Learning Unit	
Subject	Programação
Title	Aprender C++ com o Robô Botn´Roll
Authors	Nuno Barbosa
School	FORAVE – ASSOCIAÇÃO PARA A EDUCAÇÃO TECNOLÓGICA DO VALE DO AVE
Description of the unit	O objetivo desta unidade é aprender as bases de programação em C++ usando Robôs.
Contents	Programação em linguagem C++: - placa de desenvolvimento da Arduino - algoritmia - usar o IDE da Arduino - sensores de infravermelhos - sensor seguidor de linha
Learning Outcomes / Skills	Os alunos devem ser capazes de: ●Desenvolver o espírito crítico e a capacidade de trabalhar em grupo; ●Desenvolver a capacidade de resolução de problemas; • Desenvolver persistência, autonomia e à-vontade em lidar com situações que envolvam a programação no seu percurso escolar e na vida em sociedade; • Desenvolver interesse pela programação e valorizar o seu papel no desenvolvimento das outras ciências e domínios da atividade humana e social.
Target students/class	Ensino secundário (15 – 17 anos)
Prerequisites	Os alunos devem ser capazes de: ●Realizar fluxogramas de forma a estruturarem a resolução de um problema; ●Realizar pseudocódigo de forma a estruturarem a resolução de um problema; ●Utilizar compiladores/interpretadores; ●Identificar comandos em C++; ●Saber os comandos em C++ necessários para controlar um robô Botn´Roll ●Identificar as bibliotecas necessárias ao controlo de um robô Botn´Roll.
Time expected	4 horas

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Learning Unit	
Interdisciplinary links	TIC
Methodology	Exposição dos conteúdos, resolução de exercícios e problemas, resolução de fichas de trabalho e trabalho em pares.
Human Resources (internal and/or external)	Professor da Componente Técnica
Resources	●Fichas de trabalho; ●Robôs Botn´Roll; ●Computadores portáteis.
Lesson Plan	**<u>1.ª Aula</u>:** **Sumário**: Algoritmia. Fluxogramas. Pseudocódigo. O professor introduz os conceitos teóricos relacionados com fluxogramas e pseudocódigo. Depois de introduzir os conceitos, e analisar o exemplo resolvido, o professor propõe a resolução do exercício n.º 1, da ficha de trabalho, em pares. Esclarecimento de dúvidas. **<u>2.ª Aula</u>:** **Sumário**: Funcionamento de um robô. O professor introduz os conceitos teóricos relacionados com o hardware e software necessário para controlar um Robô. Os alunos deverão fazer upload de um programa exemplo para o robô Botn´Roll e analisar o seu comportamento . **<u>3.ª Aula</u>:** **Sumário**: IDE da Arduino. Biblioteca Btnroll O professor expõe os conceitos necessários á compreensão do IDE da Arduino e da biblioteca Btnroll e realiza um pequeno exercício exemplo.

Co-funded by the Erasmus+ Programme of the European Union

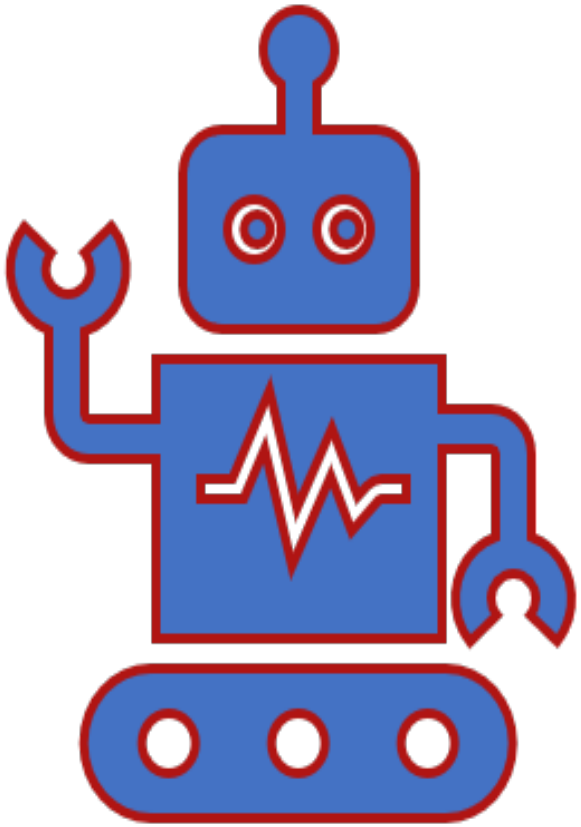
This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Learning Unit	
	Depois de introduzir os conceitos o professor propõe a resolução dos exercícios n.º 2, 3 e 4 da ficha de trabalho, em pares. Esclarecimento de dúvidas. **<u>4.ª Aula</u>:** **Sumário**: Biblioteca Btnroll. O professor expõe os conceitos necessários á compreensão da biblioteca Btnroll. Depois de introduzir os conceitos o professor propõe a resolução do exercício n.º 5 da ficha de trabalho, em pares. Correção do exercício por um dos pares a selecionar.
Assessment	**Avaliação Formativa:** • Assiduidade; ●Pontualidade; ●Comportamento: ●Atenção e participação na aula; ●Observação do desempenho do aluno na resolução dos exercícios propostos; ●Realização de fichas de trabalho (Grelhas de observação direta).
21st Century Skills	Pensamento crítico: os alunos serão capazes de analisar dados durante experiências práticas e comunicar as suas conclusões. Colaboração: os alunos serão capazes de colaborar nos seus grupos e com os restantes grupos, ajudarem-se mutuamente a compreender os conteúdos e as atividades experimentais. Comunicação: Os alunos devem ser capazes de partilhar conclusões e dúvidas com os seus colegas e professores. Literacia tecnológica: os alunos serão capazes de utilizar diferentes ferramentas tecnológicas para realizar as tarefas.
Remarks	--

Co-funded by the Erasmus+ Programme of the European Union

This project has been funded with support from the European Commission. This publication reflects the views only of the author, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Programação

