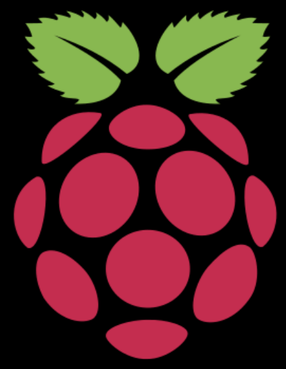


Projeto 01

Controle de Mídia – Teoria

Jan K. S. – janks@puc-rio.br

ENG 1419 – Programação de Microcontroladores

Hardware

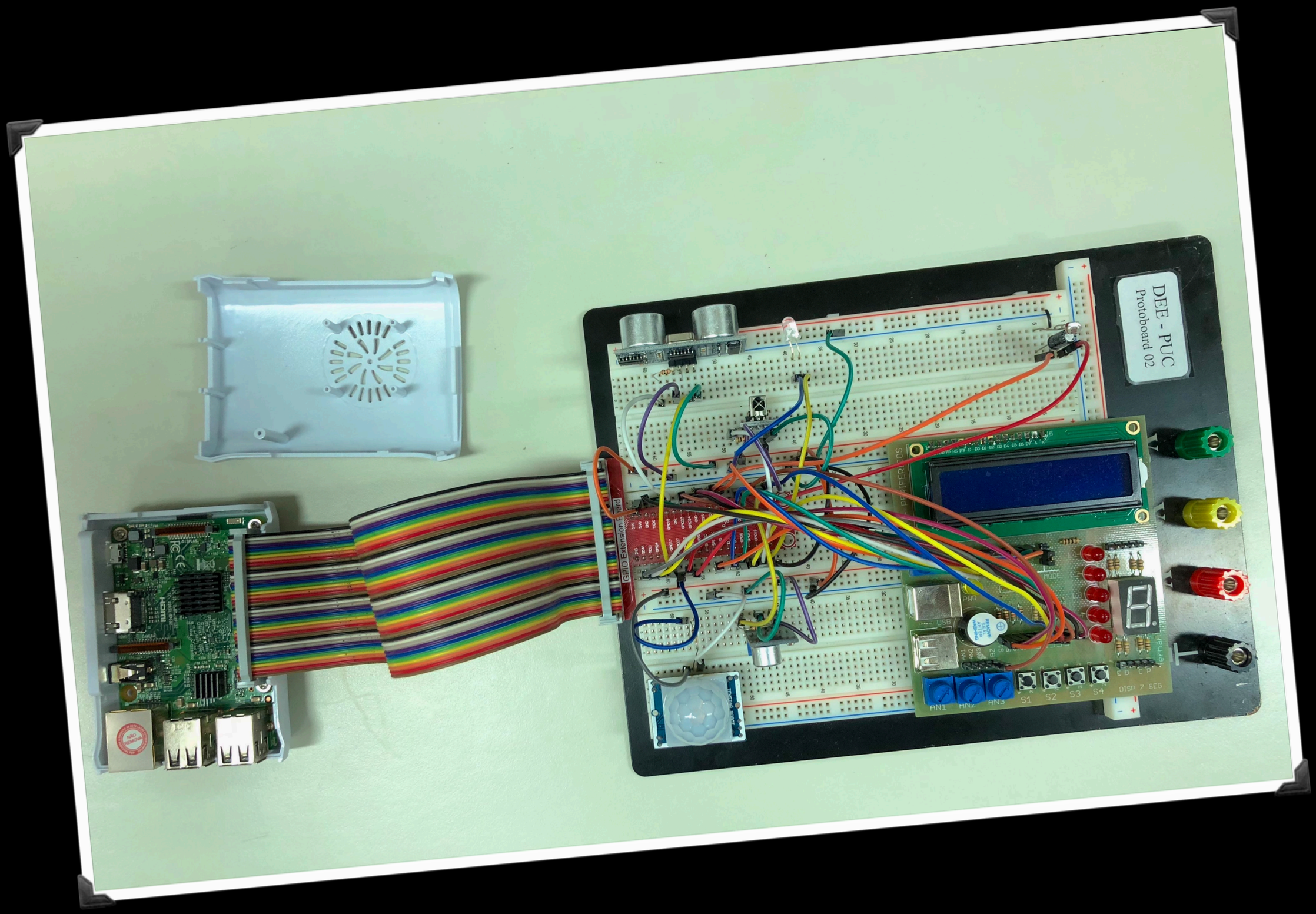
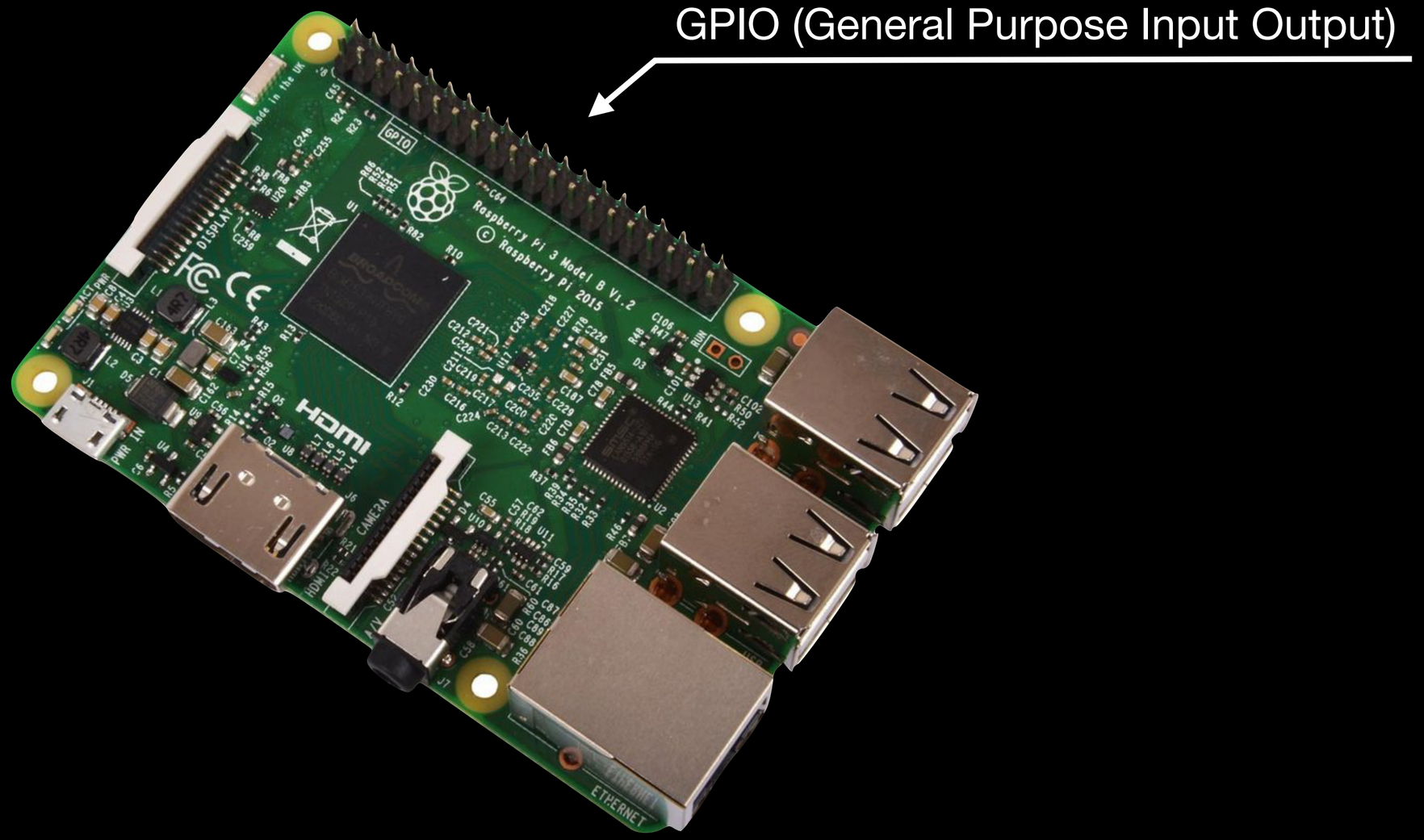
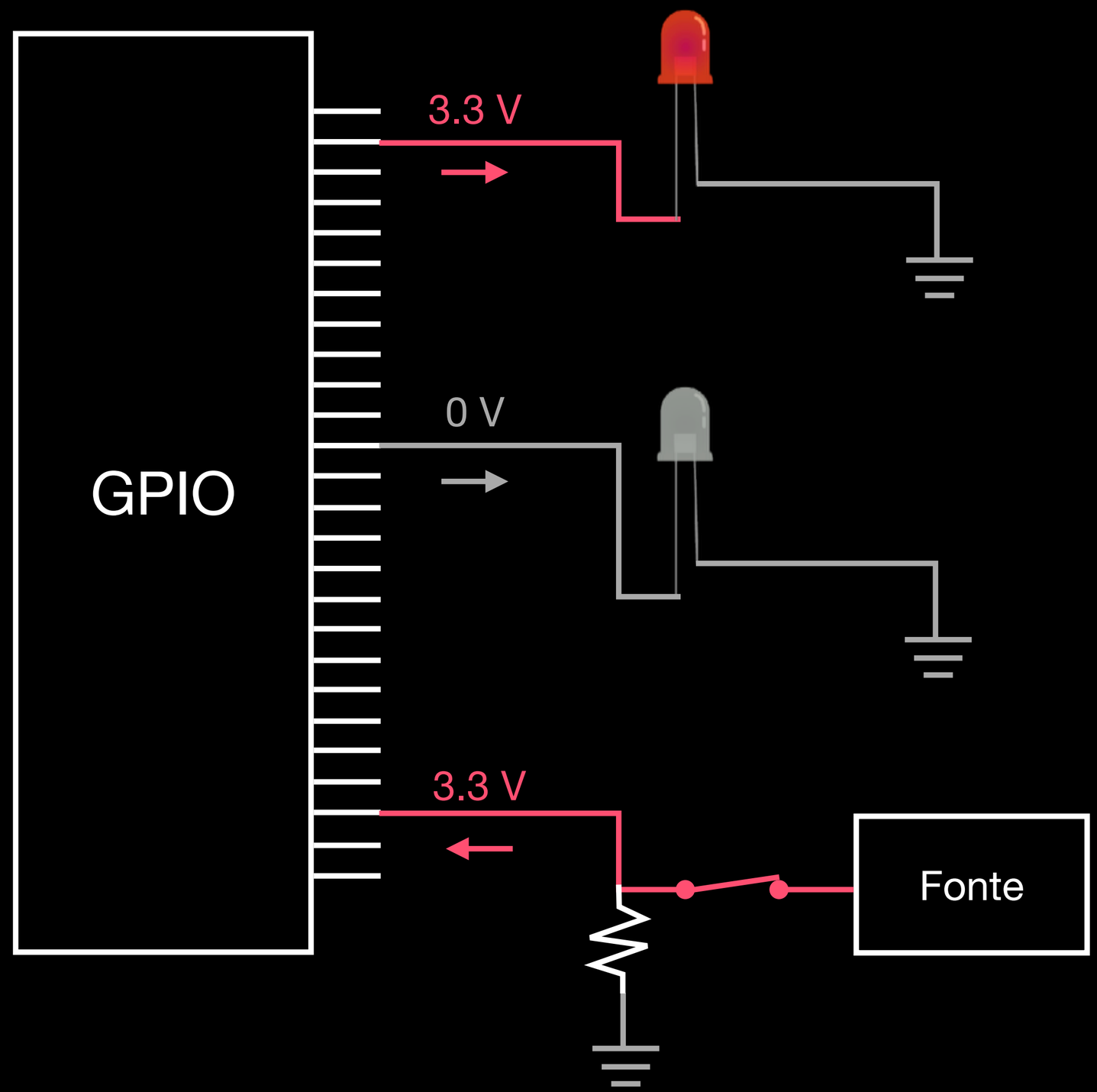


