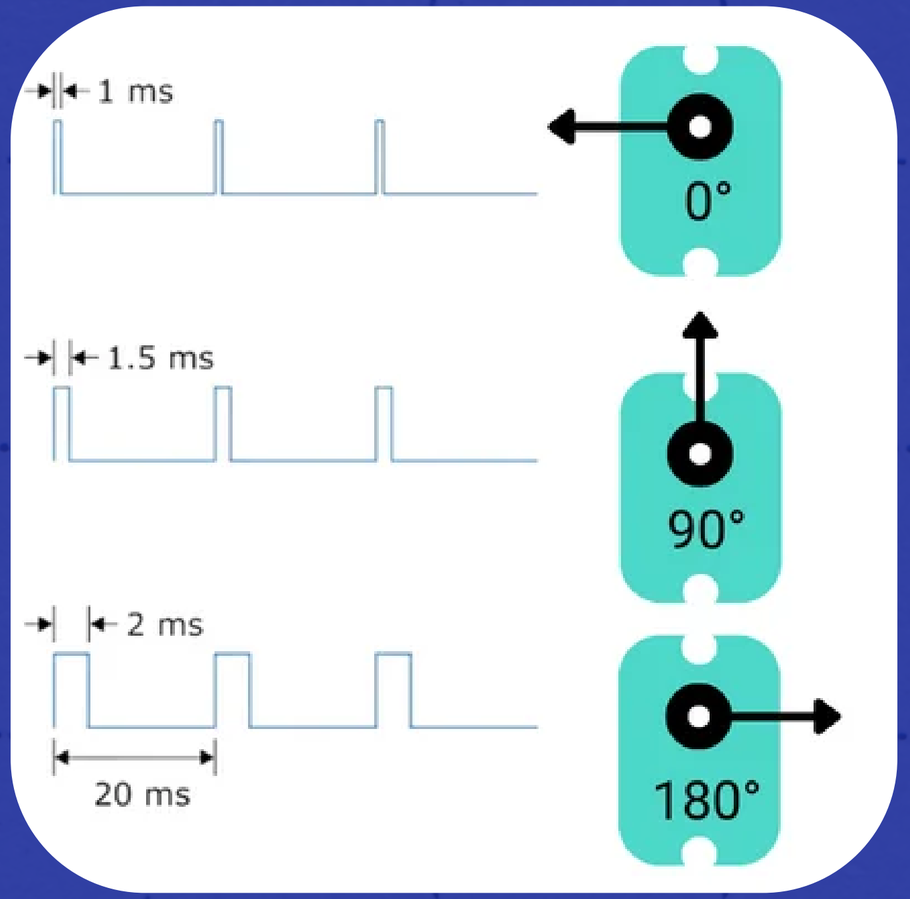


```
servoMotor.write(posicao);
delay(100);
```