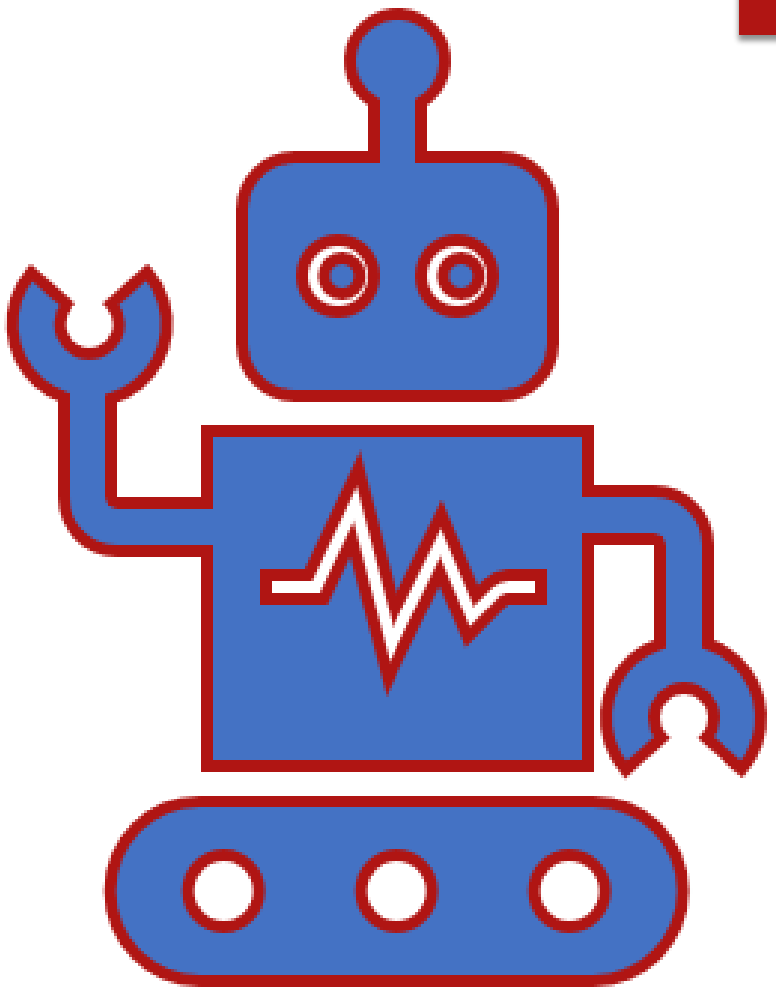


Co-funded by the Erasmus+ Programme of the European Union

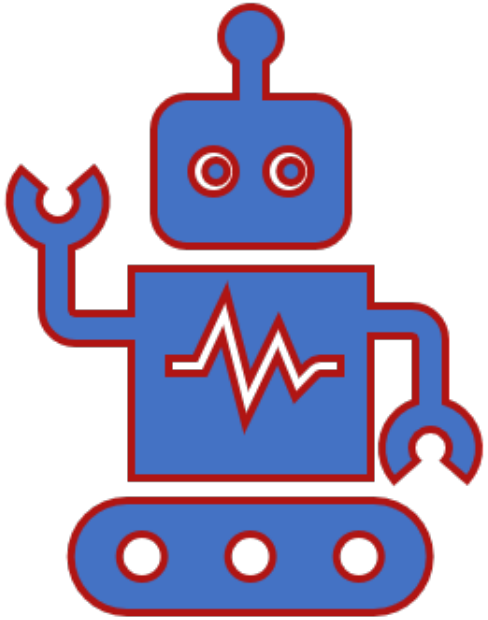
The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Conteúdos Programáticos

- Morfologia do robô
- Atuadores e Sensores para robótica
- Plataformas robóticas baseadas em microcontroladores Arduíno
- Resolução de avarias com recurso à utilização de microcontroladores Arduíno



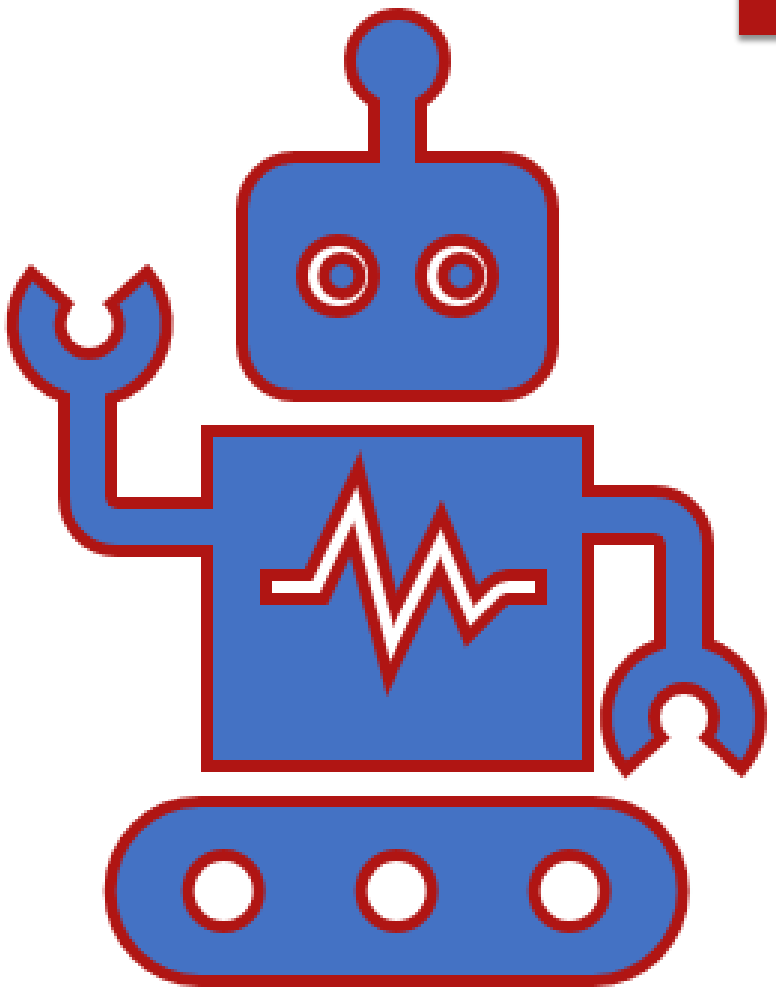
O que é um Robô ?



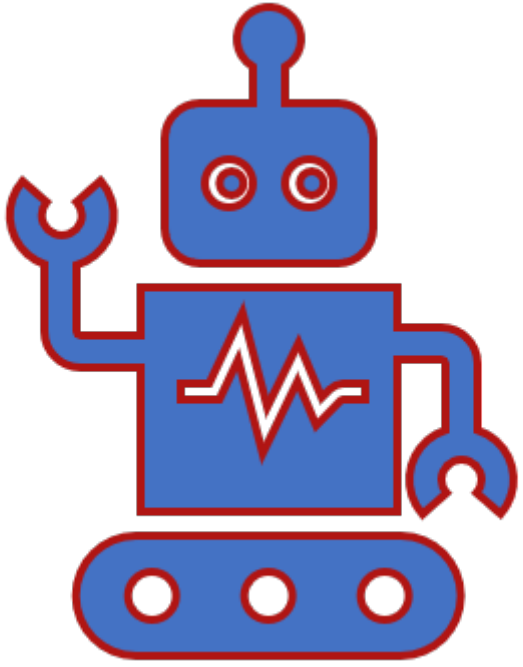
- Um robô industrial é oficialmente definido pela ISO como um "manipulador multipropósito controlado automaticamente, reprogramável, programável em três ou mais eixos".
- As aplicações típicas dos robôs industriais incluem fundição, pintura, soldagem, montagem, movimentação de cargas, inspeção de produtos, e realização de teste, tudo realizado com uma precisão, velocidade, e robustez relativamente elevadas.