Foto do Equipamento Usado em Sala



GPIO do Raspberry Pi

```
>>> ligue a porta 2

>>> desligue a porta 12

>>> leia a porta 25
```



Controle Digital Através da GPIO

3.3 V → nivel lógico 1
HIGH

0 V → nivel lógico 0
LOW

Níveis Lógicos em Eletrônica Digital

Raspberry Pi 3 GPIO Header

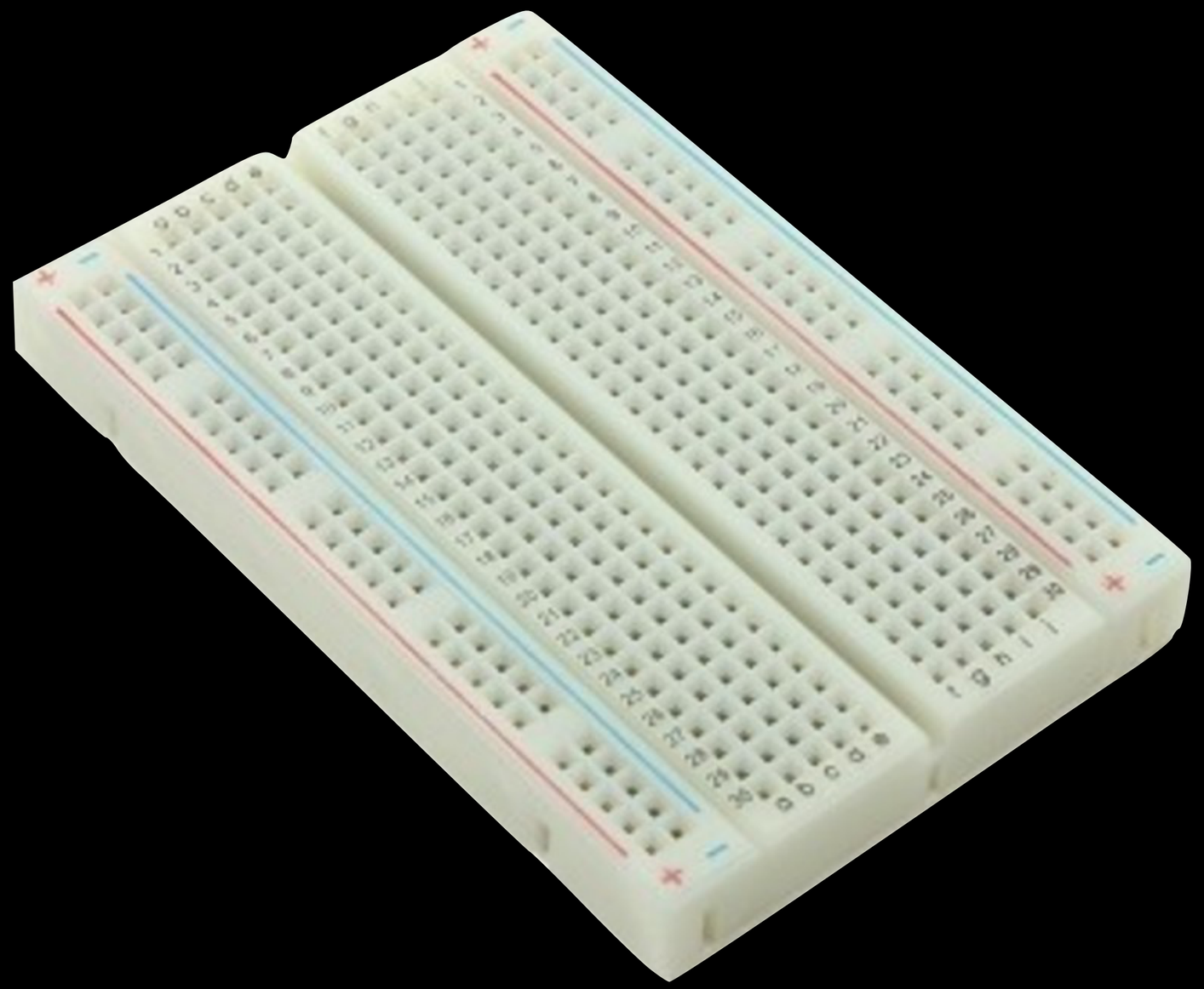
Pin#	NAME		NAME	Pin#
01	3.3v DC Power		DC Power 5v	02
03	GPIO02 (SDA1 , I²C)		DC Power 5v	04
05	GPIO03 (SCL1 , I²C)		Ground	06
07	GPIO04 (GPIO_GCLK)		(TXD0) GPIO14	08
09	Ground		(RXD0) GPIO15	10
11	GPIO17 (GPIO_GEN0)		(GPIO_GEN1) GPIO18	12
13	GPIO27 (GPIO_GEN2)		Ground	14
15	GPIO22 (GPIO_GEN3)		(GPIO_GEN4) GPIO23	16
17	3.3v DC Power		(GPIO_GEN5) GPIO24	18
19	GPIO10 (SPI_MOSI)		Ground	20
21	GPIO09 (SPI_MISO)		(GPIO_GEN6) GPIO25	22
23	GPIO11 (SPI_CLK)		(SPI_CE0_N) GPIO08	24
25	Ground		(SPI_CE1_N) GPIO07	26
27	ID_SD (I²C ID EEPROM)		(I²C ID EEPROM) ID_SC	28
29	GPIO05		Ground	30
31	GPIO06		GPIO12	32
33	GPIO13		Ground	34
35	GPIO19		GPIO16	36
37	GPIO26		GPIO20	38
39	Ground		GPIO21	40

Porta Reservada (Indisponível)