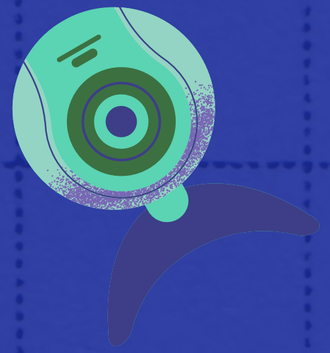
COMBU MAKER

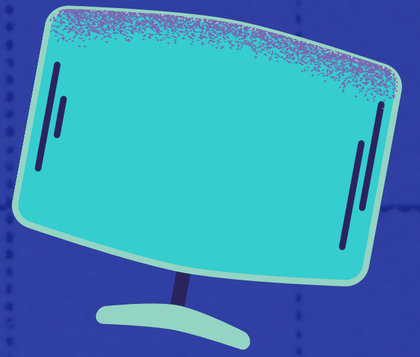
SERVO MOTOR



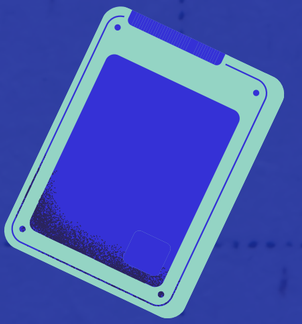
COMBU MAKER

SERVO MOTOR



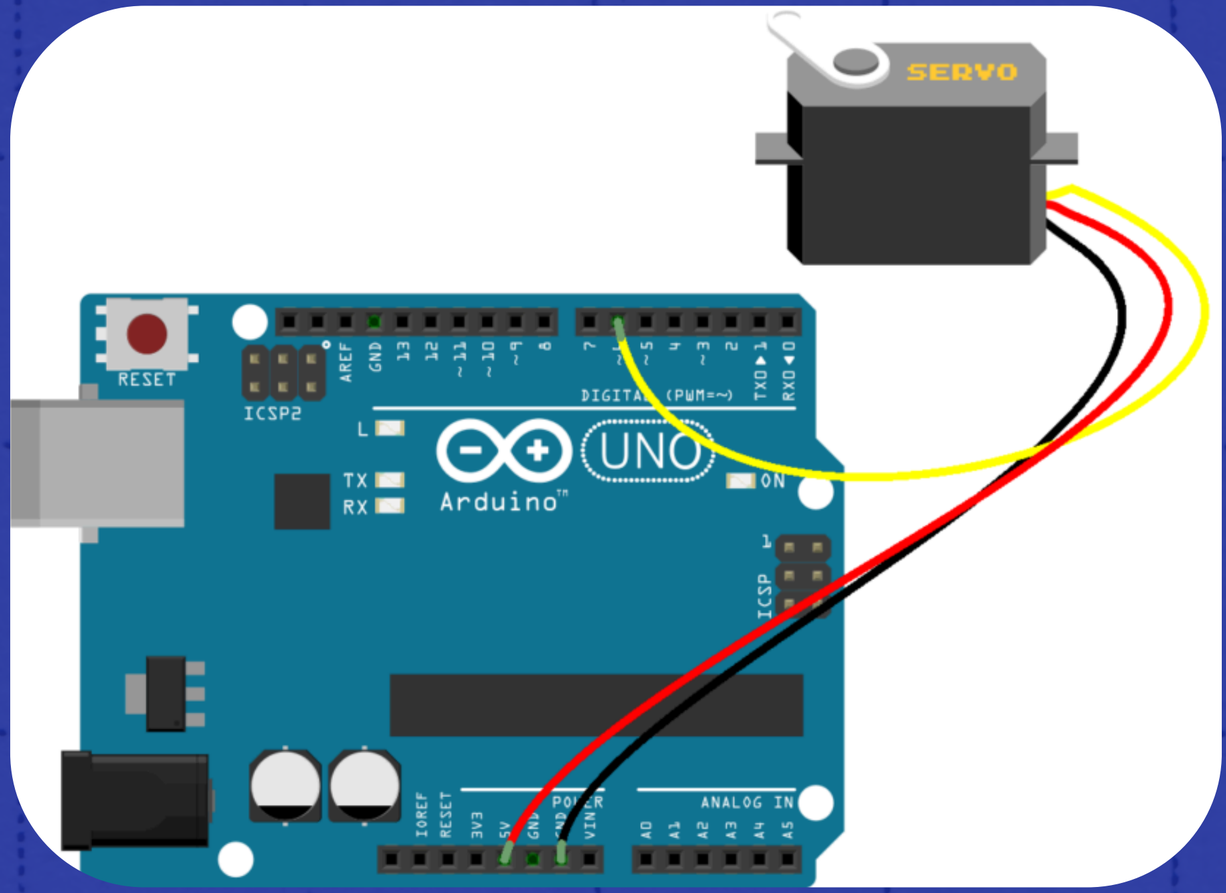


```
1     servoMotor.attach(9); // Atribui o pino digital 9 ao servo
  motor
2     Serial.begin(9600); // Inicia a comunicação serial
3   }
```



COMBU MAKER

SERVO MOTOR



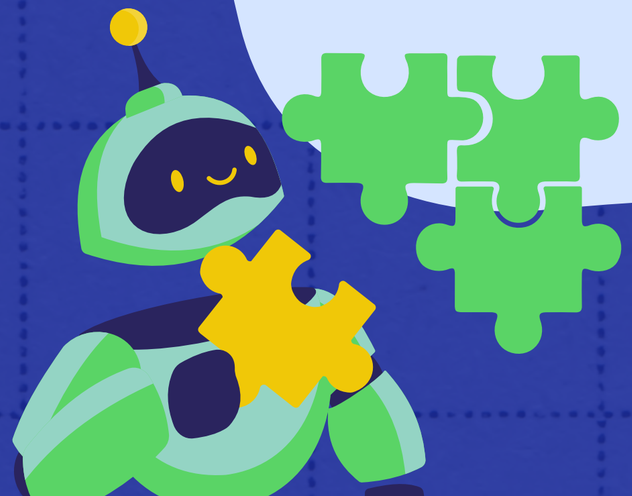
COMBU MAKER

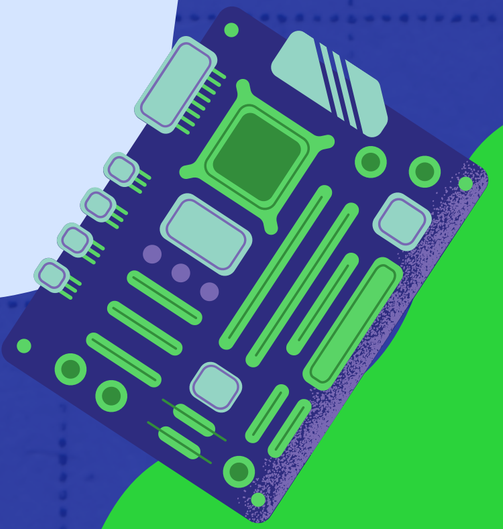
SERVO MOTOR

PWM = Laranja (⎍⎍)
Vcc = Vermelho (+)
Aterrado = Marron (–)

COMBU MAKER

DESAFIOS





COMBU MAKER

DESAFIO

1. Faça o servo motor rotacionar 180°;

2. Use um potenciômetro para controlar o servo motor.