embarca tech | Residência Tecnológica em Sistemas Embarcados

# Aplicações de SBCs

## Unidade 5 | Capítulo 2 – Caninos Loucos Labrador e interfaces seriais

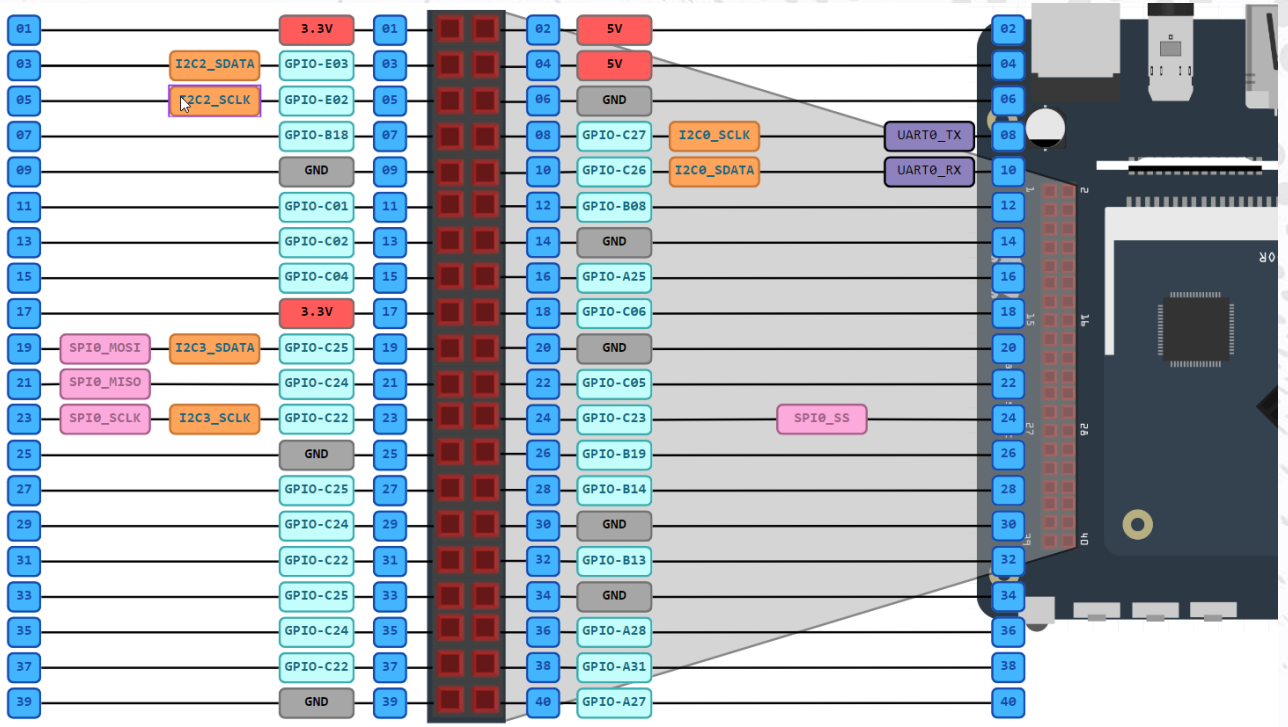
Executores:
INSTITUTO FEDERAL Piauí
INSTITUTO FEDERAL Rio Grande do Norte
INSTITUTO FEDERAL Maranhão
INSTITUTO FEDERAL Ceará
cepedi
INSTITUTO HARDWARE BR

Coordenação: Softex

Iniciativa: MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E INOVAÇÃO
GOVERNO FEDERAL BRASIL UNIÃO E RECONSTRUÇÃO