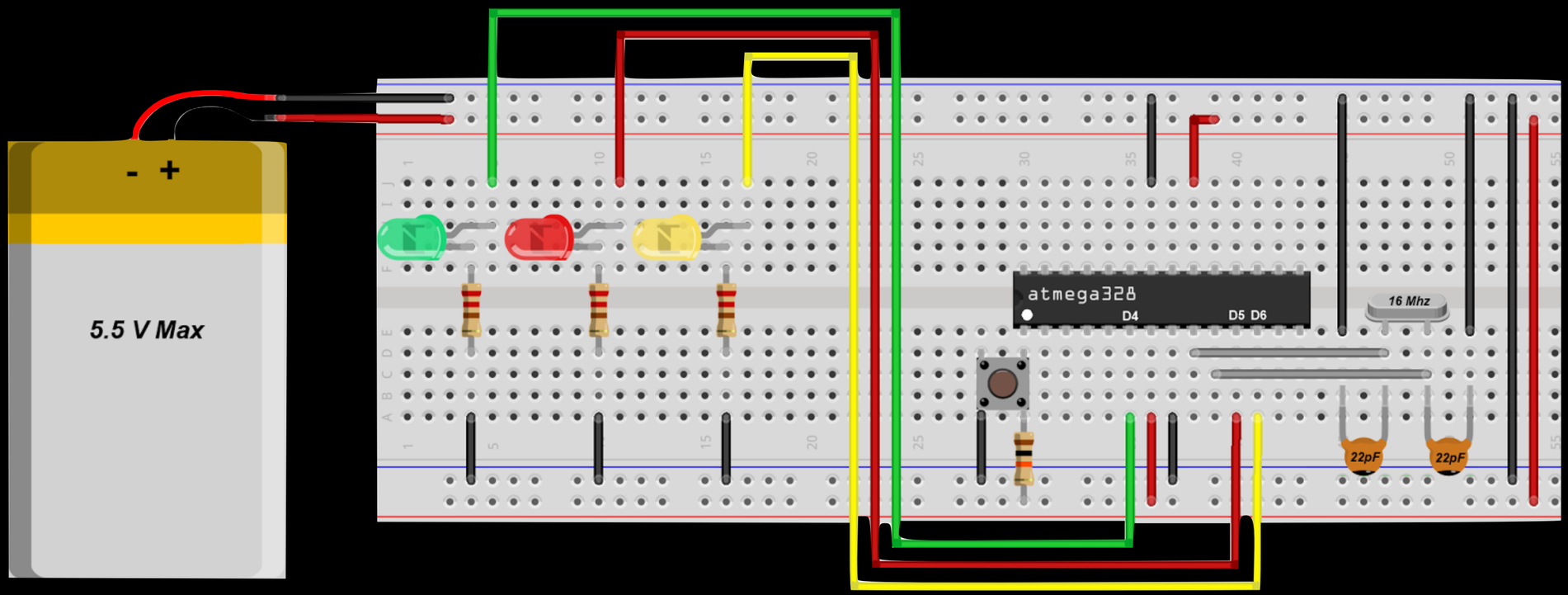
Porta Disponível

Porta Disponível com Propriedades Extras

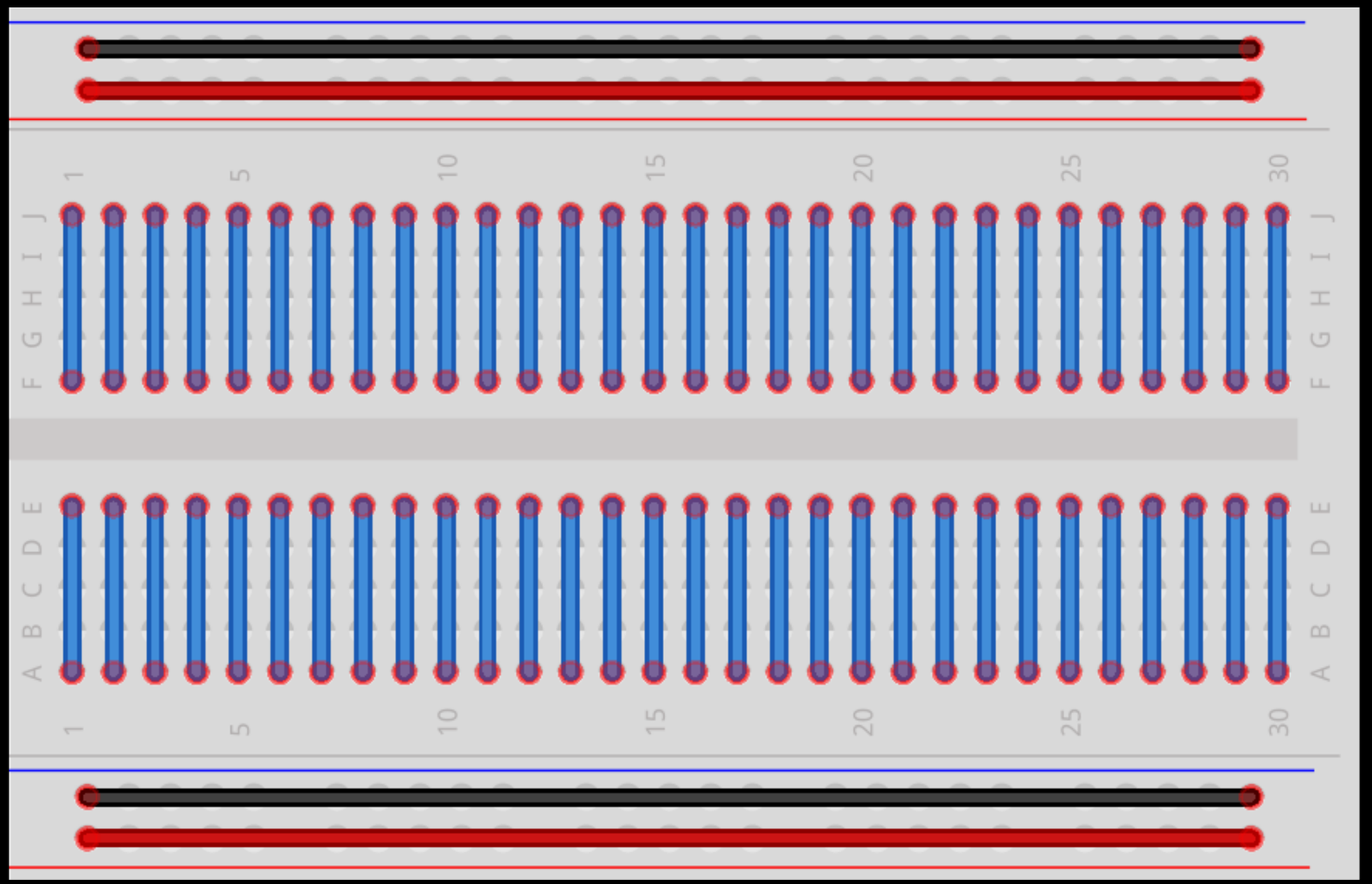
Identificação das Portas da GPIO



Protoboard

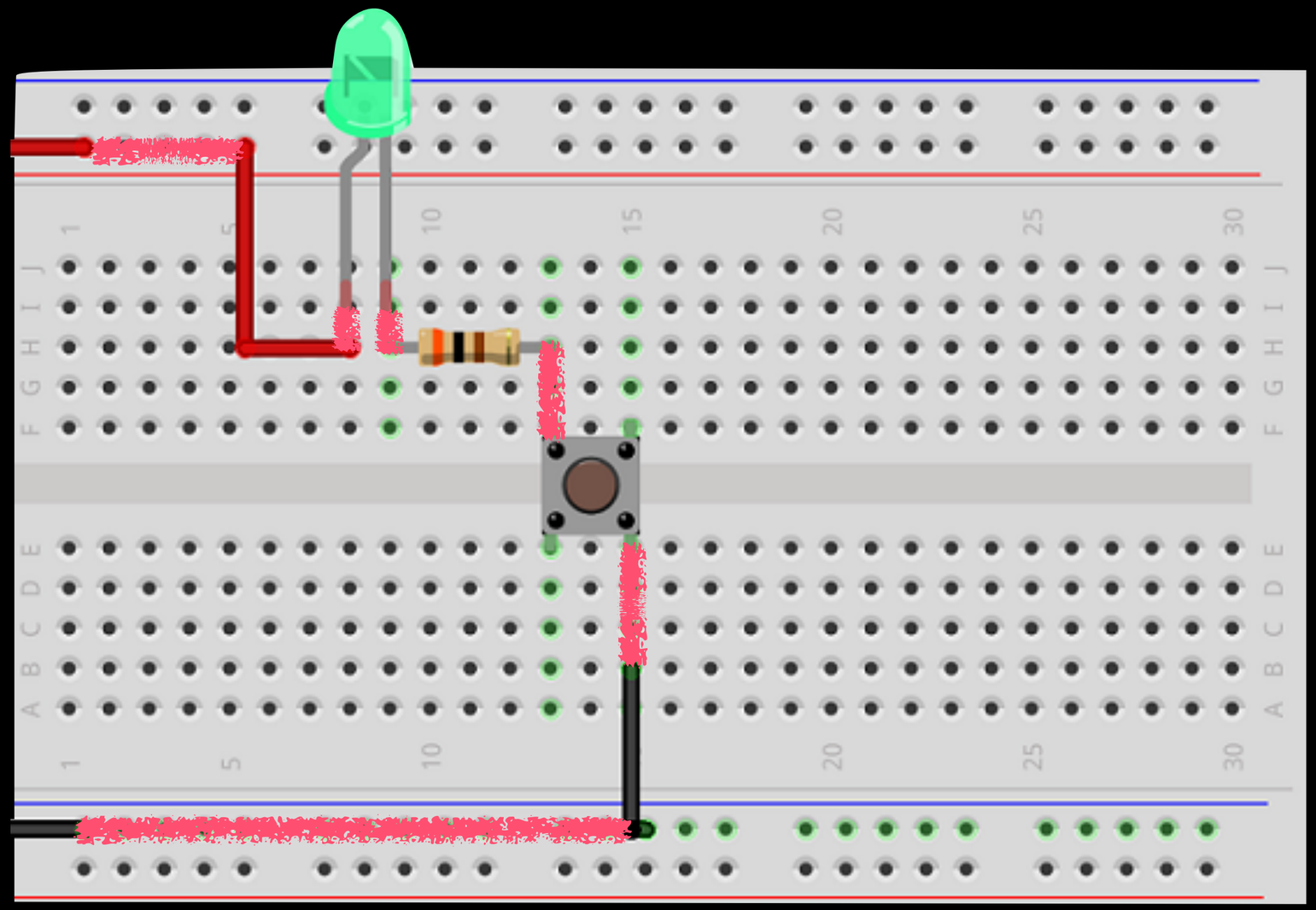


Componentes Eletrônicos em uma Protoboard



