

# Sistemas Operacionais Embarcados

I2C

# Comunicação serial

Possíveis cenários:

- Aparelhos ligados à internet (IoT)
- Envio de dados de um microcontrolador para um computador pessoal
- Troca de dados entre microcontroladores
- Leitura de sensores (GPS, acelerômetro etc.)
- Leitura e escrita em memória externa

→ USB

→ WiFi

→ Ethernet

→ Bluetooth

→ HDMI

→ VGA

→ UART

→ SPI

→ I2C

→ I2S

→ CAN

→ Etc.

# Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ USB

→ WiFi

→ Ethernet

→ Bluetooth

→ HDMI

→ UART

→ SPI

→ I2C

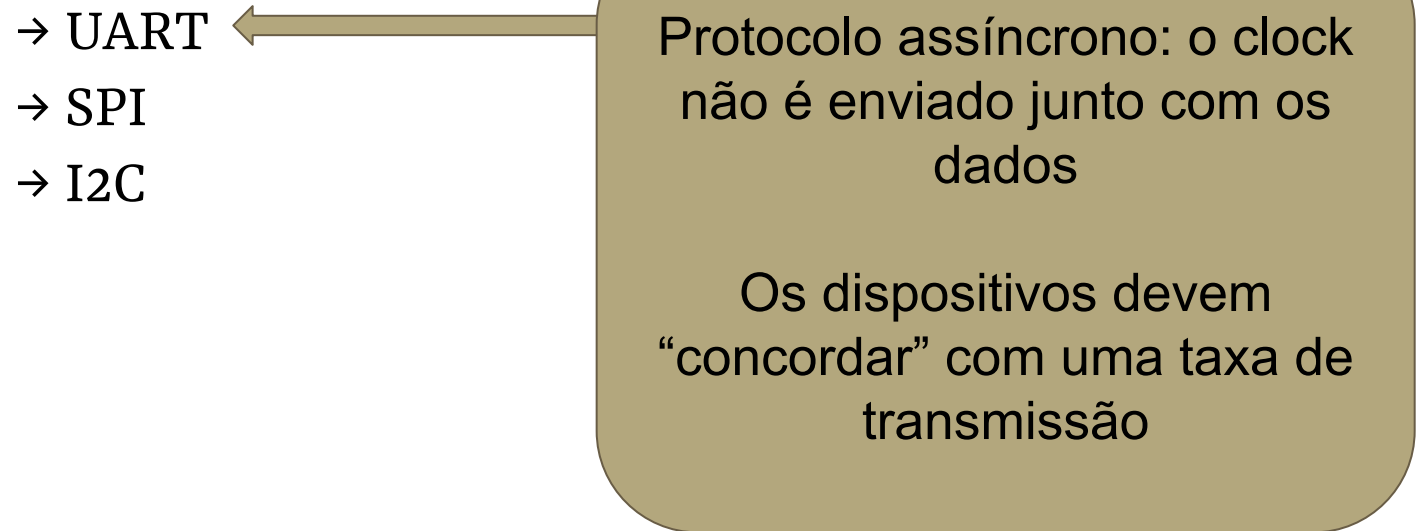
# Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C



Protocolo assíncrono: o clock  
não é enviado junto com os  
dados

The diagram consists of a light brown rounded rectangle on the right containing text. A horizontal arrow points from the left side of this rectangle to the text '→ UART' on the left. Below '→ UART' are the texts '→ SPI' and '→ I2C'.

Os dispositivos devem  
“concordar” com uma taxa de  
transmissão


# Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C



Protocolos síncronos: o clock é enviado junto com os dados

O dispositivo que gera o clock é denominado “mestre”, e os demais dispositivos são denominados “escravos”

# Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C

Necessita de 2 fios:  
transmissão e recepção

Permite comunicação  
*full-duplex*

O fio de transmissão pode  
ser usado para indicar o  
endereço do dispositivo  
(quando há mais de um  
deles)