# Introdução

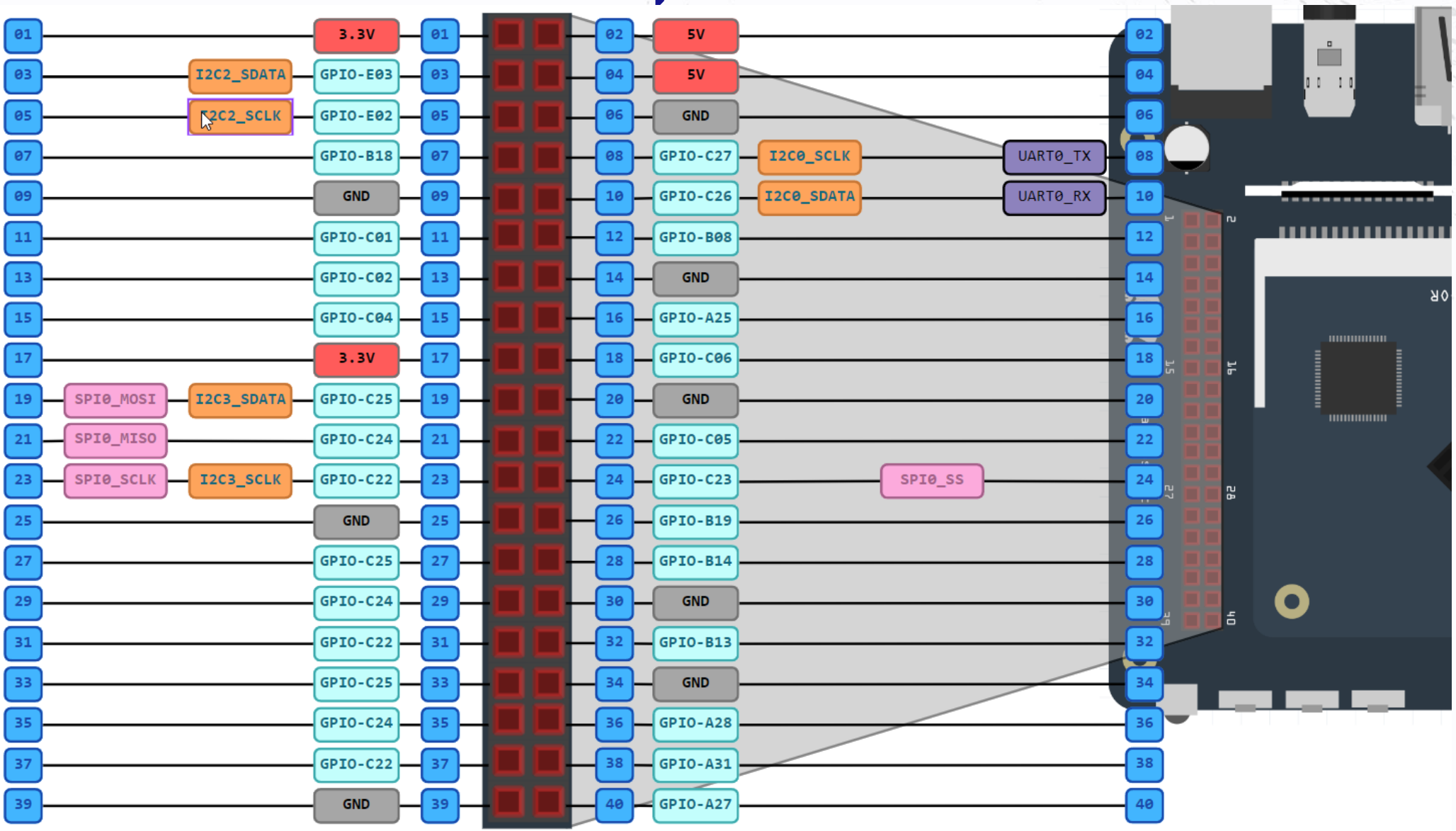
- **Interfaces seriais da Labrador:**
  - **UART**: Pinos de comunicação serial assíncrona (Tx/RX);
  - **I2C**: Pinos de barramento de comunicação síncrona a dois fios;
  - **SPI**: Pinos de barramento de comunicação de alta velocidade a três fios;




- **Biblioteca python-periphery:**
  - A biblioteca python-periphery oferece acesso direto e eficiente a periféricos de hardware em sistemas Linux, como GPIO, I2C, SPI, UART, PWM e MMIO, sendo ideal para projetos IoT e automação. Simples e de alto desempenho, facilita o controle de sensores e atuadores em dispositivos embarcados.