Características de funcionamento e programação de robôs

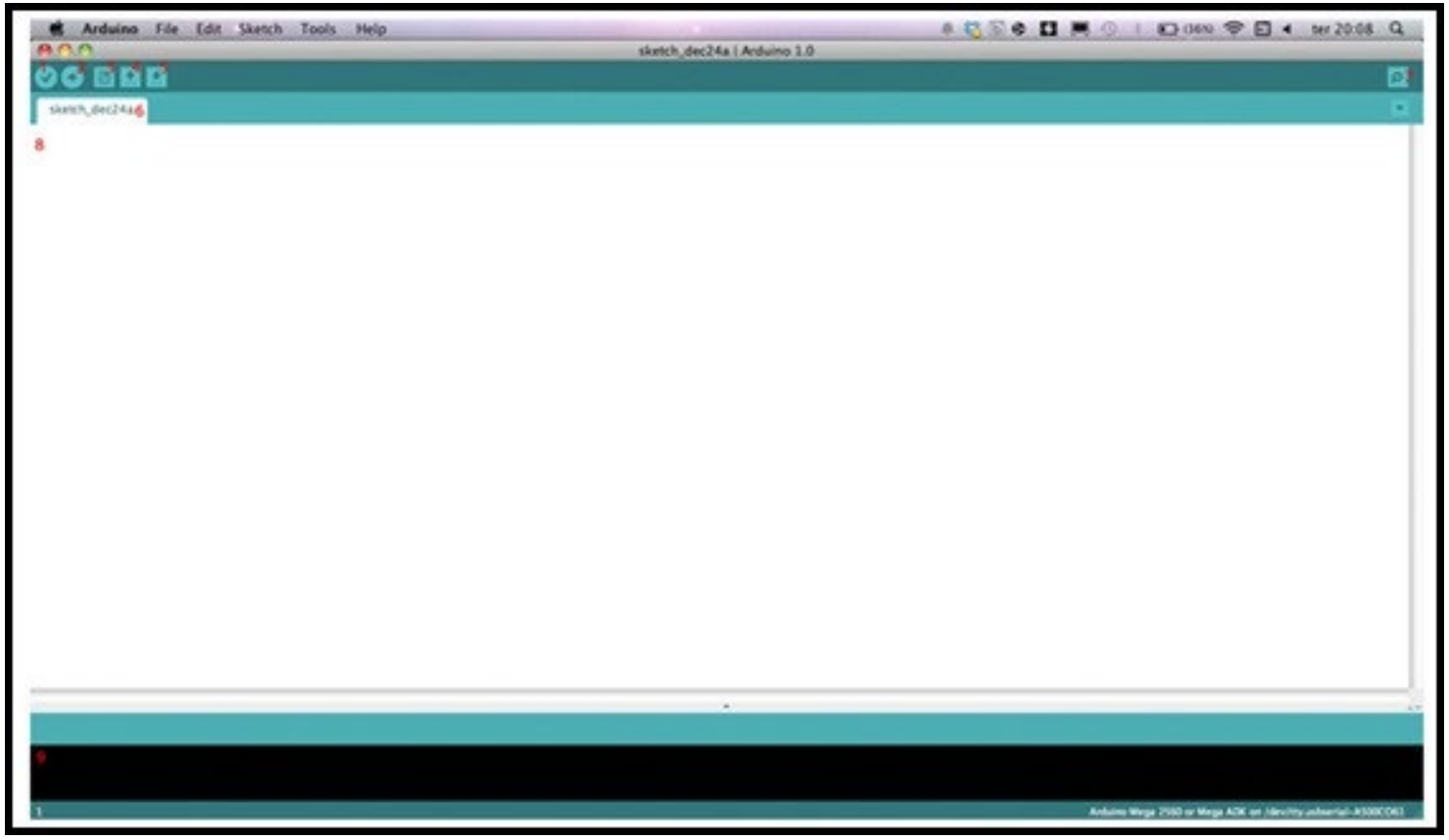
- Definem ações por leitura de valores apresentados por sensores
- São reprogramáveis, logo de fácil reutilização e reintegração em linhas de produção;
- As suas ações são validadas de forma automática sem que necessitem de interação humana para o desempenho da sua função;
- morfologia e cinemática dos robôs;
- métodos de navegação para robôs
- Programação de robôs podem ser baseados na plataforma Arduíno, Raspeberry Pi e utilização do sistema operativo ROS (Robotic Operating System).



Programação de robôs baseados na plataforma Arduíno



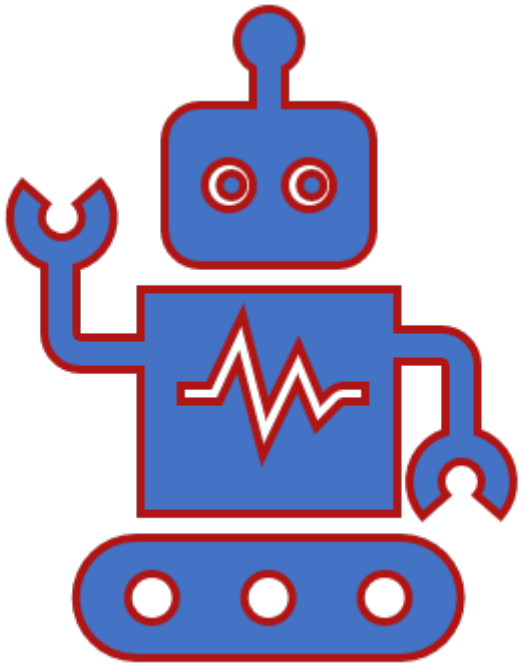
- IDE

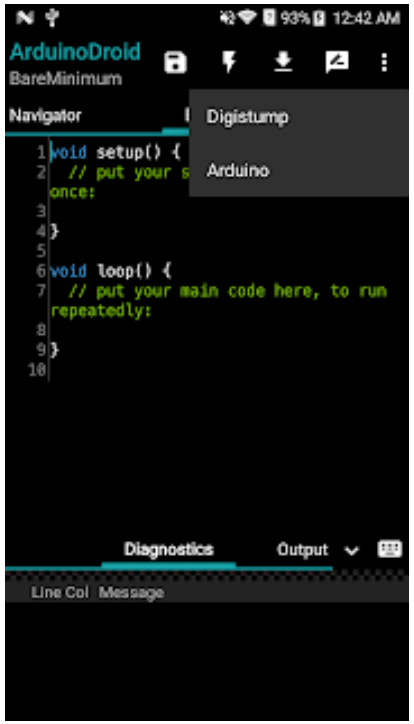


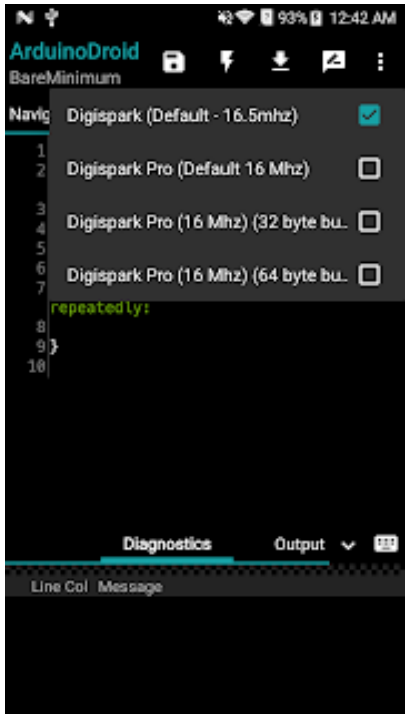
- 1 – **Compilar** – Esta função serve para verificar se há algum erro. Se houver algum erro irá ser mostrado na parte 9 (zona inferior da interface).
- 2 – **Enviar** – Este botão serve para compilar e enviar o código para o Arduino.
- 3 – **Novo Ficheiro** – Este botão serve para criarmos um novo ficheiro.
- 4 – **Abrir** – Ao clicar neste botão irá aparecer uma janela (explorador) para navegarmos até ao ficheiro que pretendemos abrir
- 5 – **Guardar** – Como o nome diz, este botão serve para guardarmos o nosso ficheiro.
- 6 – Aqui temos o nome do ficheiro.
- 7 – "**Serial Monitor**" – Ao clicarmos aqui, irá aparecer uma janela onde são mostrados alguns dados (caso esteja na programação para os mostrar, iremos aprender isso numa publicação futura.
- 8 – Aqui será onde vamos escrever o nosso código.
- 9 – Caso haja erros na programação, irão aparecer aqui.

