Curso de Arduino

Sessão 2

EPS YOUNG MINDS AVEIRO

NeRD aeTua

Na primeira sessão

- ✓**Configuração** dos Portos
- ✓Leitura e Escrita no domínio **Digital**
- ✓Leitura e Escrita no domínio **Analógico**
- ✓**Ferramentas** úteis
- ✓**Programação** em C
- ✓Montagem de **circuitos**
  - ✓LEDs
  - ✓LED RGB
  - ✓Botões
  - ✓Potenciómetro

❑ Configuração:

```
pinMode(pin, value);
```

❑ Útil:

```
delay(value);
Serial.begin(speed);
Serial.print(value);
Serial.println(value);
```

❑ Leitura/Escrita:

```
digitalRead(pin);
digitalWrite(pin, value);
analogRead(pin);
analogWrite(pin, value);
```

arduino.cc/reference

Esta sessão


iPod
Music
Videos
Photos
Podcasts
Extras
Settings
Shuffle Songs
Now Playing
MENU


CREATIVE
NOMAD
MuVo 128MB


SONY
CD WALKMAN

Esta sessão

Hardware

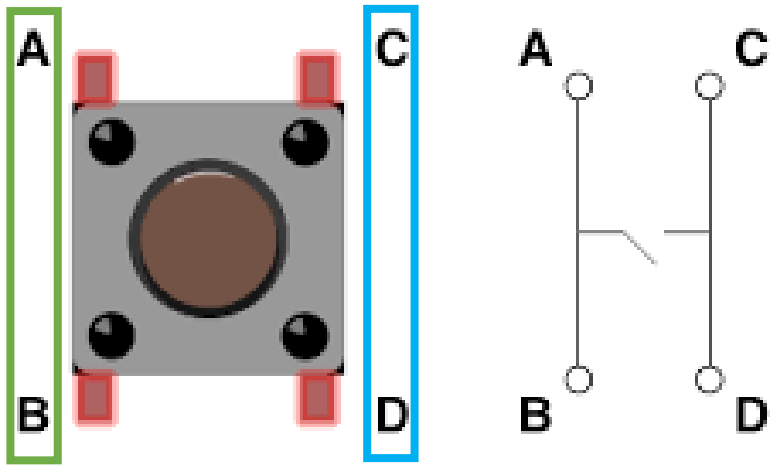
- Botões
- *Buzzer* passivo
- LCD 16x2

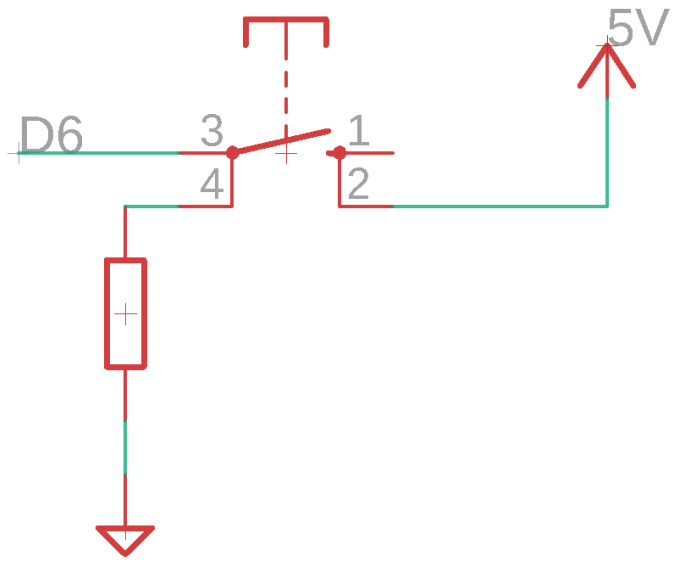
Software

- Máquina de Estados
- Ferramentas da linguagem C

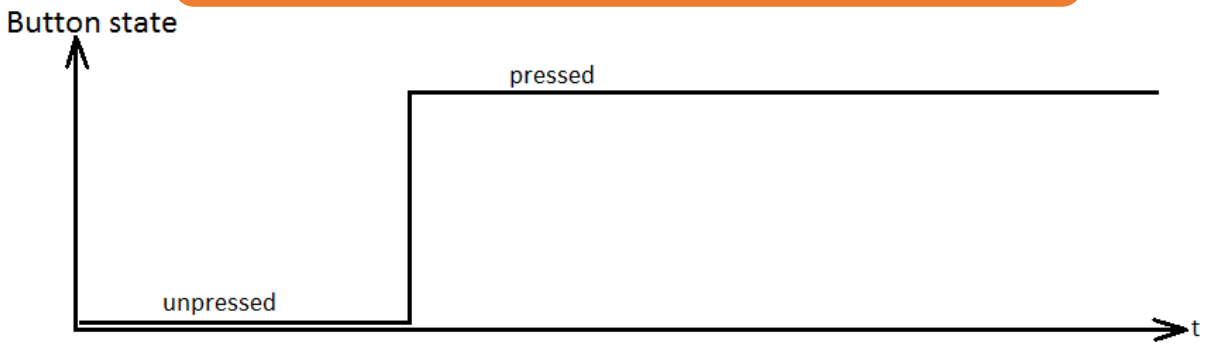
Hardware

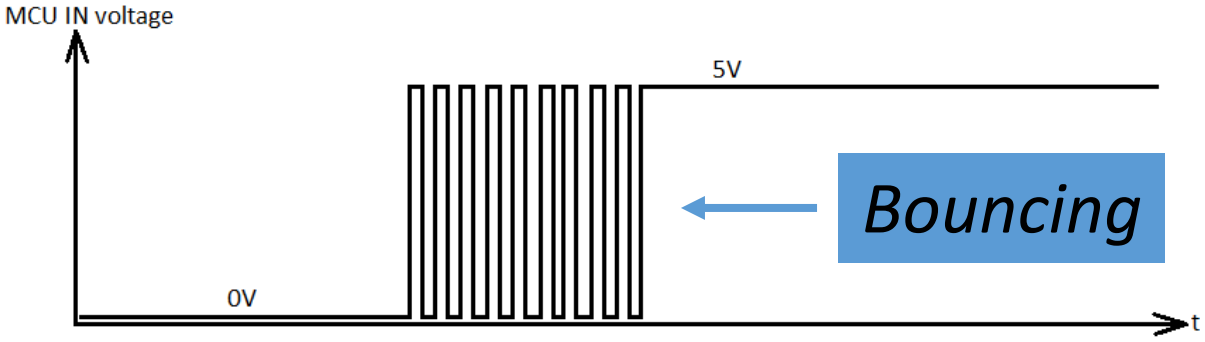
Botões





Queremos detetar o **evento** do estado do botão passar **de LOW para HIGH**. Como?





Exemplo de implementação de um **debouncer** no Arduino IDE:
Ficheiro>Exemplos>02.Digital>Debounce