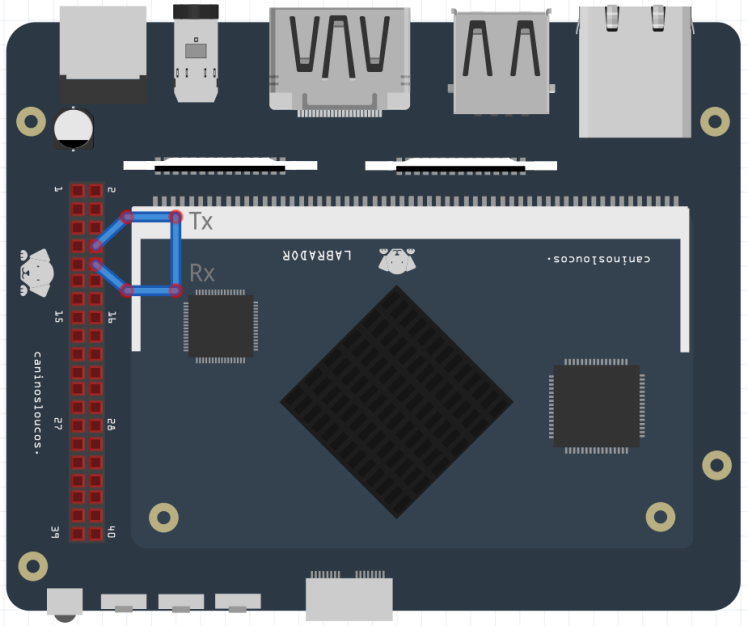
https://www.canva.com/design/DAGcMFy6Sr0/rm6CzbnHj7wLVhNnz0sguQ/edit?utm_content=DAGcMFy6Sr0&utm_campaign=designshare&utm_medium=link2&utm_source=sharebutton

# Labrador 32 – UART, I2C e SPI



# Labrador 32 – UART: 03-uart

- UART em Loopback (Tx conectado no Rx)



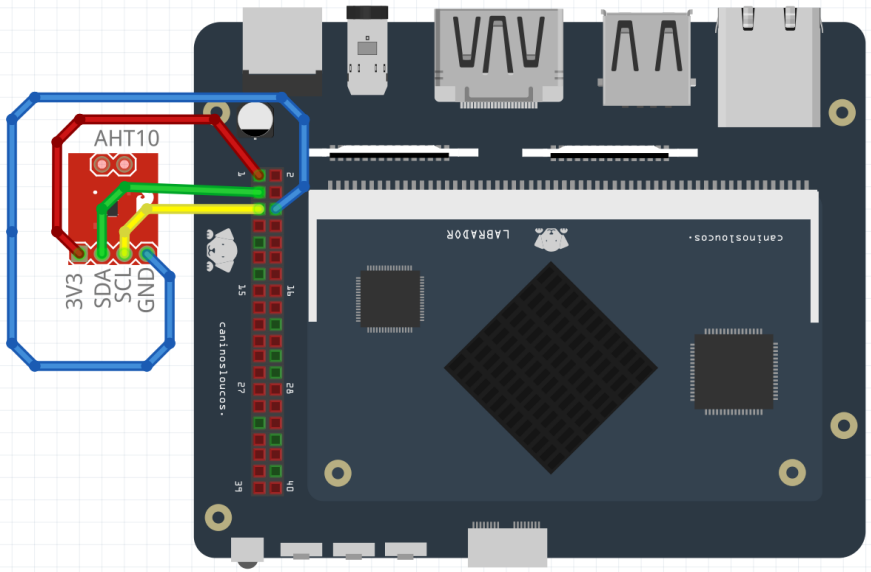

```python
from periphery import Serial

# Configura a serial /dev/ttyUSB0 com baudrate 9600
serial = Serial("/dev/ttyS0", 9600)

while(1):
    data_to_send = input("Dado a enviar: ").encode('utf-8')
    serial.write(data_to_send)
    data_readed = serial.read(len(data_to_send)).decode('utf-8')
    print(f'Dado recebido:{data_readed}’)
```

```
/dev/ttyS0
#Pino 08: UART0_Tx
#Pino 10: UART0_Rx
```