Conexões Internas da Protoboard

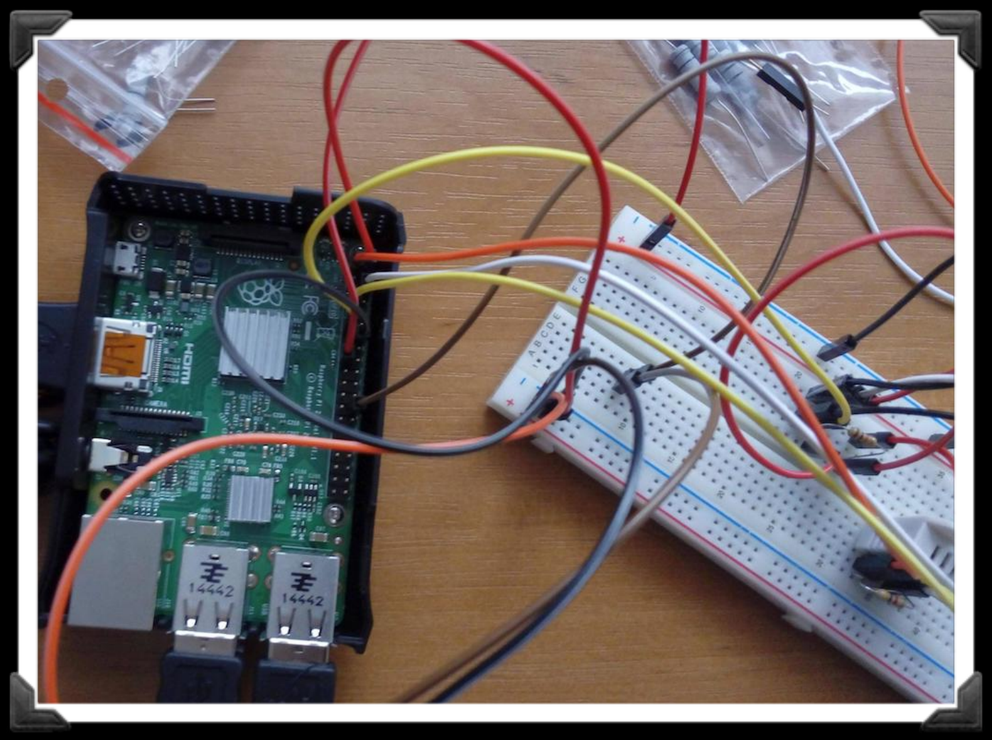
5V

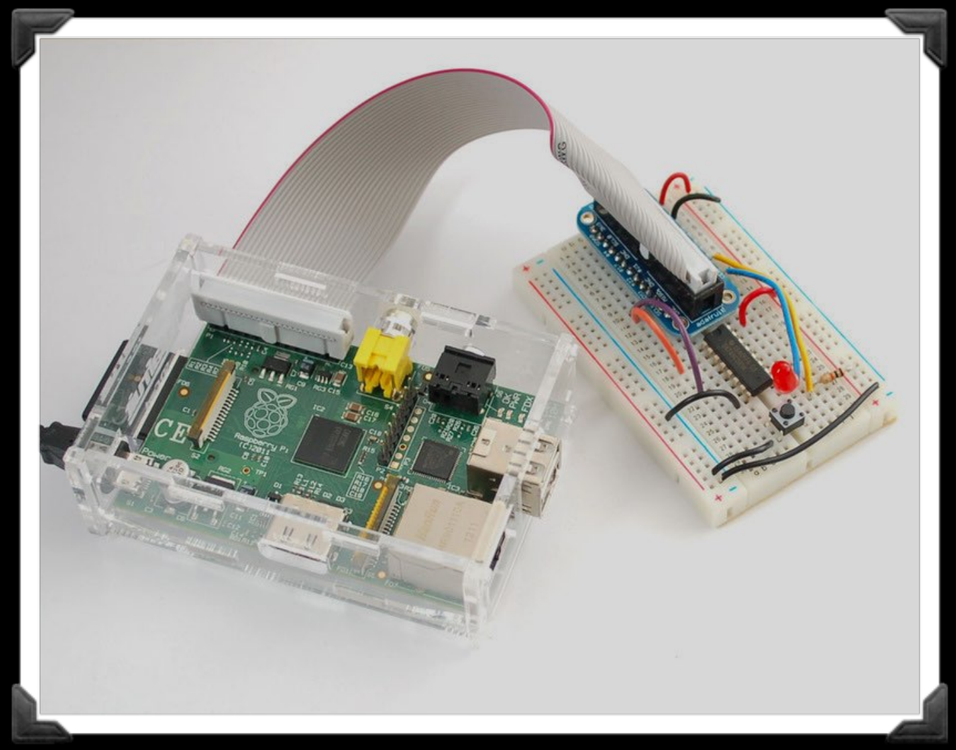


Terra

Componentes Conectados Internamente

Cabo "Flat" com Identificação





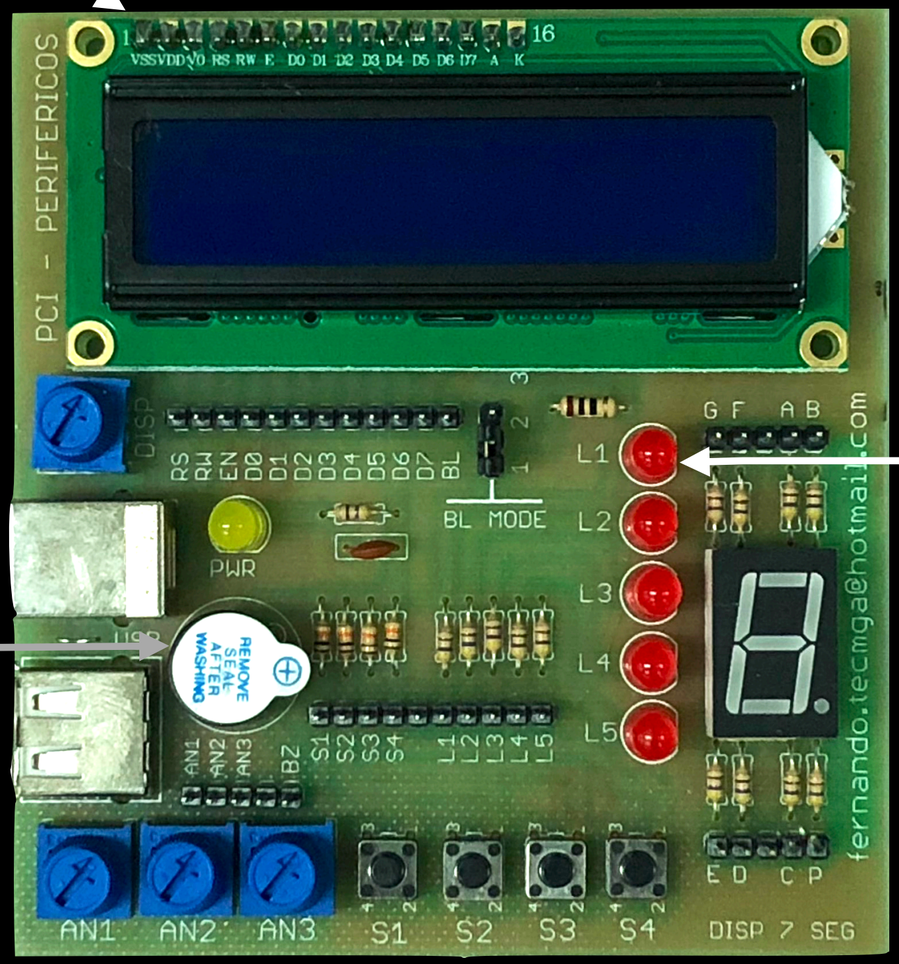
Separação entre Microcontrolador e Circuito