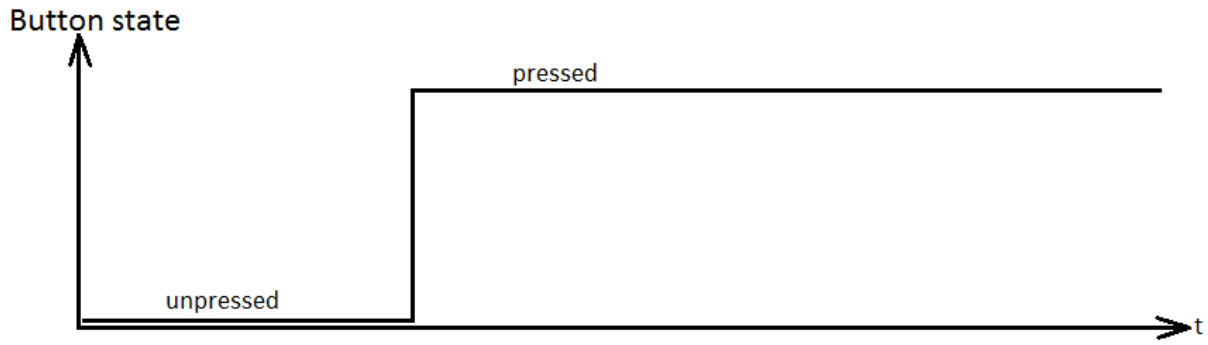
Botões

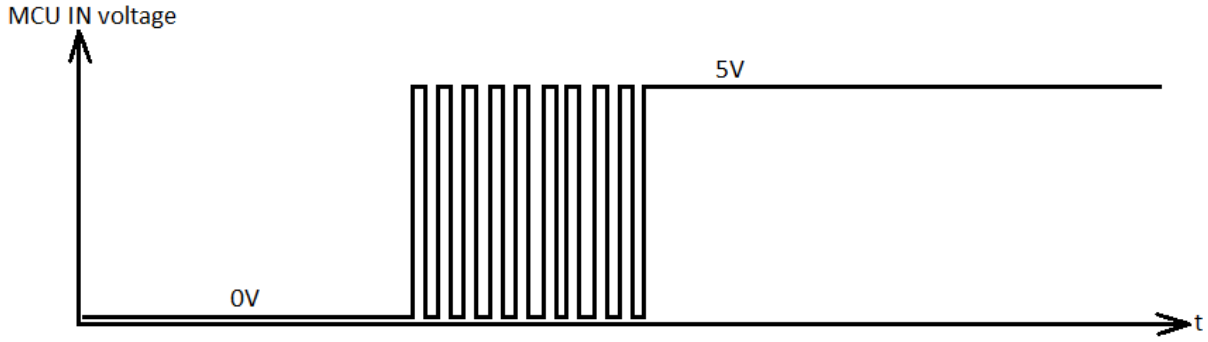
Utilização:

```
int prev = digitalRead(6);
int stop = digitalRead(7);
int next = digitalRead(8);
```

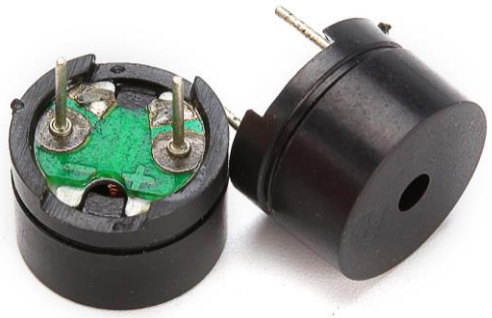
```
readButtons();

char stopFlag;
char nextFlag;
char prevFlag;
```