# Labrador 32 – I2C: - 04-i2c

- I2C é um protocolo de comunicação serial que permite a conexão de múltiplos dispositivos em um barramento compartilhado.

```
#Default I2C bus.
#- Pin 3: SDA (data)
#- Pin 5: SCK (clock)
```



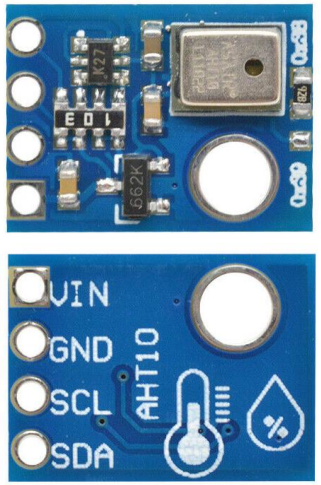

Inicialização do I2C

```
from periphery import I2C
import time

# Configurações
I2C_BUS = "/dev/i2c-2"  # Substitua pelo seu barramento I²C
I2C_ADDRESS = 0x38      # Endereço do AHT10

# Inicialização do barramento I2C
i2c = I2C(I2C_BUS)
```

# Labrador 32 – I2C: - 04-i2c

& função de leitura

- O sensor AHT10 é um sensor de temperatura e humidade que utiliza protocolo I2C para transferência dos dados de medição.




https://www.ebay.com/itm/295015225651

Inicialização do AHT10

```
def aht10_init():
    """Inicializa o sensor AHT10."""
    # Comando para inicializar o AHT10
    init_command = [0xE1, 0x08, 0x00]
    # Envia comando de inicialização
    i2c.transfer(I2C_ADDRESS, [I2C.Message(init_command)])
    # Aguarde 20ms para estabilização
    time.sleep(0.02)
```

| command | Interpretation | Code | |
|---|---|---|---|
| Initialization command | Keep the host | 1110’0001 | 0xE1 |
| Trigger measurement | Keep the host | 1010’1100 | 0xAC |
| Soft reset | | 1011’1010 | |

https://components101.com/sites/default/files/component_datasheet/AHT10.pdf

# Labrador 32 – I2C: - 04-i2c

& função de leitura

- O sensor AHT10 é um sensor de temperatura e humidade que utiliza protocolo I2C para transferência dos dados de medição.