LCD de Caracteres

Alimentação

Campainha

Chaves Analógicas



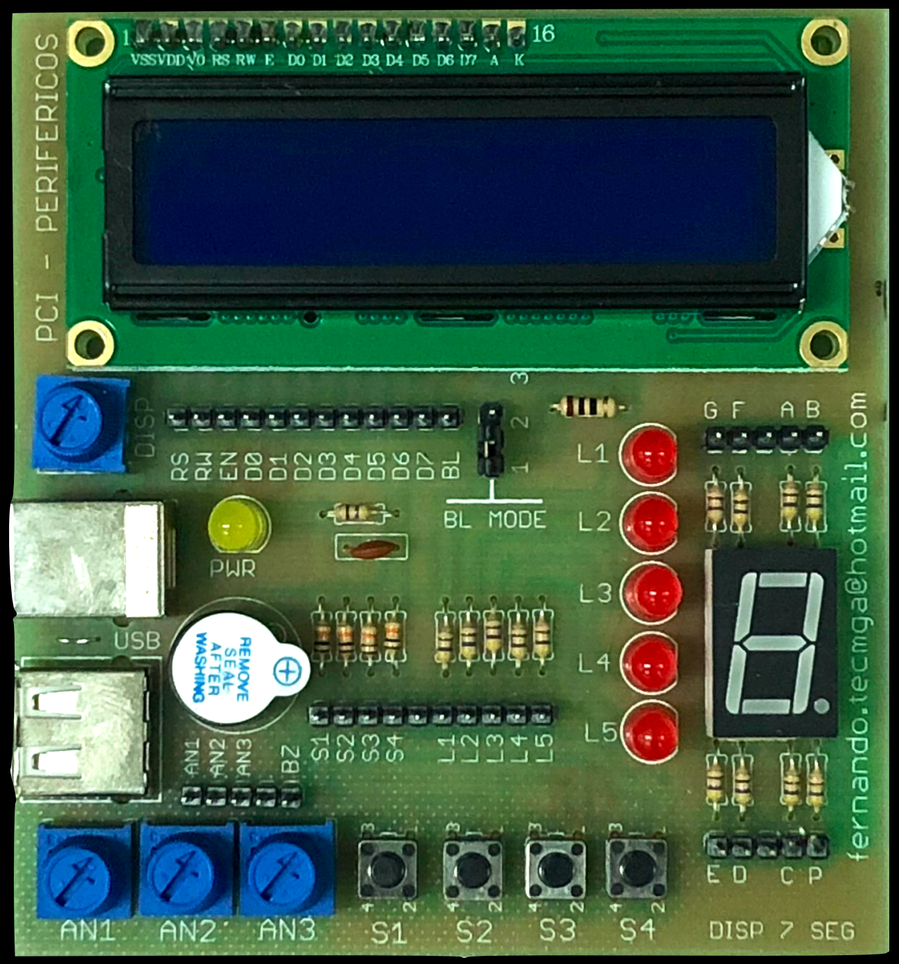
LEDs

Display de 7 Segmentos

Botões Digitais

Placa de Dispositivos Básicos

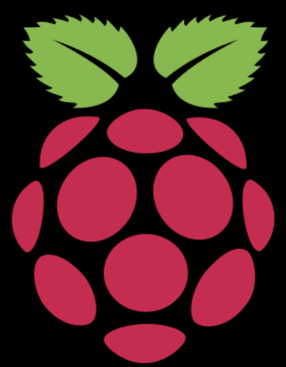
GPIO 2, 3, 4, 5, 6, 7



GPIO 21
GPIO 22
GPIO 23
GPIO 24
GPIO 25

GPIO 11, 12, 13, 14

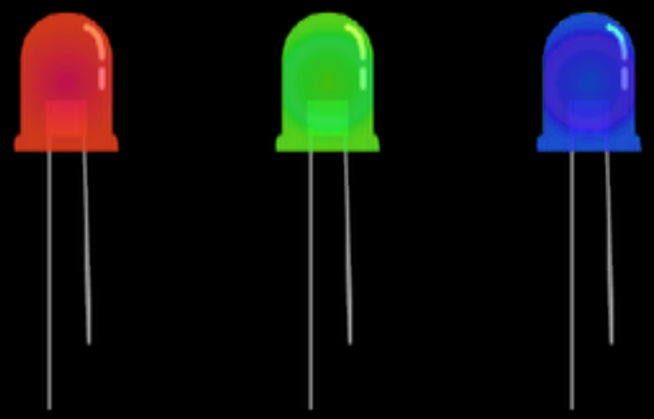
Conexões dos Botões, LEDs e LCD de Caracters com as Portas da GPIO



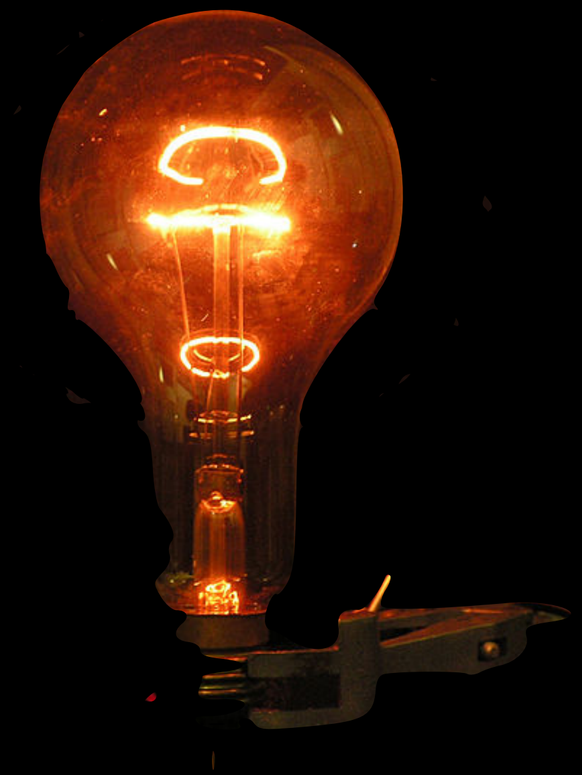
gpiozero

Adafruit_Python_CharLCD

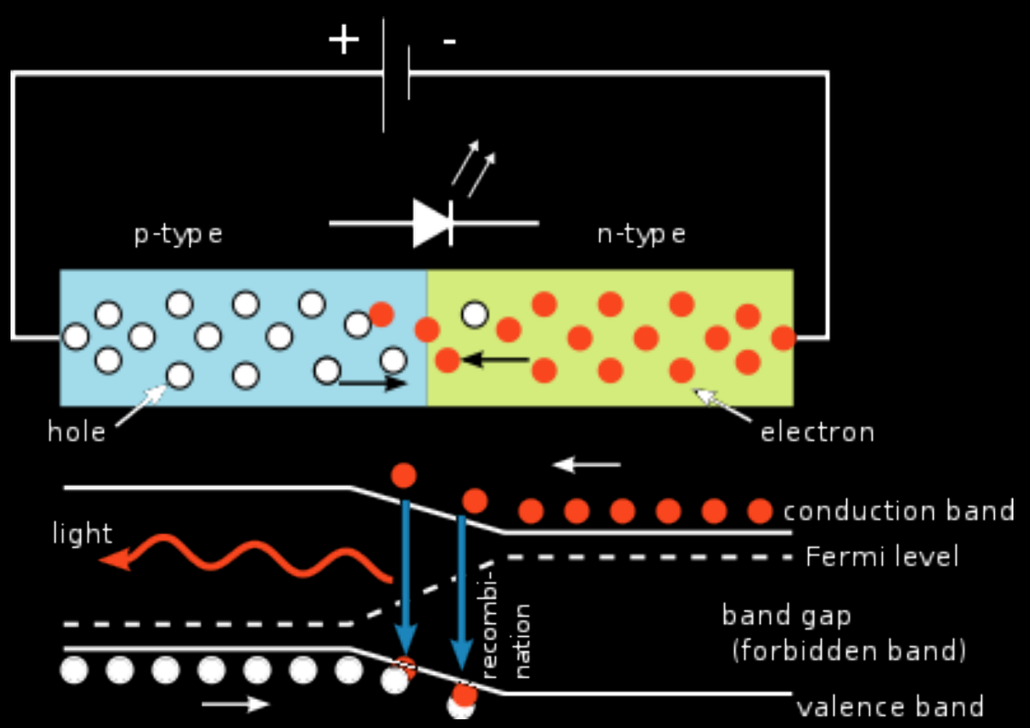
Bibliotecas para Manipular Dispositivos Básicos



LEDs



luz emitida pelo calor



luz emitida por elétrons no silício

Lâmpadas Incandescentes vs LEDs

```
>>> from gpiozero import LED
>>> led = LED(21)  # configura porta 21 da GPIO como LED
>>> led.on()          # acende o LED
>>> led.is_lit        # está aceso?
True
>>> led.off()
>>> led.is_lit
False
>>> led.toggle()    # alterna entre aceso e apagado
>>> led.is_lit
True
>>> led.toggle()
>>> led.is_lit
False
```

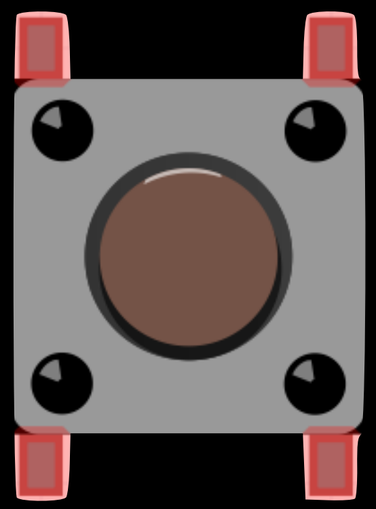
Funções para Acender e Apagar um LED

```
>>> from gpiozero import LED
>>> from time import sleep
>>> led = LED(21)
>>> for i in range(0, 4):
...     led.on()
...     sleep(1)
...     led.off()
...     sleep(1)
```

Deste jeito, o programa fica travado enquanto o LED pisca!

```
>>> from gpiozero import LED

>>> led = LED(21)

# pisque continuamente
>>> led.blink()

# LED continua piscando, mesmo rodando outros comandos
>>> x = 1000 / 57

# defina o tempo aceso e o tempo apagado
>>> led.blink(on_time=0.5, off_time=2.0)

# pisque 3 vezes
>>> led.blink(n=3)
```

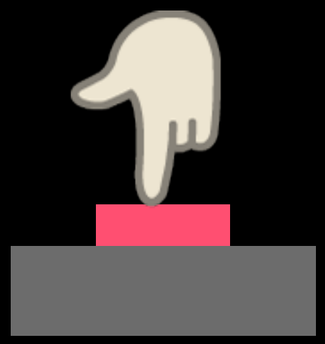
Função para Piscar o LED



Botão

```
>>> from gpiozero import Button

# configura porta 11 da GPIO como um botão
>>> botao = Button(11)

# está pressionado?
>>> botao.is_pressed
False

# pausa o programa até que o botão seja pressionado
>>> botao.wait_for_press()
```

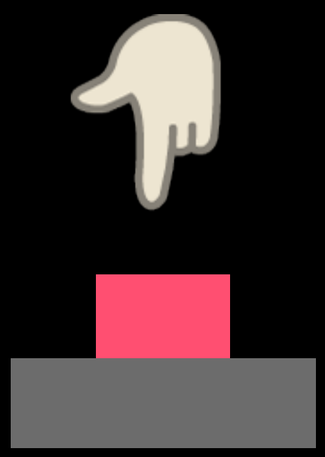
Funções para Monitorar um Botão



evento
"botão pressionado"

mantendo
pressionado

evento
"botão segurado"



evento
"botão liberado"

Eventos de um Botão

```
botao.tipo_de_evento = minha_funcao
```

chame a minha função quando o tipo de evento acontecer

```
>>> from gpiozero import Button, LED
>>> led = LED(21)
>>> def acender_led():
...     led.on()
...
>>> botao = Button(11)
>>> botao.when_pressed = acender_led
```

não coloque parênteses no final!