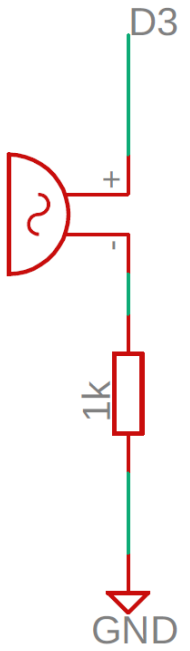


Buzzer passivo



Buzzer ativo >> Toca quando se aplica uma determinada **tensão DC**

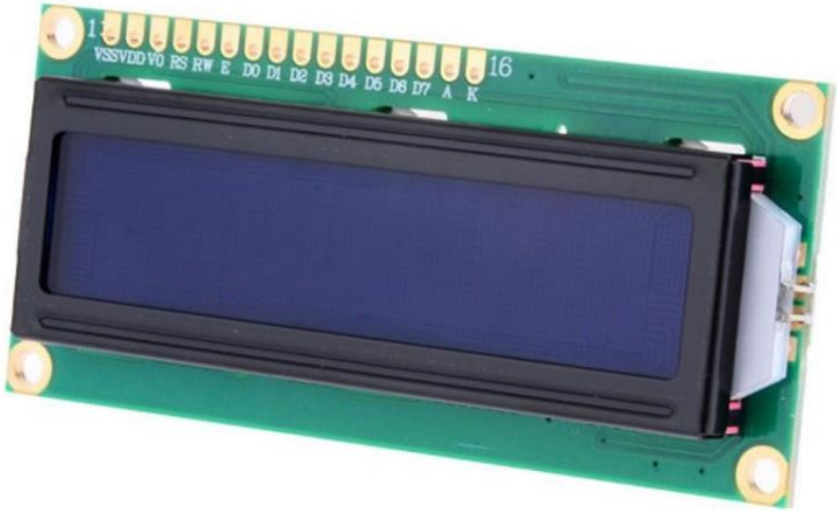
Buzzer passivo >> Toca quando se aplica uma onda com uma **determinada frequência**