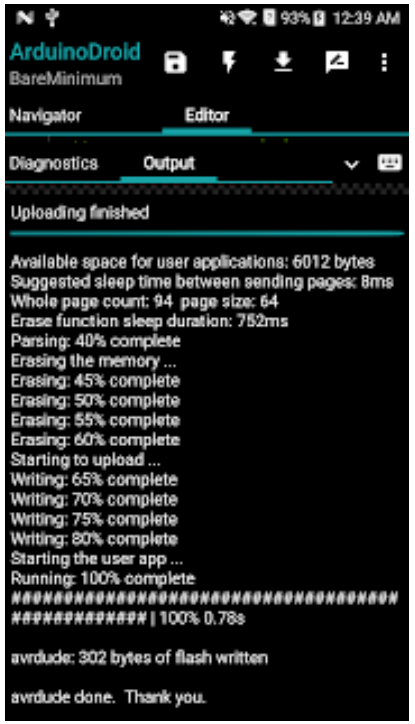
Programação de robôs baseados na plataforma Arduíno

- ArduinoDroid

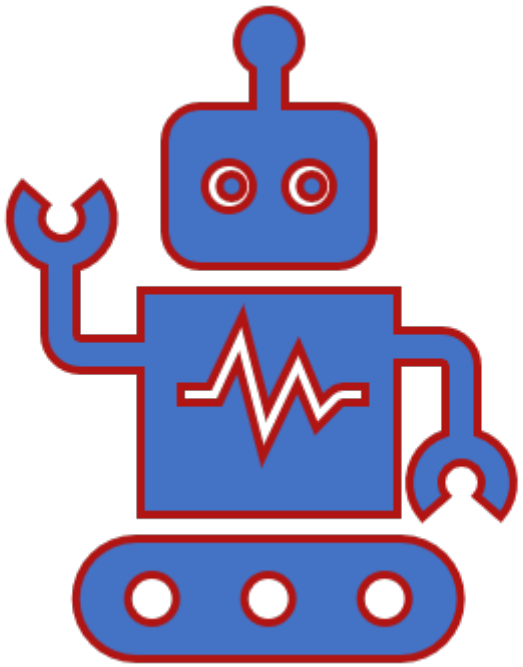




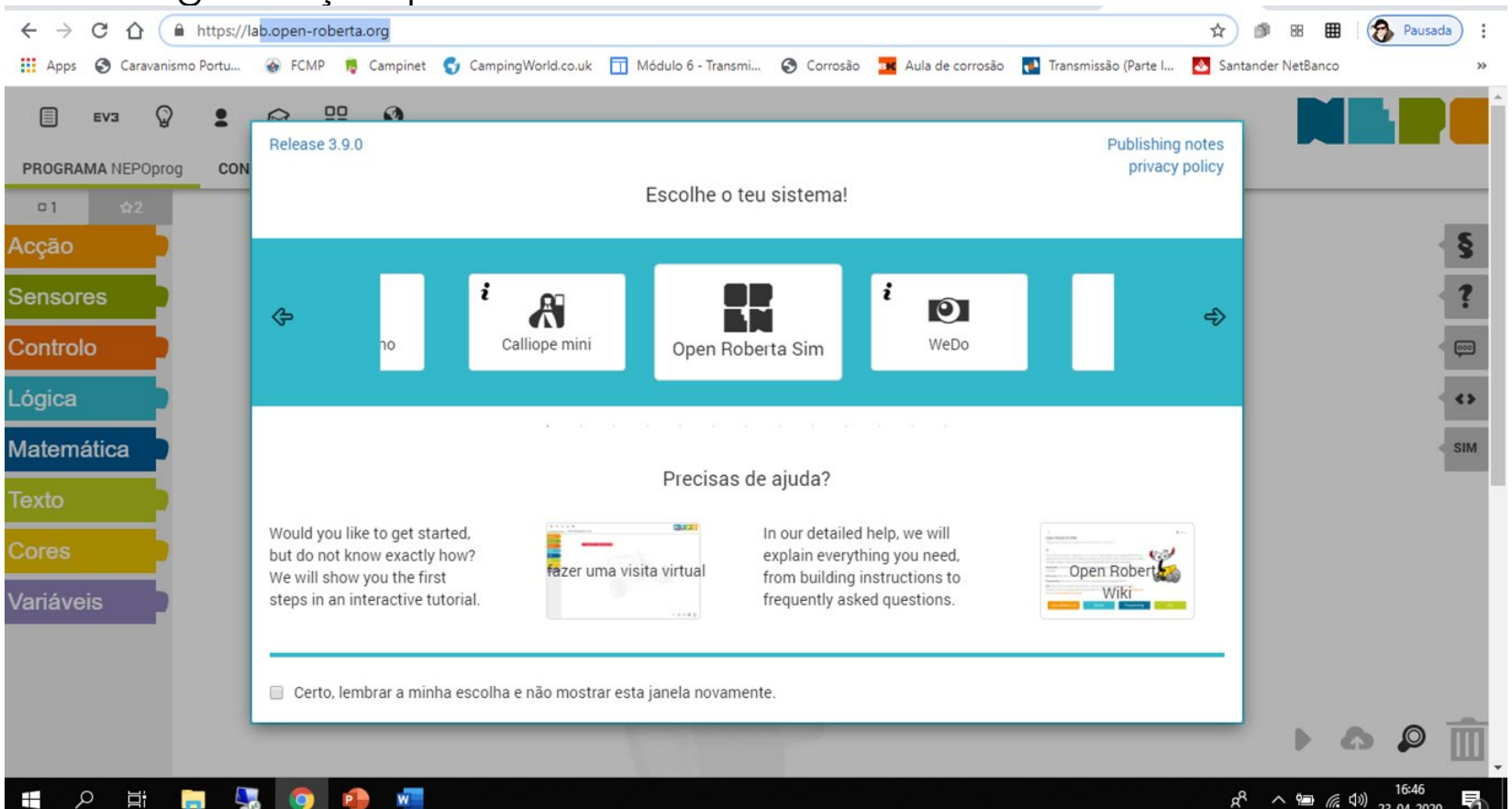




Programação de robôs baseados na plataforma Arduíno



- Programação por Blocos



Sensores

Sensor ótico refletivo

Sensor de Pulsação Infravermelo para Batimentos Cardíacos

Módulo Sensor de Nível de Água

Módulo Sensor de distância analógico

Módulo Sensor voltage 220v

Sensor de Radiação Ultravioleta UV UVM-30A

Módulo Sensor de Movimento e Vibração

Módulo Sensor de Temperatura e Humidade

Módulo Sensor de Gás

Entre muitas opções existe ainda a possibilidade de adaptação de todos os sensors convencionais à nossa necessidade…

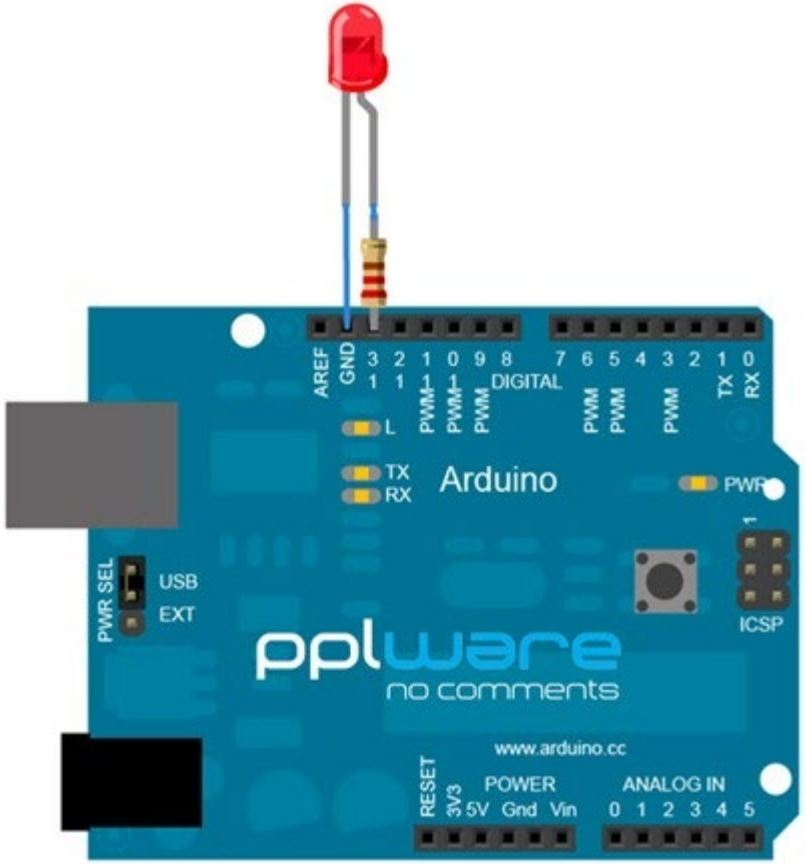
Programar

- Primeiro programa no Arduino. Desafio…

fazer um LED piscar.