Evento ao Apertar um Botão

```
>>> from gpiozero import Button, LED
>>> led = LED(21)
>>> led2 = LED(22)
>>> def acender_led():
...     led.on()
...
>>> def piscar_led2():
...     led2.blink(n=2)
...
>>> def apagar_led():
...     led.off()
...
>>> botao = Button(11)
>>> botao.when_pressed = acender_led
>>> botao.when_held = piscar_led2
>>> botao.when_released = apagar_led
```

não coloque parênteses no final!

Eventos ao Apertar, Segurar e Soltar um Botão

```
>>> from gpiozero import Button, LED
>>> led = LED(21)
>>> led2 = LED(22)
>>> def acender_led():
...     led2.blink(n=2)
...
>>> botao = Button(11)
>>> botao.when_pressed = led.on
>>> botao.when_held = piscar_led2
>>> botao.when_released = led.off
```

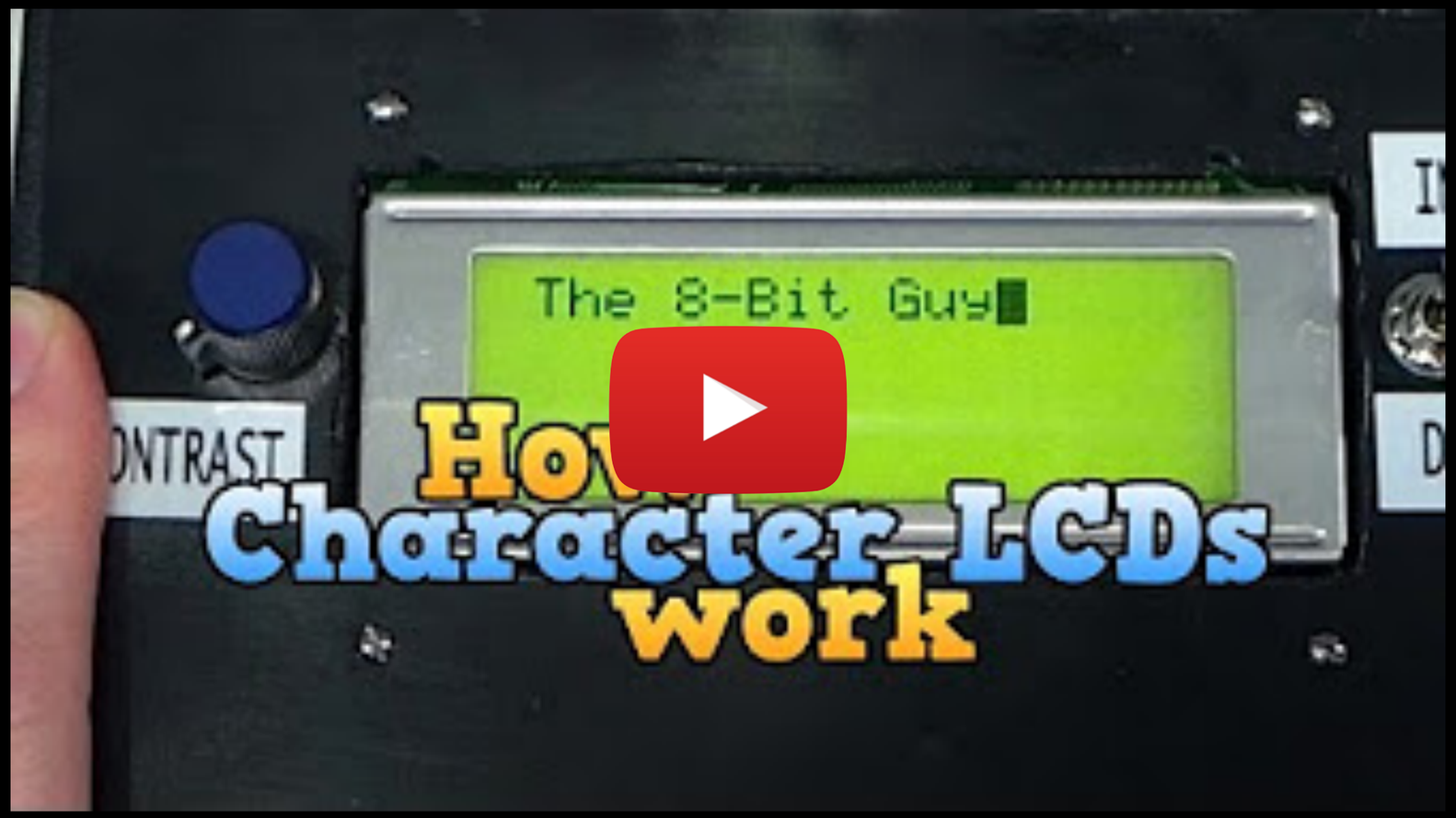
chame diretamente a função "on" do LED

Eventos ao Apertar, Segurar e Soltar um Botão

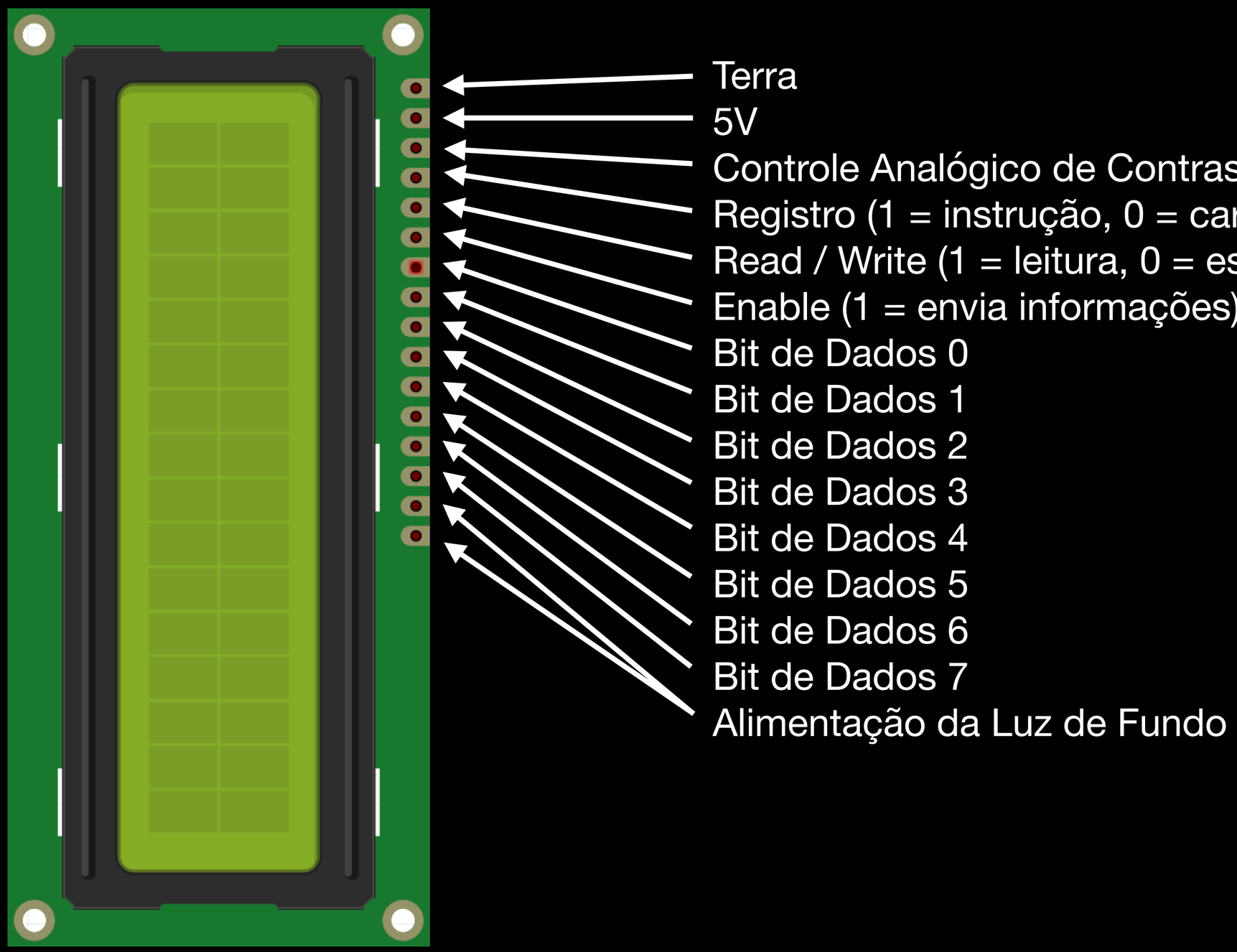
```
...
>>> botao1.when_pressed = minha_funcao1
>>> botao2.when_pressed = minha_funcao2
>>> botao3.when_pressed = minha_funcao3
...
```