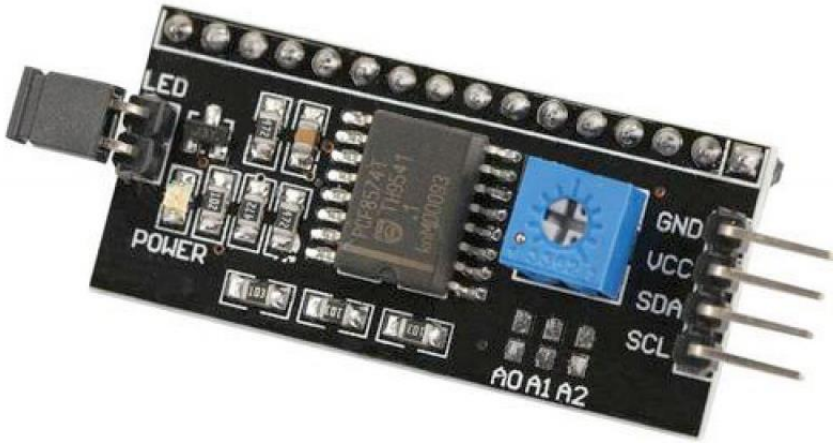


Utilização:

```
tone(pin, frequency);
noTone(pin);
```

LCD 16x2



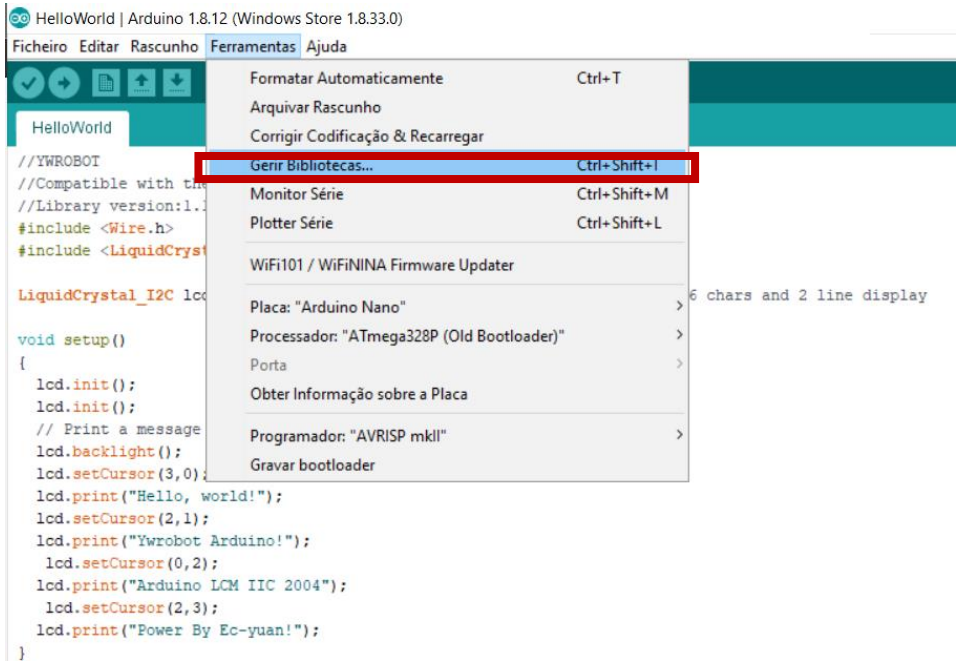


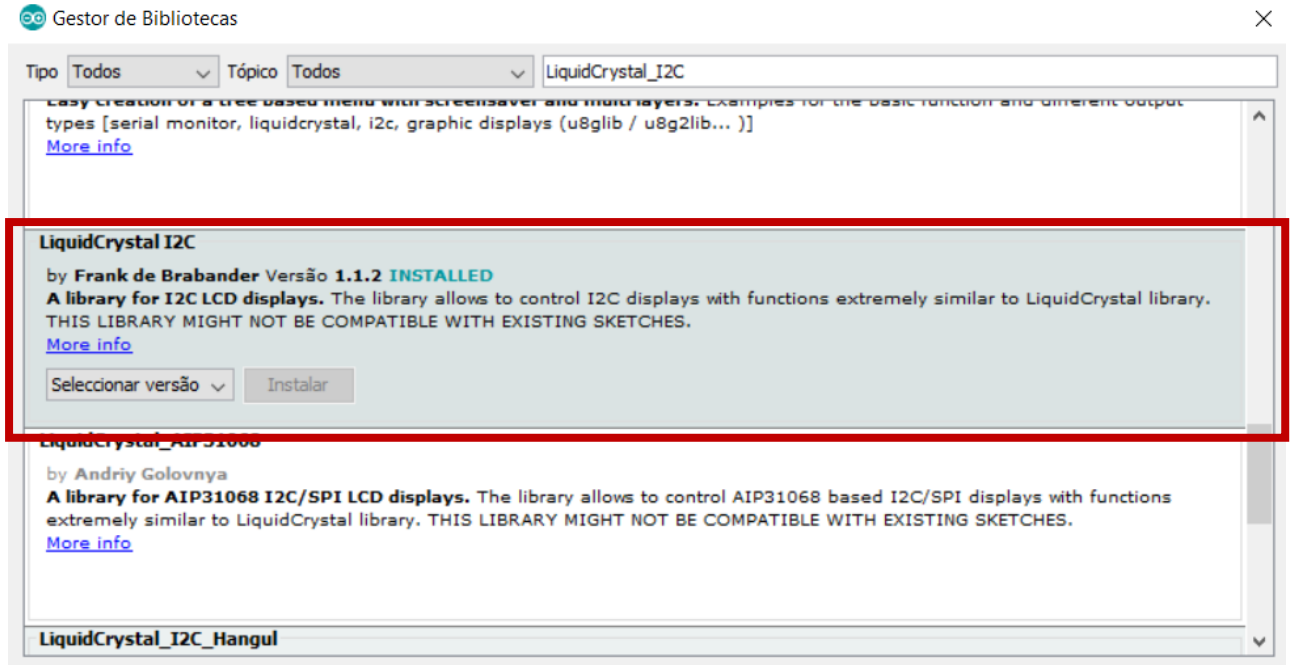
Utilização da biblioteca

LiquidCrystal_I2C

Biblioteca *LiquidCrystal_I2C*

Instalação





github.com/johnrickman/LiquidCrystal_I2C

Biblioteca *LiquidCrystal_I2C*

1. **Incluir a biblioteca**

```
#include <LiquidCrystal_I2C.h>
```

2. **Criar um objeto**

```
LiquidCrystal_I2C lcd(0x27,16,2);
```

(endereço, nº de colunas, nº de linhas)