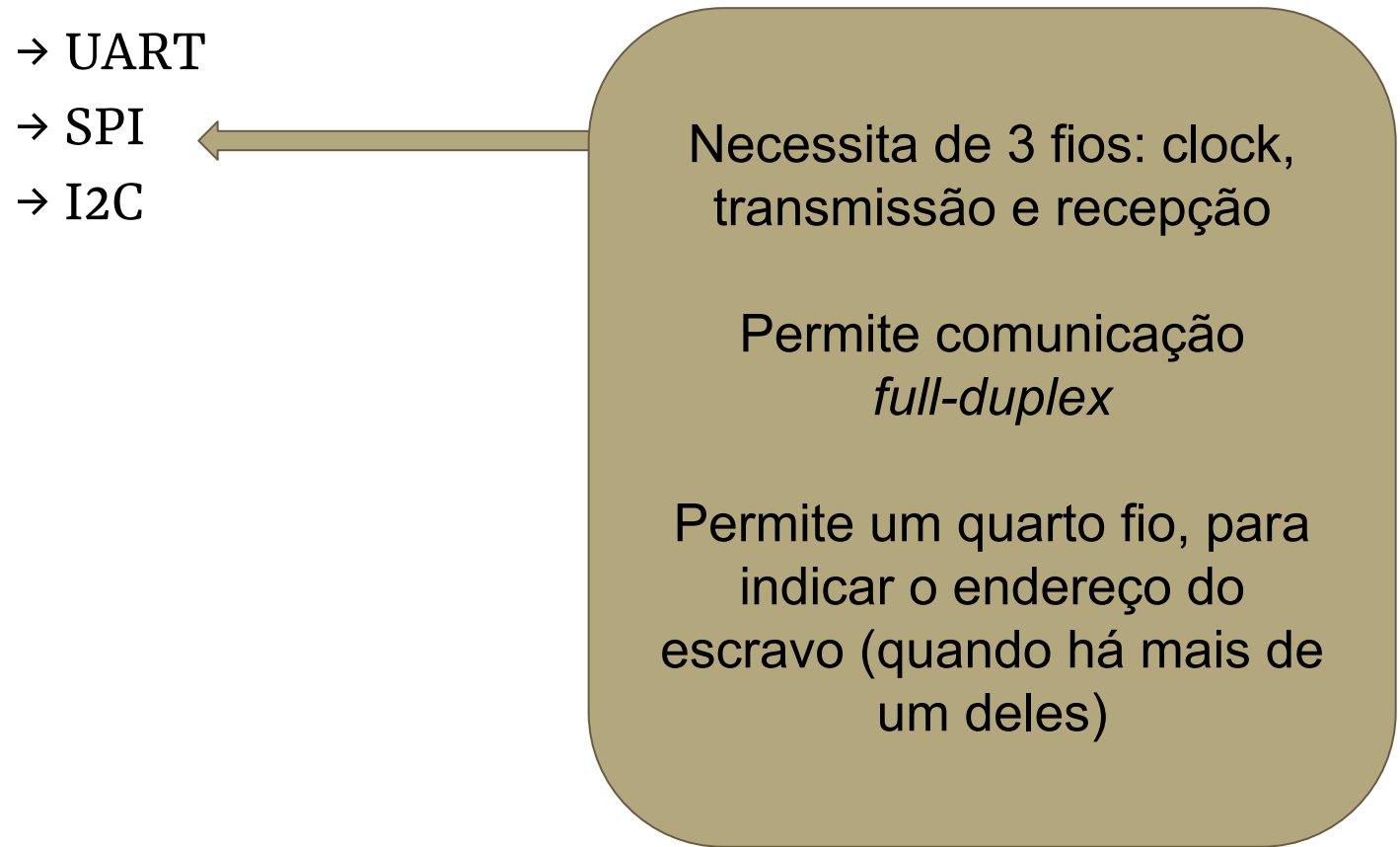
# Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C



Necessita de 3 fios: clock, transmissão e recepção

Permite comunicação  
*full-duplex*

Permite um quarto fio, para  
indicar o endereço do  
escravo (quando há mais de  
um deles)

# Comunicação serial

Protocolos disponíveis no Raspberry Pi (dependente do modelo):