$$RH[\%] = \left(\frac{S_{RH}}{2^{20}}\right) * 100\%$$

$$T(°C) = \left(\frac{S_T}{2^{20}}\right) * 200 - 50$$

https://components101.com/sites/default/files/component_datasheet/AHT10.pdf

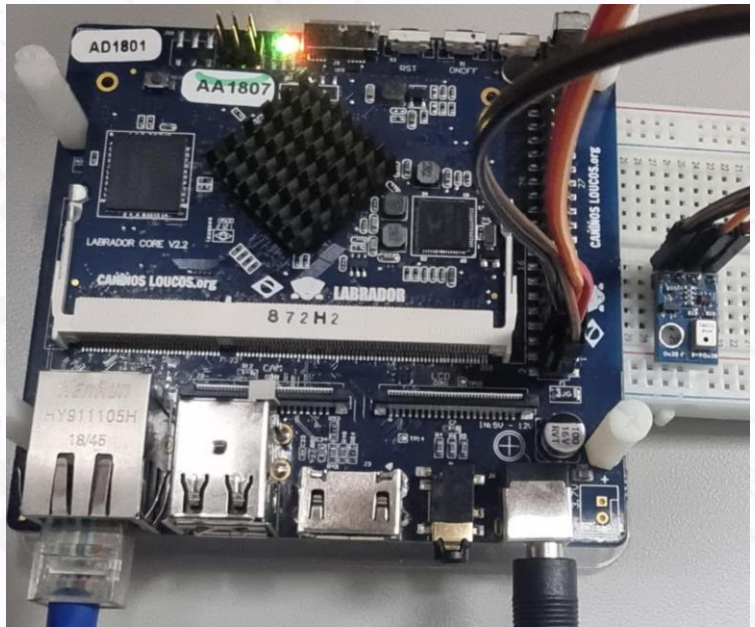
Leitura de medições do AHT10

```python
def aht10_read():
    """Lê temperatura e umidade do sensor AHT10."""
    # Envia comando para iniciar medição
    measure_command = [0xAC, 0x33, 0x00]
    i2c.transfer(I2C_ADDRESS, [I2C.Message(measure_command)])
    time.sleep(0.08)  # Aguarde 80ms para conversão

    # Lê os 6 bytes de dados do sensor
    read_message = I2C.Message([0x00] * 6, read=True)
    i2c.transfer(I2C_ADDRESS, [read_message])
    data = read_message.data

    """ Data registra os 5 bytes
    Bytes 1, 2 e a primeira metade do 3 – humidade
    Segunda metade do 3, 4 e 5 - temperatura ;"""
    humidity_raw = ((data[1] << 16) | (data[2] << 8) | data[3]) >> 4
    temperature_raw = ((data[3] & 0x0F) << 16) | (data[4] << 8) | data[5]

    # Calcula umidade e temperatura reais
    humidity = (humidity_raw / 1048576) * 100  # Em percentual
    temperature = (temperature_raw / 1048576) * 200 - 50  # Em °C

    return temperature, humidity
```

# Labrador 32 – I2C: - 04-i2c

- Operação I2C

```
#Default I2C bus.
#- Pin 3: SDA (data)
#- Pin 5: SCK (clock)
```