3. **Utilizar as funções**

Biblioteca *LiquidCrystal_I2C*

Funções mais utilizadas:

```
lcd.init();
lcd.backlight();
lcd.noBacklight();
lcd.clear();
lcd.setCursor(column, line);
lcd.print(value);
```

Mais funcionalidades da biblioteca:
github.com/johnrickman/LiquidCrystal_I2C

Software

Diagrama de Estados

Diagrama de Estados

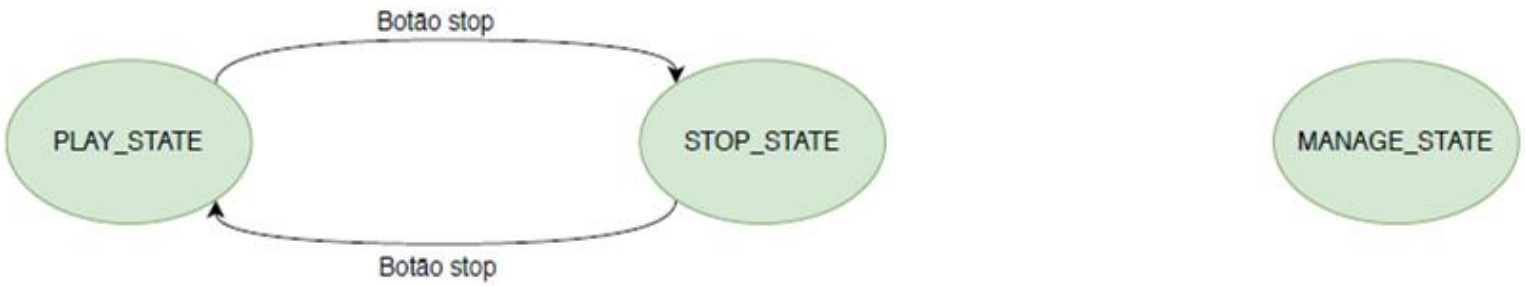


Diagrama de Estados

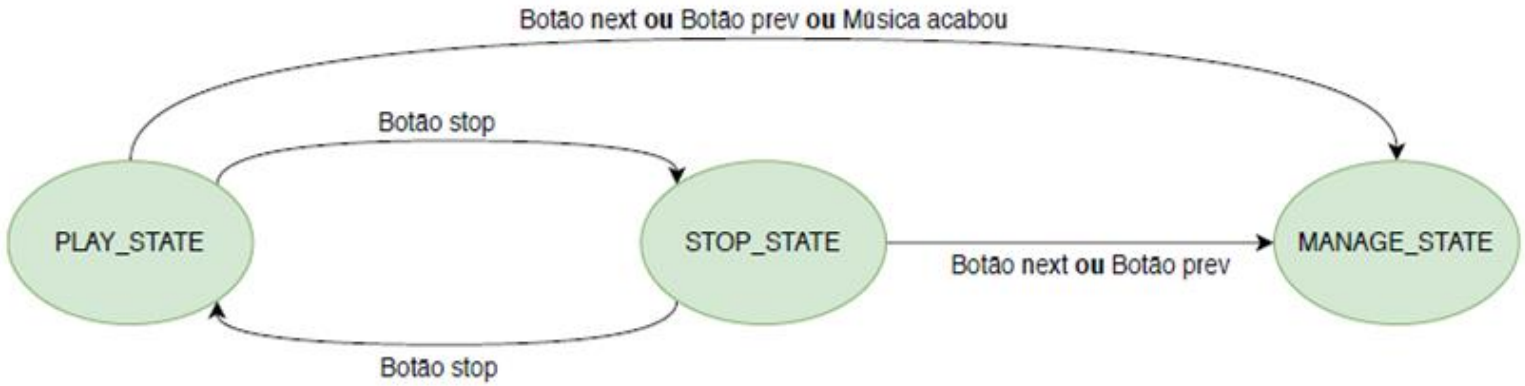


Diagrama de Estados

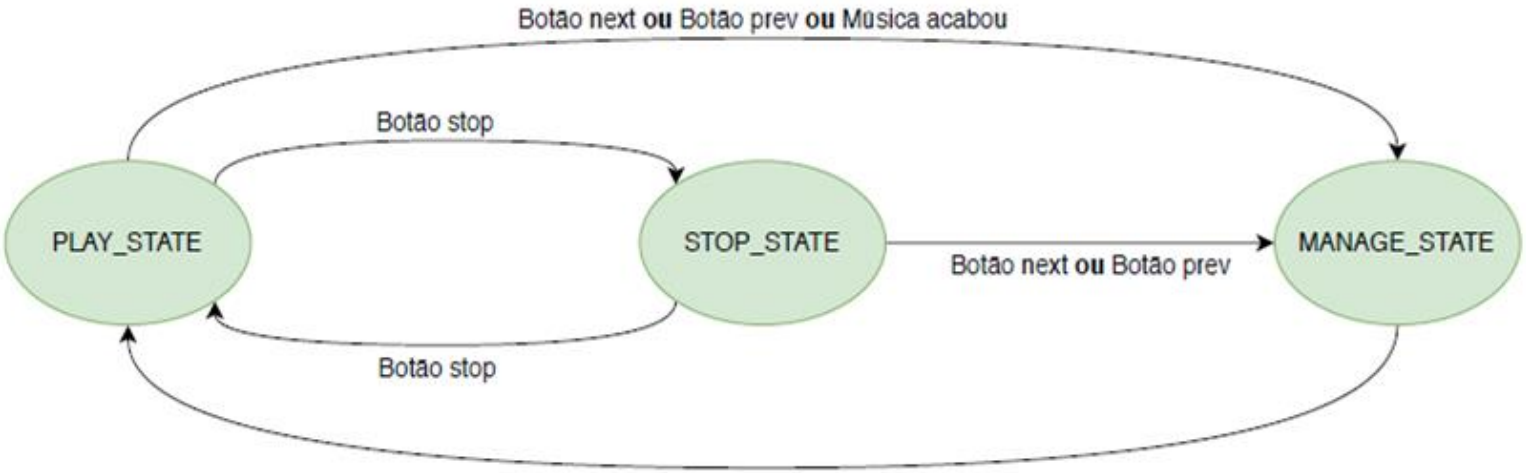
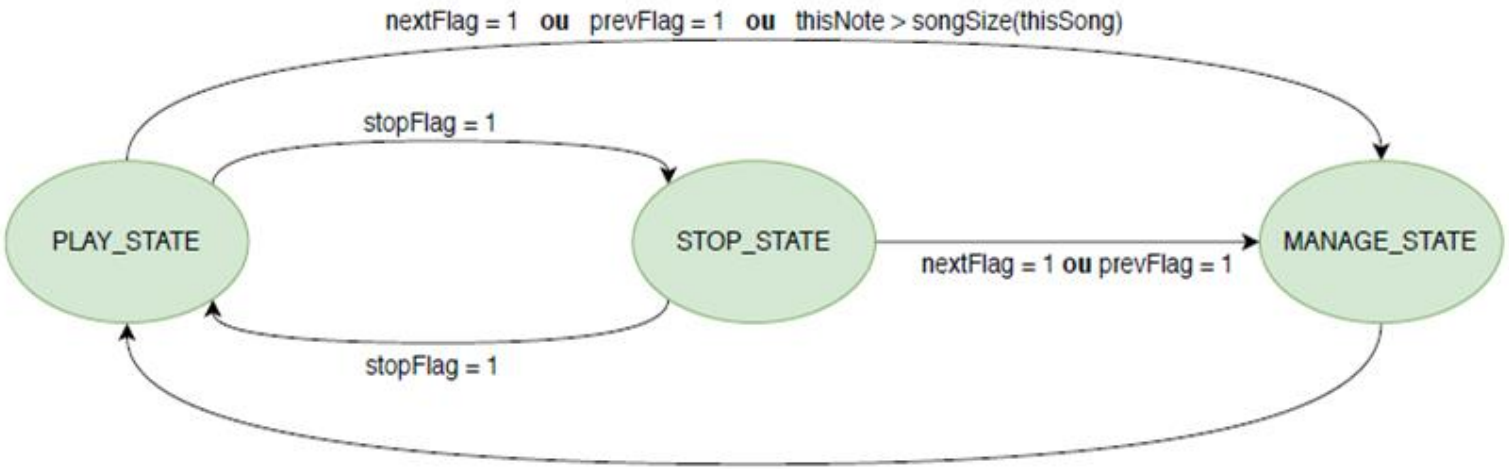