→ UART

→ SPI

→ I2C

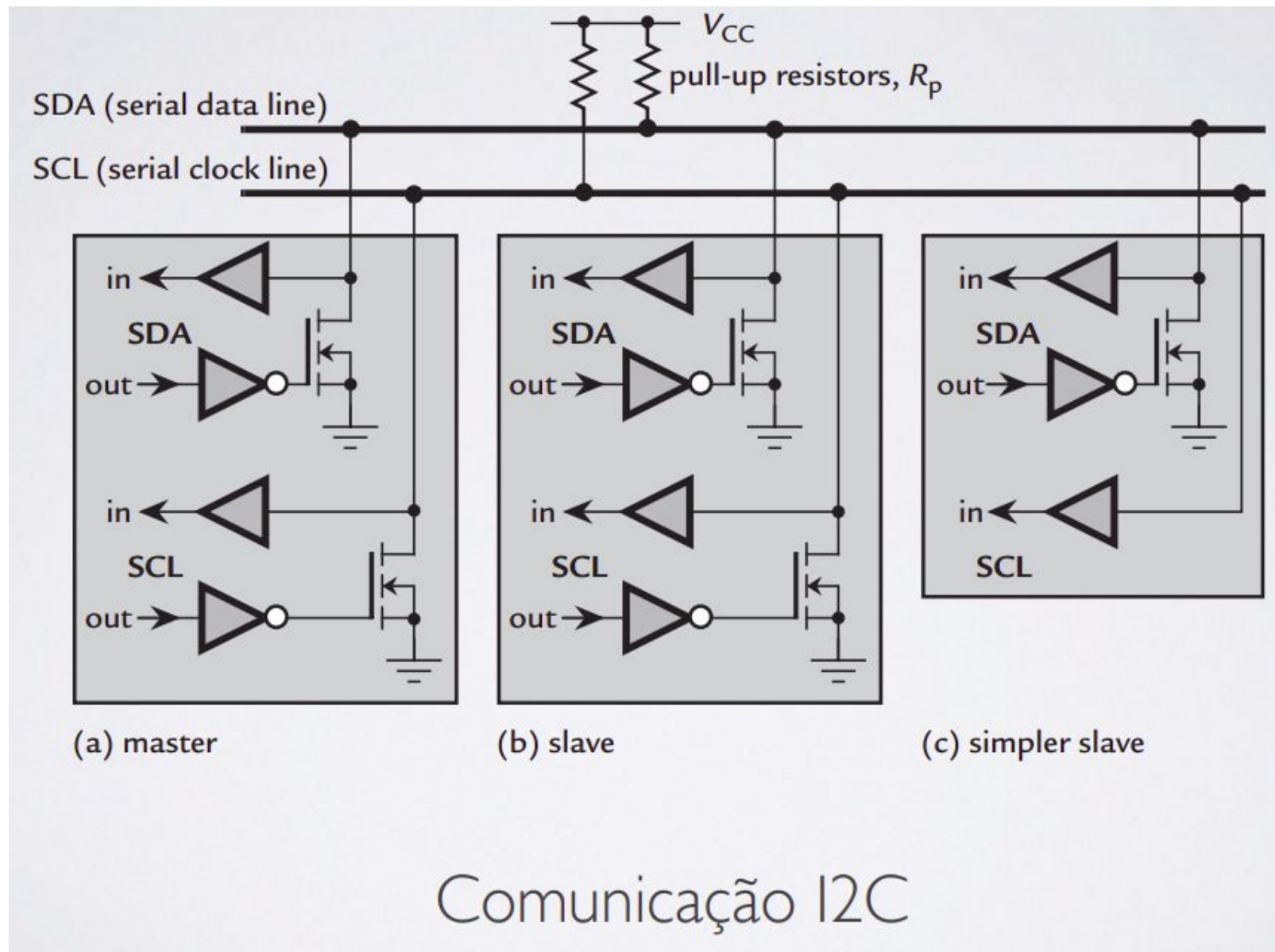
Possui dois fios: clock e dados

Permite comunicação  
*half-duplex*

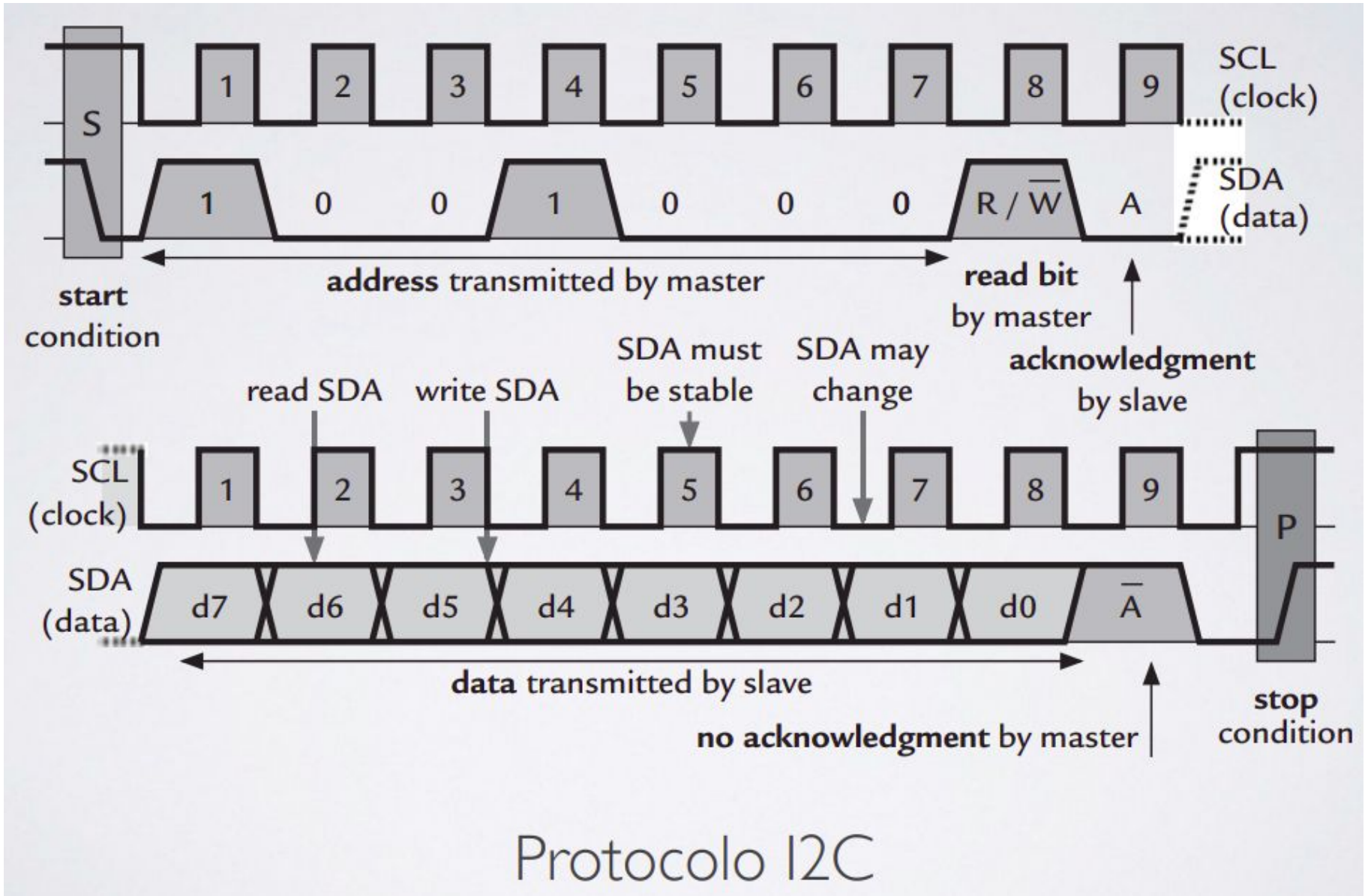
O fio de dados pode ser usado para indicar o endereço do escravo (quando há mais de um deles)



# I2C



# I2C



# I2C no RPi

3v3 Power	1			2	5v Power
GPIO 2 (Data)	3			4	5v Power
GPIO 3 (Clock)	5			6	Ground
GPIO 4 (GPCLK0)	7			8	GPIO 14 (UART TX)
Ground	9			10	GPIO 15 (UART RX)
GPIO 17	11			12	GPIO 18 (PCM CLK)
GPIO 27	13			14	Ground
GPIO 22	15			16	GPIO 23
3v3 Power	17			18	GPIO 24
GPIO 10 (SPI0 MOSI)	19			20	Ground
GPIO 9 (SPI0 MISO)	21			22	GPIO 25
GPIO 11 (SPI0 SCLK)	23			24	GPIO 8 (SPI0 CE0)
Ground	25			26	GPIO 7 (SPI0 CE1)
GPIO 0 (EEPROM Data)	27			28	GPIO 1 (EEPROM Clock)
GPIO 5	29			30	Ground
GPIO 6	31			32	GPIO 12 (PWM0)
GPIO 13 (PWM1)	33			34	Ground
GPIO 19 (PCM FS)	35			36	GPIO 16
GPIO 26	37			38	GPIO 20 (PCM DIN)
Ground	39			40	GPIO 21 (PCM DOUT)

# I2C no RPi

O Raspbian não tem a comunicação I2C habilitada inicialmente. Para utiliza-la, você deve habilita-la da seguinte maneira:

1 - Execute

```
$ sudo raspi-config
```

vá em Advanced options -> I2C, e habilite o acesso:

Would you like the ARM I2C interface to be enabled? Yes.