Vamos recorrer à plataforma

tinkercad.com

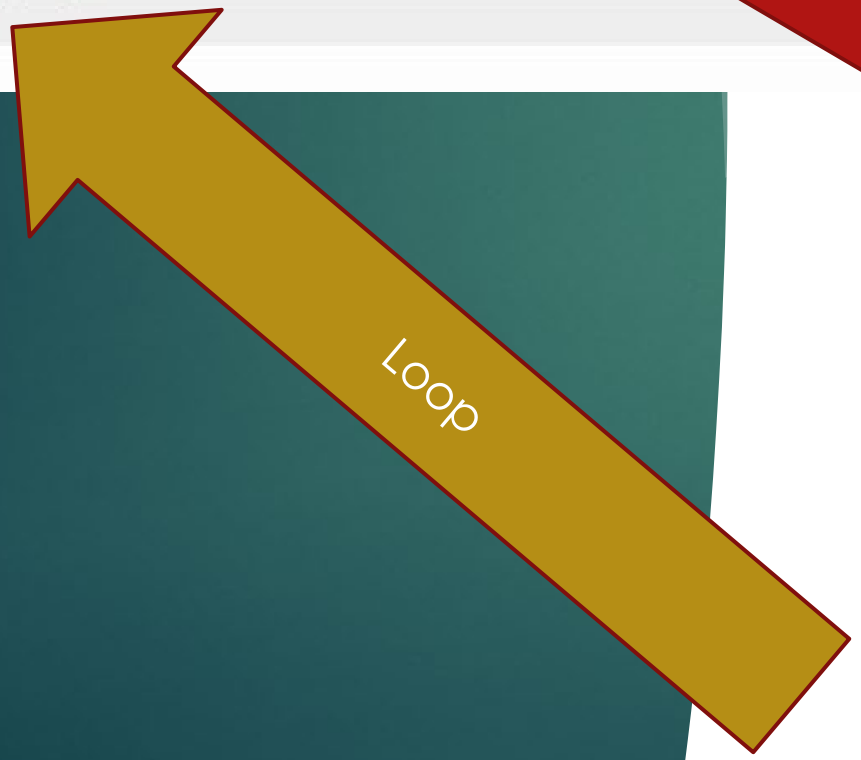


```
void setup() {
pinMode(13, OUTPUT);

}

void loop() {

}
```



```
void setup() {

pinMode(13, OUTPUT);

}

void loop() {

digitalWrite(13, HIGH);

}
```

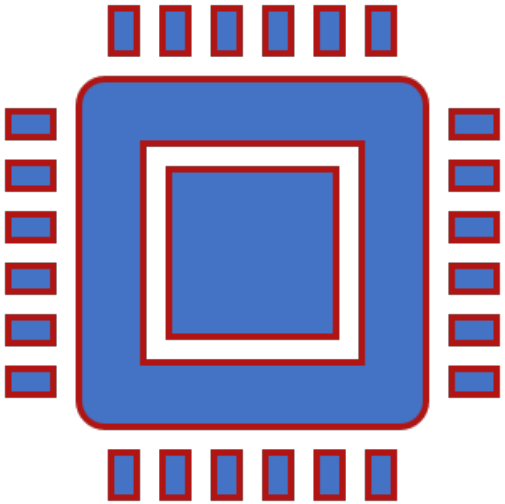
```
void setup() {
pinMode(13, OUTPUT);
}
void loop() {
digitalWrite(13, HIGH);
delay(1000);
digitalWrite(13, LOW);
delay(1000);
}
```

Comandos usuais

FUNÇÕES

ENTRADAS E SAIDAS DIGITAIS

digitalRead()



Lê o valor de um pino digital especificado, que pode ser HIGH ou LOW.

Sintaxe

digitalRead(pino)

Parâmetros
pino: o número do pino digital do Arduino que você quiser verificar

Retorna
HIGH ou LOW

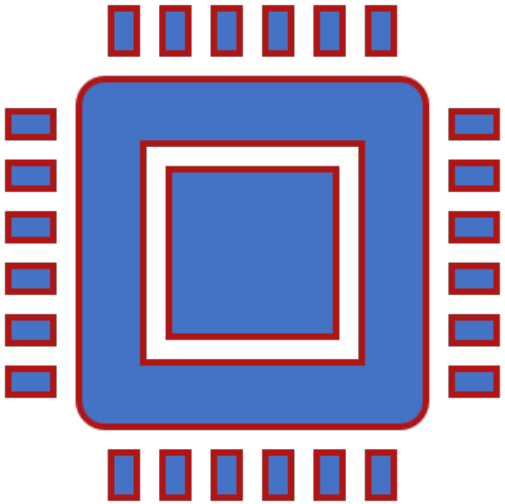
Código de Exemplo
Aciona o pino 13 para o mesmo valor que o pino 7, declarado como entrada.

```
int ledPin = 13;  // LED conectado ao pino digital 13
int inPin = 7;    // botão conectado ao pino digital 7
int val = 0;      // variável para guardar o valor lido

void setup() {
  pinMode(ledPin, OUTPUT);    // configura o pino digital 13 como saída
  pinMode(inPin, INPUT);      // configura o pino digital 7 como entrada
}

void loop() {
  val = digitalRead(inPin);   // lê o pino de entrada
  digitalWrite(ledPin, val);  // aciona o LED com o valor lido do botão
}
```

digitalWrite()



Aciona um valor HIGH ou LOW num pino digital.

Sintaxe

digitalWrite(pino, valor)

Parâmetros

pino: o número do pino do Arduino

valor: HIGH ou LOW

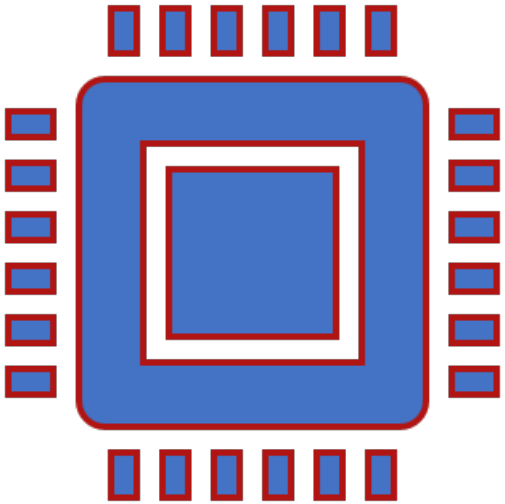
Código de Exemplo

0 código configura o pino digital 13 como OUTPUT e troca o seu estado entre HIGH e LOW

```
void setup() {
  pinMode(13, OUTPUT);    // configura o pino digital 13 como saída
}

void loop() {
  digitalWrite(13, HIGH); // ativa o pino digital 13
  delay(1000);            // espera por um segundo
  digitalWrite(13, LOW);  // desativa o pino digital 13
  delay(1000);            // espera por um segundo
}
```

pinMode()



Configura o pino especificado para funcionar como uma entrada ou saída.