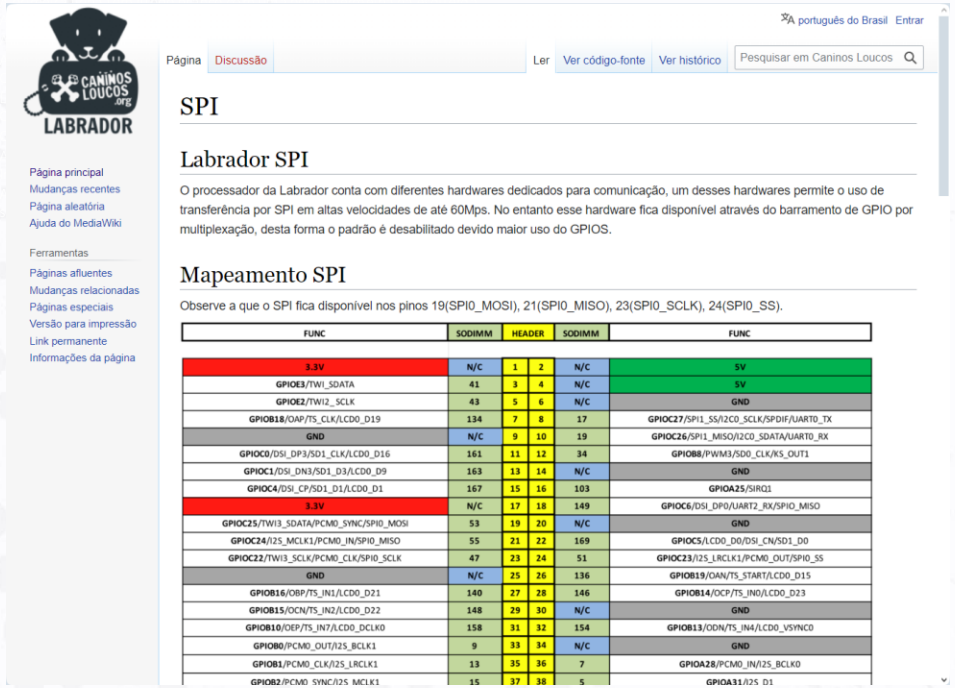

Execução principal

```
while True:
    aht10_init()
    # Inicializa o sensor
    temp, hum = aht10_read()
    # Lê os valores de temperatura e umidade
    print(f"Temperatura: {temp:.2f} °C, Umidade: {hum:.2f} %")
```

```
caninos@labrador: ~/labrador-examples/periphery
caninos@labrador:~/labrador-examples/periphery$ python3 04-i2c.py
Temperatura: 25.66 °C, Umidade: 49.48 %
caninos@labrador:~/labrador-examples/periphery$
```

# Labrador 32 – SPI: https://wiki.caninosloucos.org/index.php/SPI

```
"/dev/spidev0.0"
```



português do Brasil Entrar

Página Discussão Ler Ver código-fonte Ver histórico Pesquisar em Caninos Loucos

LABRADOR

Página principal
Mudanças recentes
Página aleatória
Ajuda do MediaWiki

Ferramentas
Páginas afluentes
Mudanças relacionadas
Páginas especiais
Versão para impressão
Link permanente
Informações da página