2 - Reinicie (reboot) o sistema para tornar esta mudança efetiva.

# I2C no RPi

3 - Execute

```
$ cat /boot/config.txt | grep i2c
```

Deverá aparecer o texto

```
dtoverlay=i2c-arms
```

indicando que a I2C está habilitada.

# I2C no RPi

## 4 - Execute

```
sudo apt-get install -y i2c-tools
```

para obter ferramentas de teste e configuração da comunicação I2C pelo terminal. Depois de instaladas, execute

```
i2cdetect -y 1
```

para enxergar o endereço de todos os dispositivos I2C conectados. (O código nesta pasta para o MSP430 o configura com endereço 0xF.)

# I2C no RPi

5 - Para mudar a taxa de transmissão I2C para 50000 Hz, por exemplo, deve-se acrescentar a seguinte linha ao arquivo `/boot/config.txt`:

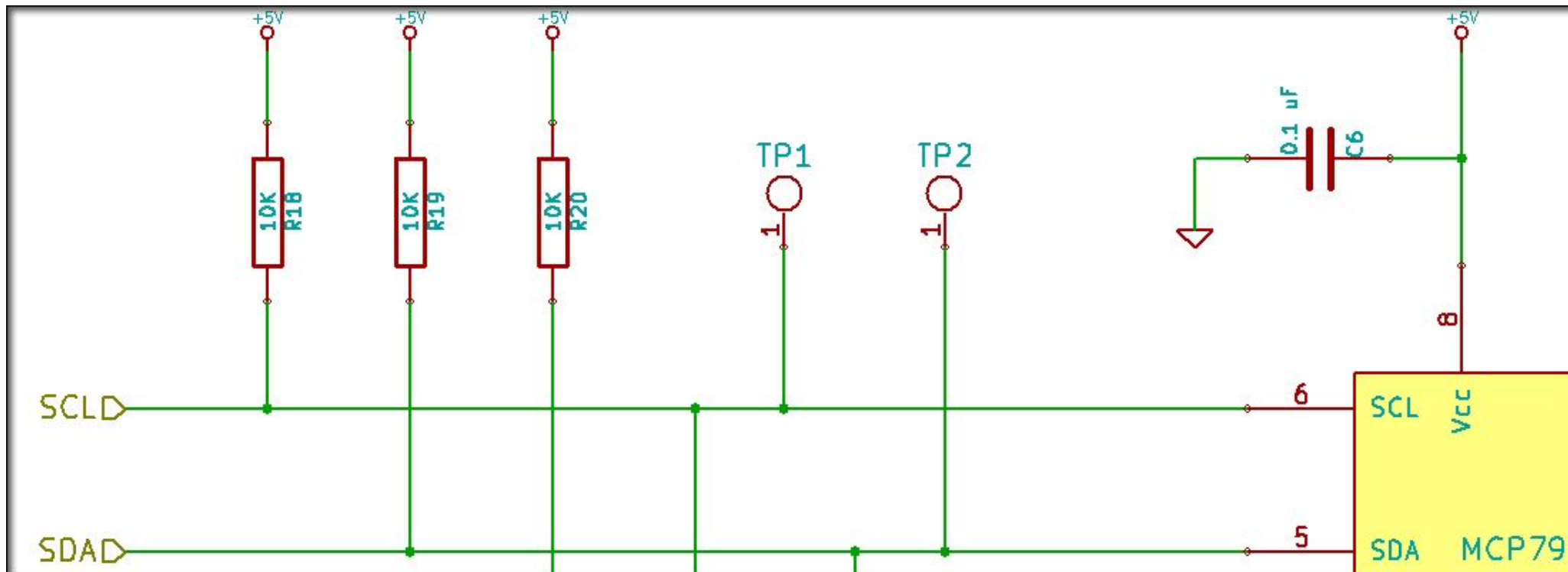
```
dtoverlay=i2c_baudrate=50000
```

A mudança só será efetivada depois do reboot.