Diagrama de Estados



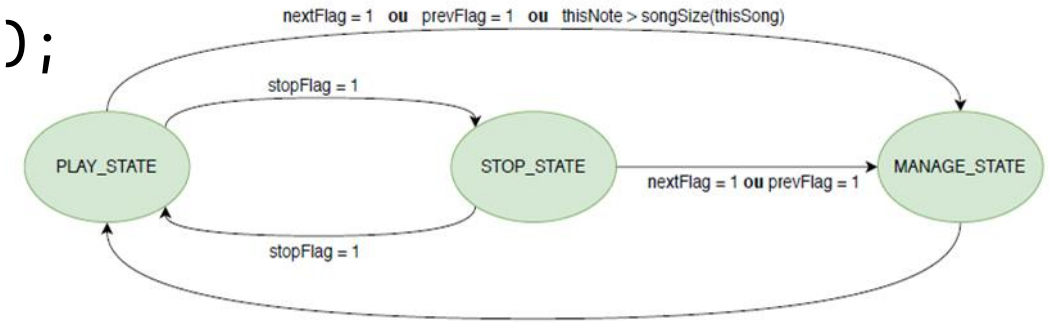
Biblioteca *music.h*

```
void readButtons();
int songSize(int song);
void play(int song, int note);
String getName(int song);
void changeTempo(int song, int tempo);
```

```
char stopFlag;
char nextFlag;
char prevFlag;

int thisNote;
int thisSong;

char state;
```



Implementação

```
void loop() {

  readButtons();

  switch(state) {
    case PLAY_STATE:

      break;

    case STOP_STATE:

      break;

    case MANAGE_STATE:

      break;
  }

}
```

Variável de estado

```
if (state == PLAY_STATE) {
    // código
}
else if (state == STOP_STATE) {
    // código
}
else if (state == MANAGE_STATE) {
    // código
}
```

Implementação

```
void loop() {

  readButtons();

  switch(state) {
    case PLAY_STATE:
      play(thisSong,thisNote);
      thisNote++;
      if(thisNote > songSize(thisSong)) state = MANAGE_STATE;
      break;

    case STOP_STATE:
      /* código */
      break;

    case MANAGE_STATE:




      break;
  }

}
```

Implementação

```
void loop() {

  readButtons();

  switch(state) {
    case PLAY_STATE:
      play(thisSong,thisNote);
      thisNote++;
      if(thisNote > songSize(thisSong)) state = MANAGE_STATE;
      break;

    case STOP_STATE:
      /* código */
      break;

    case MANAGE_STATE:
      thisSong++;
      if(thisSong == NUMBER_OF_SONGS) thisSong = 0;
      thisNote = 0;
      state = PLAY_STATE;
      lcd.clear();
      lcd.setCursor(0,0);
      lcd.print("Faixa ");
      lcd.print(thisSong+1);
      delay(200);
      break;
  }

}
```

Mais uma ferramenta

```
if(a > 1 || b > 3) {
  //código
}
```

```
if(a > 1 && b > 3) {
  //código
}
```

Qual é a diferença?

&& é um **E** lógico

|| é um **OU** lógico