## SPI

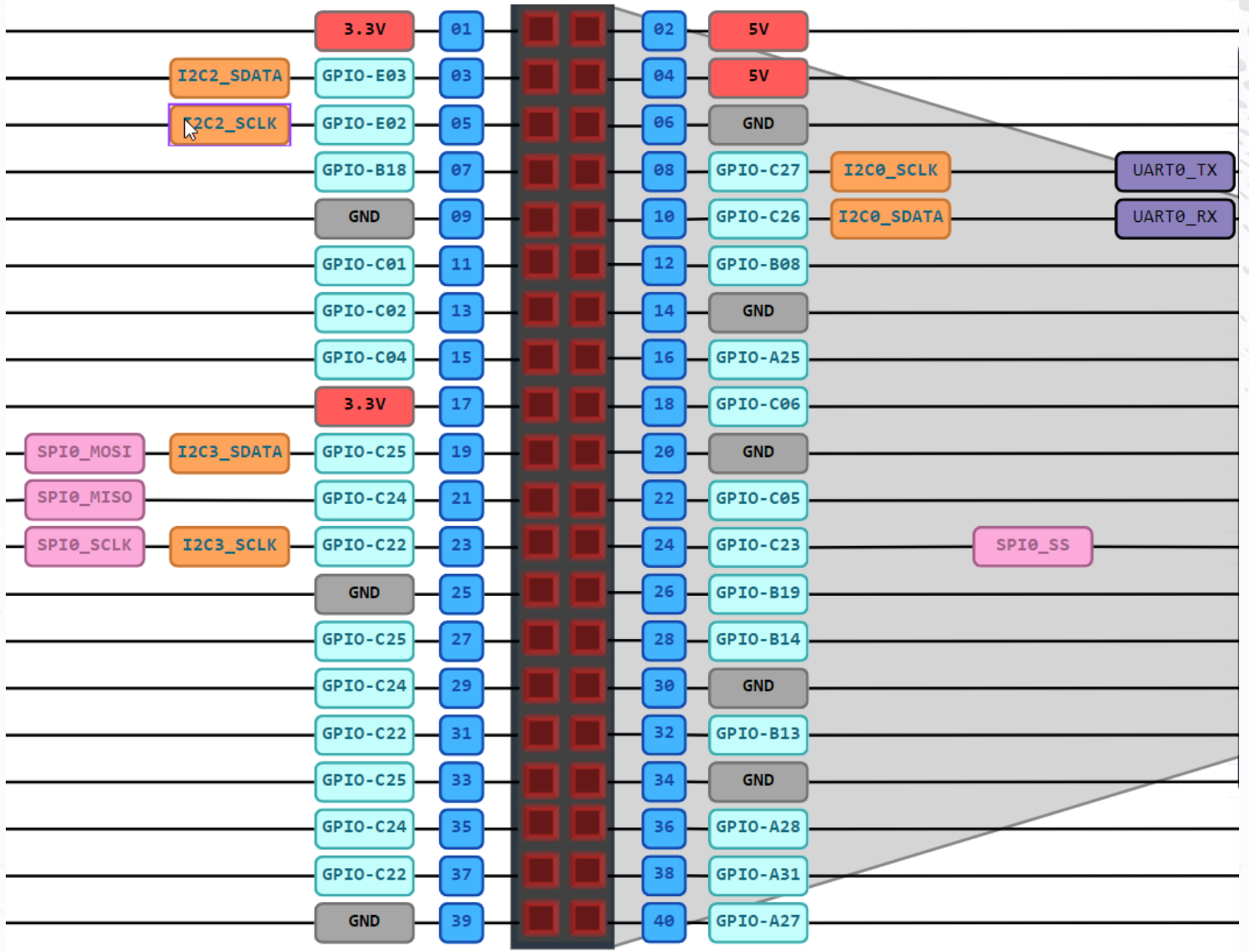
### Labrador SPI

O processador da Labrador conta com diferentes hardwares dedicados para comunicação, um desses hardwares permite o uso de transferência por SPI em altas velocidades de até 60Mps. No entanto esse hardware fica disponível através do barramento de GPIO por multiplexação, desta forma o padrão é desabilitado devido maior uso do GPIOS.

### Mapeamento SPI

Observe a que o SPI fica disponível nos pinos 19(SPI0_MOSI), 21(SPI0_MISO), 23(SPI0_SCLK), 24(SPI0_SS).

| FUNC | SODIMM | HEADER | | SODIMM | FUNC |
|---|---|---|---|---|---|
| 3.3V | N/C | 1 | 2 | N/C | 5V |
| GPIOE3/TWI_SDATA | 41 | 3 | 4 | N/C | 5V |
| GPIOE2/TWI2_SCLK | 43 | 5 | 6 | N/C | GND |
| GPIOB18/OAP/TS_CLK/LCD0_D19 | 134 | 7 | 8 | 17 | GPIOC27/SPI1_SS/I2C0_SCLK/SPDIF/UART0_TX |
| GND | N/C | 9 | 10 | 19 | GPIOC26/SPI1_MISO/I2C0_SDATA/UART0_RX |
| GPIOC0/DSI_DP3/SD1_CLK/LCD0_D16 | 161 | 11 | 12 | 34 | GPIOB8/PWM3/SD0_CLK/KS_OUT1 |
| GPIOC1/DSI_DN3/SD1_D3/LCD0_D9 | 163 | 13 | 14 | N/C | GND |
| GPIOC4/DSI_CP/SD1_D1/LCD0_D1 | 167 | 15 | 16 | 103 | GPIOA25/SIRQ1 |
| 3.3V | N/C | 17 | 18 | 149 | GPIOC6/DSI_DP0/UART2_RX/SPIO_MISO |
| GPIOC25/TWI3_SDATA/PCM0_SYNC/SPIO_MOSI | 53 | 19 | 20 | N/C | GND |
| GPIOC24/I2S_MCLK1/PCM0_IN/SPIO_MISO | 55 | 21 | 22 | 169 | GPIOC5/LCD0_D0/DSI_CN/SD1_D0 |
| GPIOC22/TWI3_SCLK/PCM0_CLK/SPIO_SCLK | 47 | 23 | 24 | 51 | GPIOC23/I2S_LRCLK1/PCM0_OUT/SPIO_SS |
| GND | N/C | 25 | 26 | 136 | GPIOB19/OAN/TS_START/LCD0_D15 |
| GPIOB16/OBP/TS_IN1/LCD0_D21 | 140 | 27 | 28 | 146 | GPIOB14/OCP/TS_IN0/LCD0_D23 |
| GPIOB15/OCN/TS_IN2/LCD0_D22 | 148 | 29 | 30 | N/C | GND |
| GPIOB10/OEP/TS_IN7/LCD0_DCLK0 | 158 | 31 | 32 | 154 | GPIOB13/ODN/TS_IN4/LCD0_VSYNC0 |
| GPIOB0/PCM0_OUT/I2S_BCLK1 | 9 | 33 | 34 | N/C | GND |
| GPIOB1/PCM0_CLK/I2S_LRCLK1 | 13 | 35 | 36 | 7 | GPIOA28/PCM0_IN/I2S_BCLK0 |
| GPIOB2/PCM0_SYNC/I2S_MCLK1 | 15 | 37 | 38 | 5 | GPIOA31/I2S_D1 |