# I2C no RPi

6 - É necessário conectar resistores de pullup nas duas linhas, SCL (clock) e SDA (dados). A comunicação I2C requer estes resistores porque as linhas são conectadas na configuração coletor aberto

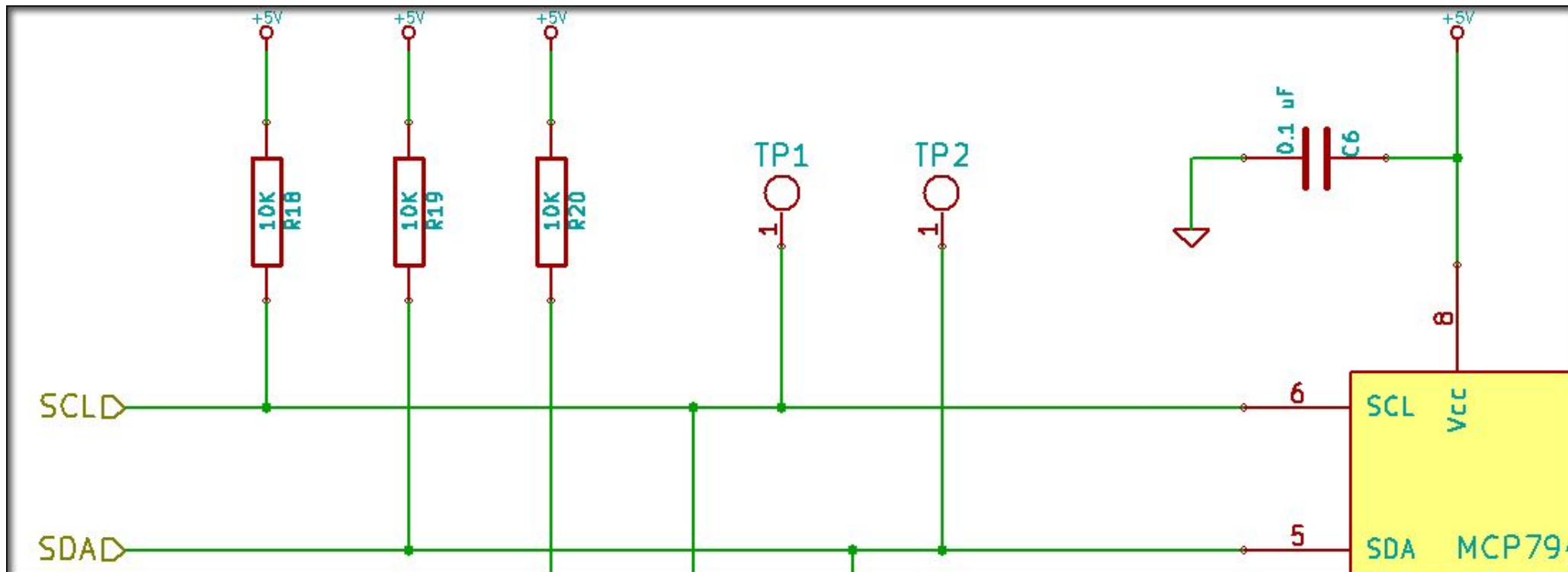
(<https://learn.sparkfun.com/tutorials/i2c/i2c-at-the-hardware-level>):





# I2C no RPi

Nesta configuração, qualquer dispositivo, mestre ou escravo, consegue levar a linha para 0 (GND) a qualquer momento, e qualquer dispositivo só consegue levar a linha para 1 (Vcc) se nenhum outro dispositivo "segurar" a linha em 0.



# I2C no RPi

7 - As linhas SCL e SDA possuem uma capacitância intrínseca, o que afeta a escolha dos resistores de pull-up. Quanto maior o resistor de pull-up, maior o tempo para levar as linhas de 0 para 1, afetando a performance do sistema. Quanto mais comprida a conexão física da linha, maior a capacitância. Geralmente, escolhem-se resistores menores do que 10000 ohms.

Se os valores escolhidos para os resistores não estiverem adequados, utilize as seguintes referências para calculá-los:

- <http://www.ti.com/lit/an/slva689/slva689.pdf>
- <http://dsscircuits.com/articles/effects-of-varying-i2c-pull-up-resistors>

# Hardware para exemplos I2C