Sintaxe

pinMode(pino, modo)

Parâmetros

pino: the número do pino do Arduino no qual se quer configurar o modo

Código de Exemplo

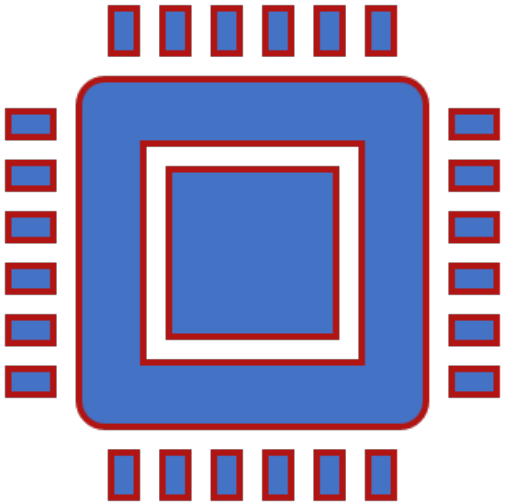
o código configura o pino digital 13 como OUTPUT e troca seu estado entre HIGH e LOW

```
void setup() {
  pinMode(13, OUTPUT);    // configura o pino digital 13 como saída
}

void loop() {
  digitalWrite(13, HIGH); // ativa o pino digital 13
  delay(1000);            // espera por um segundo
  digitalWrite(13, LOW);  // desativa o pino digital 13
  delay(1000);            // espera por um segundo
}
```

ENTRADAS E SAIDAS ANALÓGICAS

analogRead()



Lê o valor de um pino analógico especificado.

Sintaxe

analogRead(pino)

Parâmetros

pino: o nome do pino de entrada analógica que se quer ler

Código de Exemplo

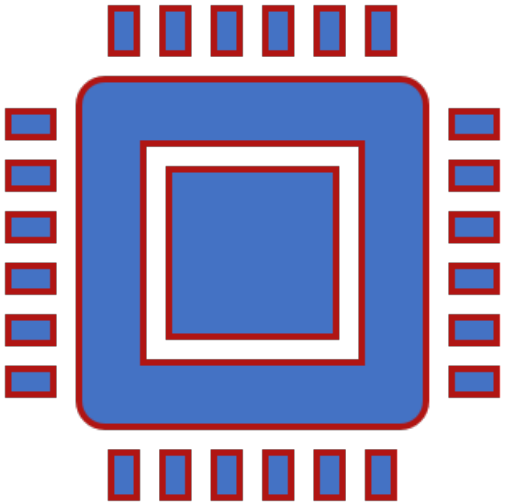
O código abaixo lê o valor de um pino de entrada analógica e mostra seu valor na porta serial.

```
int analogPin = A3; // terminal do meio de um potênciometro conectado ao pino analógico 3
                    // terminais mais externos são conectados um no ground e o outro em +5V
int val = 0;        // variável para guardar o valor lido

void setup() {
  Serial.begin(9600);           // configura a porta serial
}

void loop() {
  val = analogRead(analogPin);  // lê o pino de entrada
  Serial.println(val);          // imprime o valor na porta serial
}
```

analogWrite()



Aciona uma onda PWM em um pino. Pode ser usada para variar o brilho de um LED ou acionar um motor a diversas velocidades. Após a função analogWrite() ser chamada, no pino haverá uma onda quadrada com o duty cycle (ciclo de trabalho) especificado até a próxima chamada de analogWrite() (ou uma chamada de digitalRead() ou digitalWrite() no mesmo pino). A frequência do sinal PWM na maioria dos pinos é aproximadamente 490 Hz. No Uno e placas similares, pinos 5 e 6 usam uma frequência de aproximadamente 980 Hz.

Sintaxe

analogWrite(pino, valor)

Parâmetros

pino: o pino escolhido do Arduino. Tipos de dados permitidos: int.
valor: o duty cycle: entre 0 (sempre desligado) and 255 (sempre ligado). Tipos de dados permitidos: int

Código de Exemplo

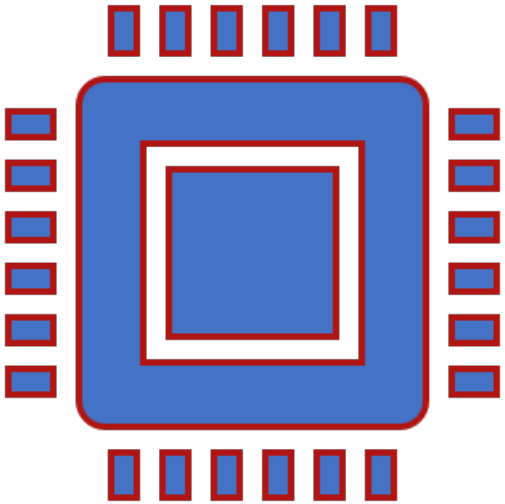
Controla a saída para um LED proporcionalmente a um valor lido de um potenciômetro.

```
int ledPin = 9;      // LED conectado ao pino digital 9
int analogPin = 3;   // potenciômetro conectado ao pino analógico 3
int val = 0;         // variável para guradar o valor lido

void setup() {
  pinMode(ledPin, OUTPUT);  // configura  o pino como saída
}

void loop() {
  val = analogRead(analogPin);  // lê o pino de entrada analógica
  analogWrite(ledPin, val / 4); // analogRead retorna valores de 0 a 1023, analogWrite recebe de 0 a 255
}
```

analogReference()



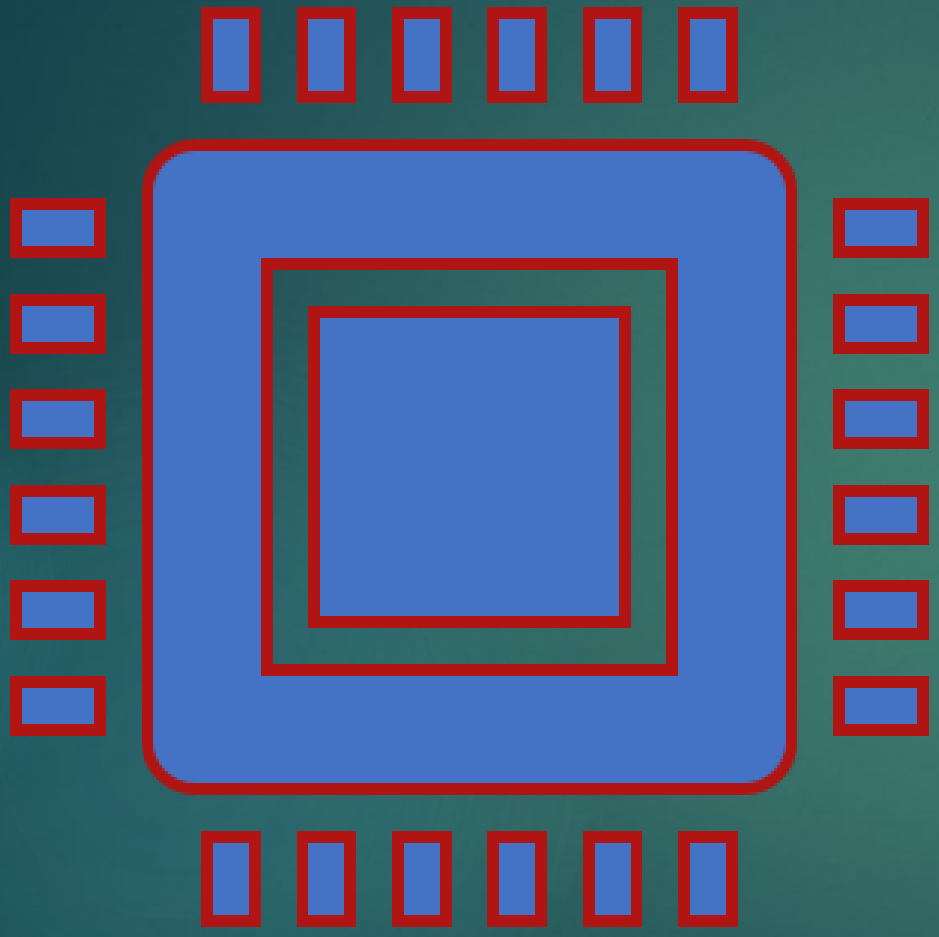
Configura a tensão de referência para a entrada analógica (o valor máximo do intervalo de entrada).

Sintaxe

analogReference(tipo)

Parâmetros
tipo: qual tipo de referência usar (DEFAULT, INTERNAL, INTERNAL1V1, INTERNAL2V56 ou EXTERNAL).