Uso Mais Comum: Apenas Evento de Botão Pressionado

LCD de Caracteres



Vídeo sobre o Display LCD 16 x 2 no YouTube



Pinos do LCD de Caracteres

Instruction	Instruction code										Description	Execution time (fosc=270KHz)
	RS	R/W	DB7	DB6	DB5	DB4	DB3	DB2	DB1	DB0		
Clear Display	0	0	0	0	0	0	0	0	0	1	Clears entire display and sets DDRAM address to 00H.	1.53ms
Return Home	0	0	0	0	0	0	0	0	1	-	Sets DDRAM address to 00H in AC and returns shifted display to its original position. The contents of DDRAM remain unchanged.	1.53ms
Entry Mode Set	0	0	0	0	0	0	0	1	I/D	SH	Sets cursor move direction and enable the shift of entire display. These operations are performed during data write and read.	39 μ s
Display ON/ OFF Control	0	0	0	0	0	0	1	D	C	B	Set ON/OFF of entire display (D), cursor ON/OFF(C), and blinking of cursor position character(B).	39 μ s
Cursor or Display Shift	0	0	0	0	0	1	S/C	R/L	-	-	Moves cursor and shifts display without changing DDRAM contents.	39 μ s
Function Set	0	0	0	0	1	DL	N	F	-	-	Sets interface data length (DL: 8-bit/4-bit), numbers of display line (N: 2-line/1-line), and display font type (F: 5x11dots/5x8dots)	39 μ s
Set CGRAM Address	0	0	0	1	AC5	AC4	AC3	AC2	AC1	AC0	Set CGRAM address in address counter.	39 μ s
Set DDRAM Address	0	0	1	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Set DDRAM address in address Counter.	39 μ s
Read Busy Flag and Address	0	1	BF	AC6	AC5	AC4	AC3	AC2	AC1	AC0	Reads busy flag (BF) indicating internal operation is being performed and reads address counter contents.	0 μ s
Write data to CG or DD RAM	1	0	D7	D6	D5	D4	D3	D2	D1	D0	Write data into internal RAM (DDRAM/CGRAM).	43us
Read data from CG or DD RAM	1	1	D7	D6	D5	D4	D3	D2	D1	D0	Read data from internal RAM (DDRAM/CGRAM).	43us

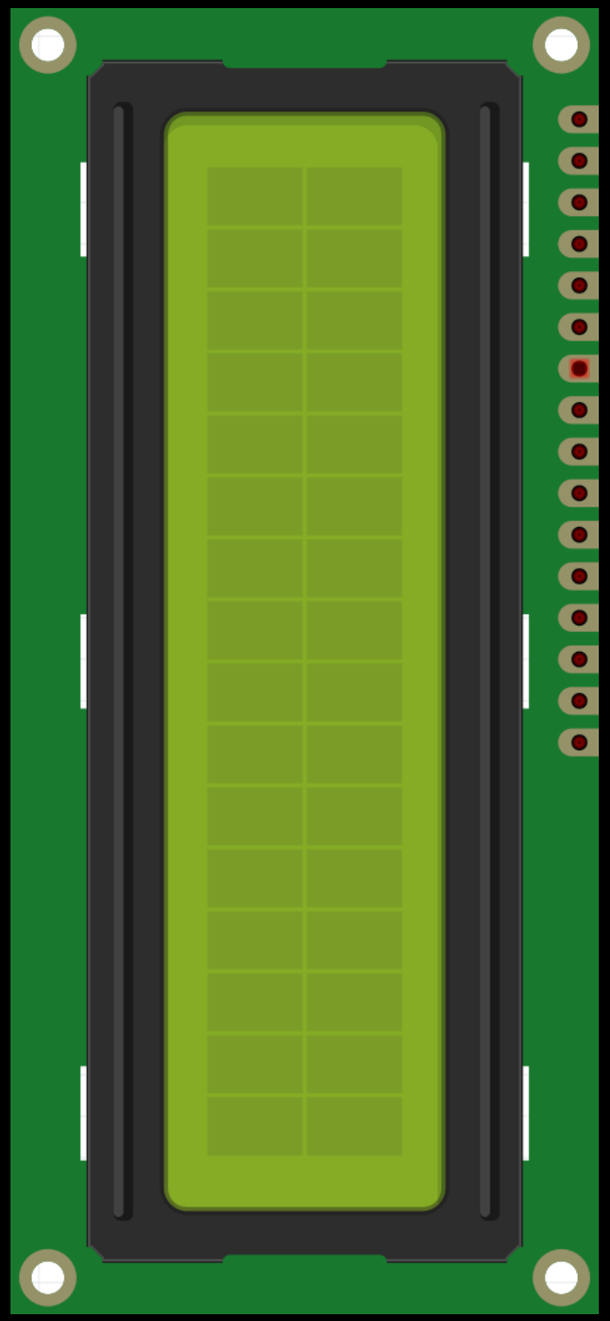
Comandos para o LCD

Precisamos implementar todos esses comandos e códigos???

Relaxa, a gente já fez tudo isso para você!

adafruit

Comandos para o LCD



Terra
5V
Controle Analógico de Contraste
Registro (1 = instrução, 0 = caracter)
Read / Write (1 = leitura, 0 = escrita)
Enable (1 = envia informações)
Bit de Dados 0
Bit de Dados 1
Bit de Dados 2
Bit de Dados 3
Bit de Dados 4
Bit de Dados 5
Bit de Dados 6
Bit de Dados 7
Alimentação da Luz de Fundo

pinos conectados na GPIO

dimensões do LCD

```
Adafruit_CharLCD(RS, EN, D4, D5, D6, D7, colunas, linhas)
```

Pinos Usados pela Biblioteca da Adafruit

```
Adafruit_CharLCD(RS, EN, D4, D5, D6, D7, colunas, linhas)
```

no nosso caso...

```
Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)
```

Pinos Usados pela Biblioteca da Adafruit

```
>>> from Adafruit_CharLCD import Adafruit_CharLCD
>>> lcd = Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)
>>> lcd.message("Teste")
>>> lcd.message("Teste 2")
>>> lcd.clear()
>>> lcd.message("Teste 3")
```

Exibição de Mensagens no LCD de Caracteres

```
>>> from Adafruit_CharLCD import Adafruit_CharLCD
>>> lcd = Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)
>>> lcd.message("Teste com texto longo")
>>> lcd.clear()
>>> lcd.message("Teste com\n texto longo")
```

Mensagens Longas no LCD de Caracteres

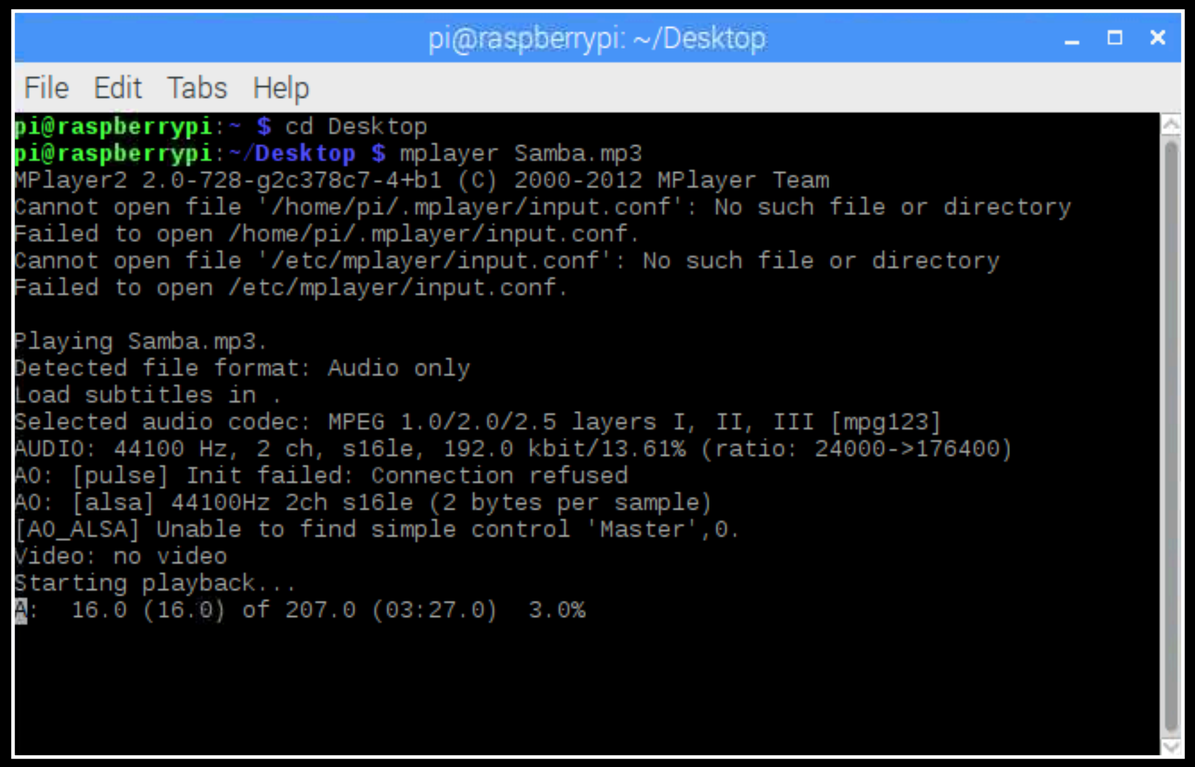
```python
# bibliotecas
from gpiozero import LED, Button
from Adafruit_CharLCD import Adafruit_CharLCD
from time import sleep

# funções
def minha_funcao1():
    ...

def minha_funcao2():
    ...

# componentes
led1 = LED(21)
led2 = LED(22)
lcd = Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)
botao = Button(11)
botao.when_pressed = minha_funcao1 ← fora do while!
...

# loop infinito (se necessário)
while True:
    ...
    sleep(0.5)
```

Estrutura Geral do Código

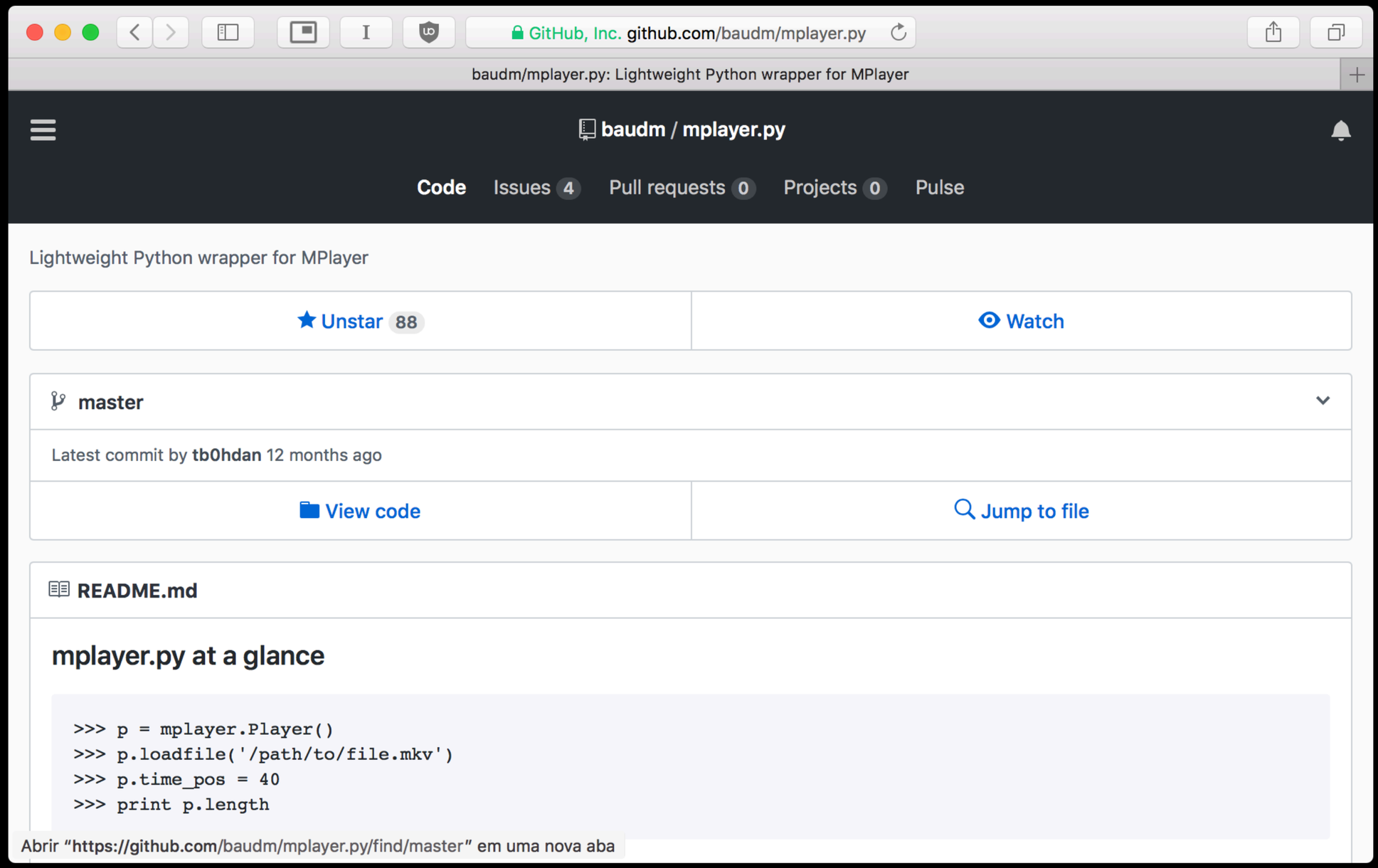
Software

Aplicativo MPlayer

Interface Gráfica do MPlayer



Uso do MPlayer via Terminal



Integração do MPlayer com Python

Foco da Aula: Reprodução de Áudio

```
>>> from mplayer import Player
>>> player = Player() # inicializa o player
>>> player.loadfile("Musica.mp3")
>>> player.pause()
>>> player.paused
True
>>> player.pause()    # alterna entre pausa e despausa
>>> player.paused
False
>>> player.stop()
>>> player.quit()
```

```
>>> from mplayer import Player
>>> player = Player()
>>> player.loadfile("Musica.mp3")
# duração total do áudio em segundos
>>> player.length
490.985944
# instante de tempo atual da reprodução de áudio
>>> player.time_pos
145.358277
# vá para uma posição de tempo específica
>>> player.time_pos = 60
# pule 5 segundos
>>> player.time_pos += 5
```

Controle do Tempo

```
>>> from mplayer import Player
>>> player = Player()
>>> player.loadfile("Musica.mp3")
>>> player.metadata
{'Title': 'Ixtapa', 'Artist': 'Rodrigo y Gabriela',
'Album': 'Area 52'}
>>> player.metadata["Artist"]
'Rodrigo y Gabriela'
>>> player.metadata["Genre"]
Traceback (most recent call last):
  File "<pyshell>", line 1, in <module>
KeyError: 'Genre'
>>> "Genre" in player.metadata
False
```

chave pode não existir nos metadados, resultando em erro

Metadados do Áudio

```
>>> from mplayer import Player
>>> player = Player()
>>> player.loadlist("lista.txt")
>>> player.pt_step(1)    # avance para a próxima faixa
>>> player.pt_step(2)    # avance 2 faixas
>>> player.pt_step(-1)   # vá para a faixa anterior
```

lista.txt

```
caminho/para/musica01.mp3
caminho/para/musica02.mp3
caminho/para/musica03.mp3
caminho/para/musica04.mp3
caminho/para/musica05.mp3
caminho/para/musica06.mp3
caminho/para/musica07.mp3
caminho/para/musica08.mp3
caminho/para/musica09.mp3
caminho/para/musica10.mp3
caminho/para/musica11.mp3
caminho/para/musica12.mp3
caminho/para/musica13.mp3
caminho/para/musica14.mp3
```

Reprodução de uma Lista

```
>>> player.volume          # valor entre 0 e 100
100

>>> player.volume = 70
>>> player.speed = 2      # toque 2 vezes mais rápido
>>> player.speed = -2     # toque 2 vezes mais lento
```

Outras Funções do MPlayer

Resumo da Opera

Funcionalidade	Comandos

LED
acessar documentação
	`from gpiozero import LED • led = LED(21)`
	`led.on() • led.off() • led.toggle() • led.is_lit`
	`led.blink() • led.blink(n=4, on_time=0.5, off_time=2)`

Botão
acessar documentação
	`from gpiozero import Button`
	`botao = Button(11)`
	`botao.is_pressed`
	`botao.when_pressed = funcao`
	`botao.when_released = funcao`
	`botao.when_held = funcao`

LCD
acessar documentação
	`from Adafruit_CharLCD import Adafruit_CharLCD`
	`lcd = Adafruit_CharLCD(2, 3, 4, 5, 6, 7, 16, 2)`
	`lcd.clear()`
	`lcd.message("Texto 1\nTexto 2")`

MPlayer	`from mplayer import Player • player = Player()`
	`player.loadfile("Musica.mp3") • player.loadlist("lista.txt")`
	`player.pause() • player.paused • player.quit()`
	`player.time_pos = 2 • player.length • player.pt_step(-1)`
	`player.metadata["Title"] • player.metadata["Artist"]`
	`player.volume = 70 • player.speed = 2`

Resumo para o Projeto 01

Funcionalidade	Comandos
Funções	`x = input("Digite um número: ")` • `print("Resultado: ", x)` `from time import sleep` • `sleep(0.5)`

Listas
acessar documentação
	`lista = [1, 2, 3]` • `lista2 = ["texto", [0, 0], 5]`
	`lista[0]` • `lista[1:3]` • `total_de_elementos = len(lista)`
	`lista.append(novo_elemento)` • `del lista[indice]`

Dicionários acessar documentação	`dicionario = {"chave 1": 42, "chave 2": [1, 2, 3]}` `dicionario["chave 1"]` • `dicionario["chave 3"] = "Olá!"`

Textos (Strings)
acessar documentação
	`texto = "olá!"` • `texto[0]` • `texto[1:4]` • `len(texto)`
	`texto + str(numero)` • `"x = %.2f, y = %d" % (num1, num2)`
	`2 + int("11")` • `4 / float("23.5")`

Condicionais	`if x == 0 and z < 4:`
	`    y = 4`
	
	`if x not in [1, 2]:`
	`    y = 4`
	`else:`
	`    y = 0`
	
	`if x != 0:`
	`    y = 4`
	`elif x >= 0:`
	`    y = 3`
	`else:`
	`    y = 0`

Repetições	`for elem in lista:`
	`    ...`
	
	`for i in range(1, 4):`
	`    ...`
	
	`while x > 1:`
	`    ...`

Criação de Funções e Variáveis Globais	`def funcao1(x):`
	`    return x + 2`
	
	`def funcao2(x, y, z):`
	`    ...`
	
	`global x`
	`x = 2`