FUNÇÕES MATEMÁTICAS



abs()

constrain()

map()

max()

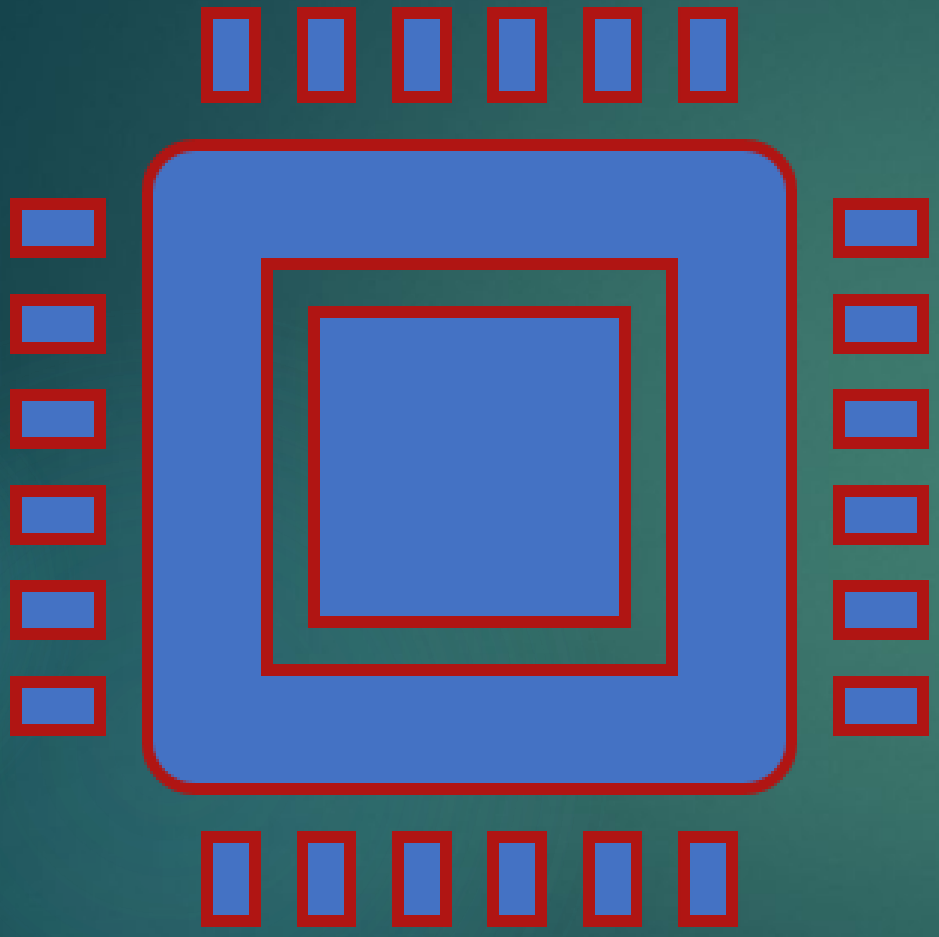
min()

pow()

sq()

sqrt()

FUNÇÕES TEMPORIZADORAS



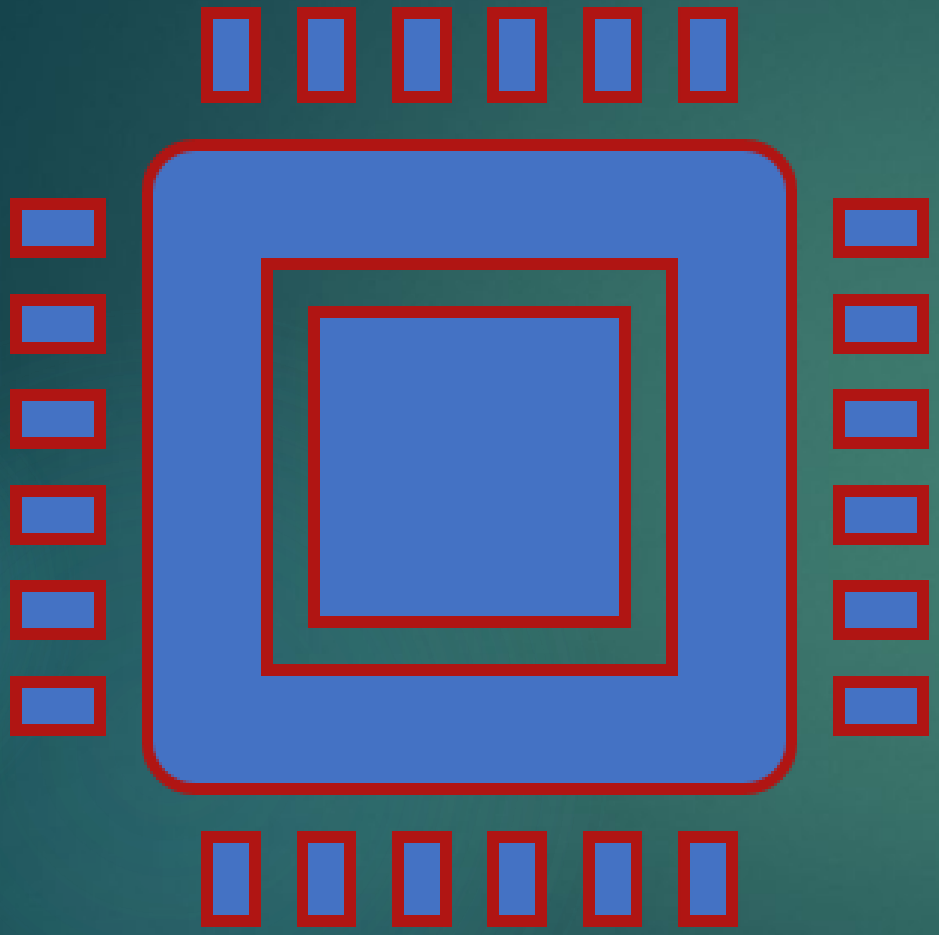
delay()

delayMicroseconds()

micros()

millis()

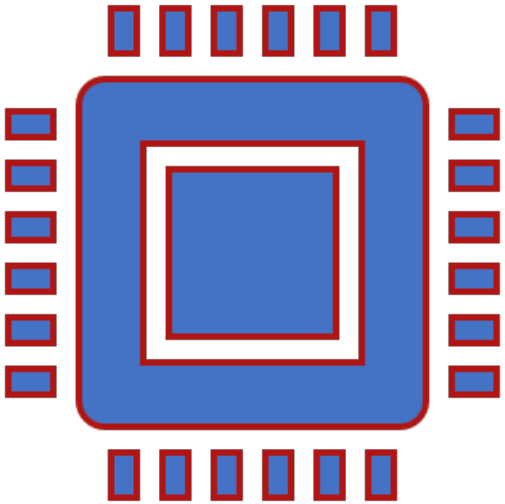
COMUNICAÇÃO



Serial

Stream

IF



O comando if checa uma condição e executas o comando a seguir ou um bloco de comandos delimitados por chavetas, se a condição é verdadeira ('true').

Sintaxe

```
if (condição) {
  //comando(s)
}
```

Código de Exemplo

As chaves podem ser omitidas depois de um comando if. Se isso é feito, a próxima linha (definida pelo ponto e vírgula) é interpretada como o único comando condicional. Para mais de um comando, use as chavetas para delimitar o bloco de comandos.

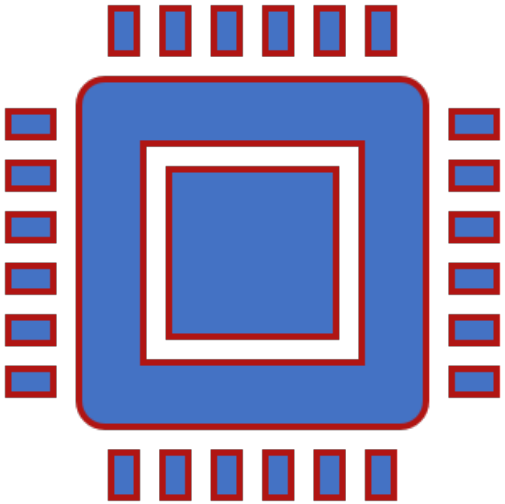
```
if (x > 120) {
  digitalWrite(pinoLED, HIGH);
}

if (x > 120) {
  digitalWrite(pinoLED, HIGH);
}

if (x > 120) {
  digitalWrite(pinoLED, HIGH);
}

if (x > 120) {
  digitalWrite(pinoLED1, HIGH);
  digitalWrite(pinoLED2, HIGH);
} // todas as formas acima estão corretas
```

ELSE



A combinação if…else permite maior controle sobre o fluxo de código que o comando mais básico if, por permitir múltiplos testes serem agrupados juntos. Uma cláusula else (se presente) será executada se a condição do comando if resulta em false. O else pode proceder outro teste if, tal que múltiplos, testes mutualmente exclusivos podem ser executados ao mesmo tempo.

Sintaxe

```
if (condição1) {
  // faz coisa A
}
else if (condição2) {
  // faz coisa B
}
else {
  // faz coisa C
}
```

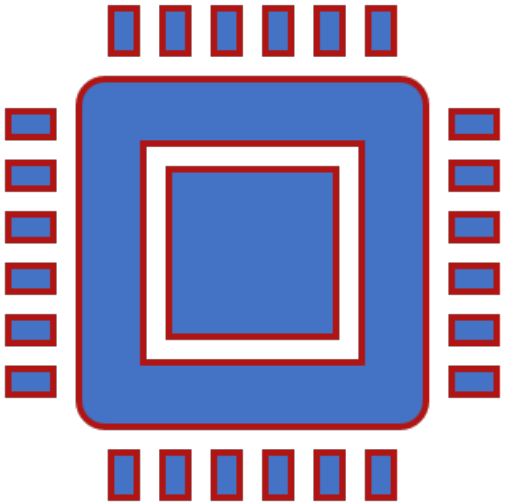
Código de Exemplo

Abaixo um trecho de código de um sistema de controle de temperatura

```
if (temperatura >= 70) {
  //Perigo! Desligar o sistema
}
else if (temperatura >= 60 && temperatura < 70) {
  //Cuidado! Requerida a atenção do usuário
}
else {
  //Seguro! Continue as tarefas usuais...
}
```

OPERADORES LÓGICOS

E - &&

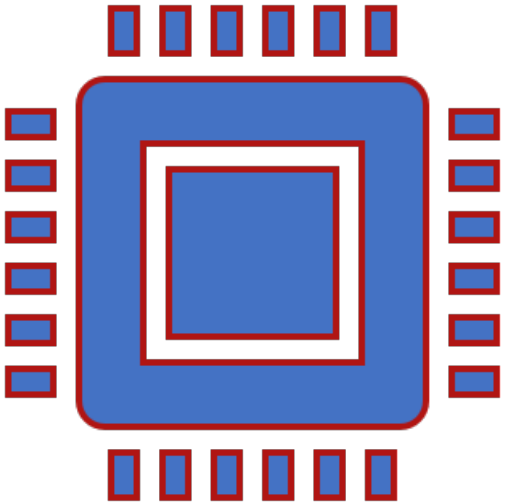


O E lógico resulta em verdadeiro, apenas se ambos os operandos são verdadeiros.

Código de Exemplo

```
if (digitalRead(2) == HIGH  && digitalRead(3) == HIGH) {  // se AMBOS os botões estão em HIGH
  // código a ser executado caso as duas condições sejam verdadeiras
}
```

Ou - | |

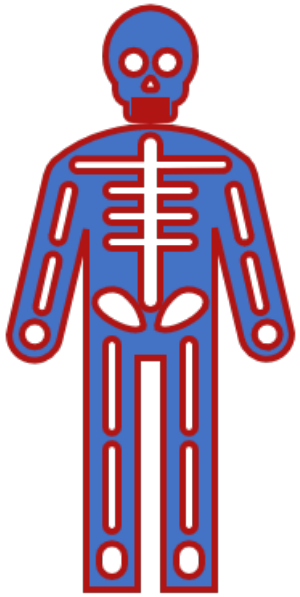


O OU lógico resulta em verdadeiro se pelo menos um dos operandos é verdadeiro

Código de Exemplo

```
if (x > 0 || y > 0) { // se x ou y é maior que zero
  // código a ser executado
}
```

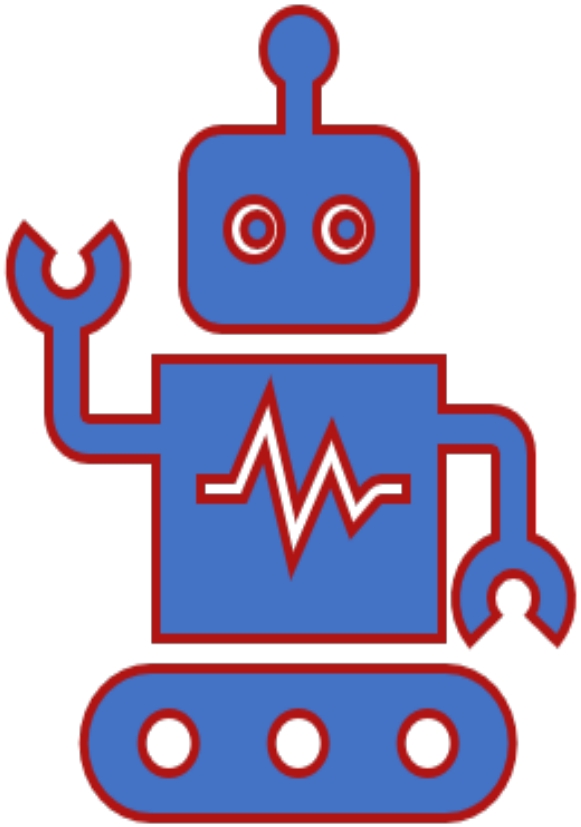
Desafio



Usando o TINKERCAD, construa um robô que de ordem de marcha a dois motores CC quando este não se encontre em aquecimento e não encontre nenhum obstáculo pelo seu caminho.
Deverá colocar um sensor de movimento para o lado direito e para o lado esquerdo da máquina.
Deverá colocar um motor do lado esquerdo e do lado direito da máquina
O motor Esq deverá parar de rolar quando é detetado presença de objeto do lado direito da máquina.
O motor Drt deverá parar de rolar quando é detetado presença de objeto do lado esquerdo da máquina.

Bom Trabalho!

Programação



Co-funded by the Erasmus+ Programme of the European Union

The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.

Worksheet No. 1

Learning Unit	**Aprender C++ com robôs**
Authors	Nuno Barbosa
School	FORAVE – ASSOCIAÇÃO PARA A EDUCAÇÃO TECNOLÓGICA DO VALE DO AVE
Date	

1 – Faça um script em C++ para adquirir o estado da bateria do Botn'Roll.

2 - Faça um script em C++ para que o Botn'Roll avance em linha reta durante 5 segundos.

3 - Faça um script em C++ para que o Botn'Roll avance em linha reta durante 5 segundos e que depois recue durante mais 5 segundos.

4 - Faça um script em C++ para que o Botn´Roll se desvie de obstáculos usando os sensores de Infravermelhos.

5 - Faça um script em C++ para que o Botn´Roll siga uma linha preta em fundo branco usando o sensor seguidor de linha.

Co-funded by the Erasmus+ Programme of the European Union

The European Commission support for the production of this publication does not constitute an endorsement of the contents which reflects the views only of the authors, and the Commission cannot be held responsible for any use which may be made of the information contained therein.