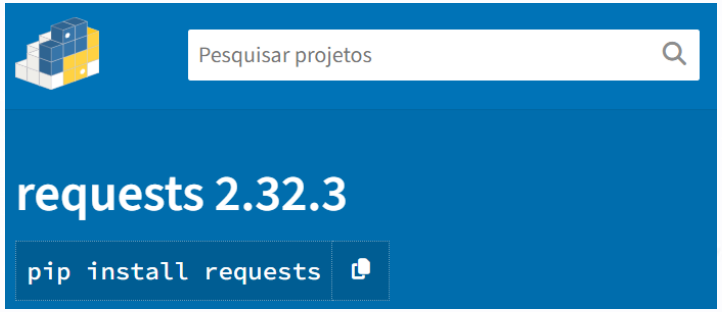


# Labrador 32 – Outras Comunicações

## Ethernet e Wi-Fi

- A Labrador suporta conexões via **Ethernet** e **Wi-Fi**, permitindo a comunicação com redes locais e a internet.
- Com essa conectividade, é possível interagir com sites, consumir APIs e integrar a placa em soluções de Internet das Coisas (IoT).
- Utilizando bibliotecas como requests em Python, pode-se enviar dados para servidores remotos ou obter informações de serviços web.

```
import requests
r = requests.get('https://httpbin.org/basic-auth/user/pass', auth=('user', 'pass'))
r.status_code
#200
r.headers['content-type']
#'application/json; charset=utf8'
r.encoding
#'utf-8'
r.text
#'{"authenticated": true, ...'
r.json()
#{'authenticated': True, ...}
```



# Labrador 32 – Outras Comunicações

## Conexão Bluetooth

- A Labrador possui suporte a Bluetooth, permitindo a comunicação sem fio com diversos dispositivos compatíveis.
- Essa funcionalidade é útil para conectar periféricos, como teclados, mouses, sensores e outros dispositivos Bluetooth, ampliando as possibilidades de interação.
- A inicialização do Bluetooth ocorre durante o boot do sistema, sendo possível ajustar seu comportamento alterando o script localizado em /etc/init.d/wifi_bt_start.sh

# Caninos SDK – Conclusão

- **Resumo dos Tópicos Abordados:**
  - Comunicação UART0 em Loopback;
  - Revisão de I2C e aplicação com sensor AHT10 de temperatura e humidade;
  - Apresentação do SPI, Ethernet, Wi-fi e Bluetooh

    https://github.com/jorgewattes/labrador-examples/tree/main/embarcatech
- **Próximos Passos:**
  - Modulação por Largura de Pulso
  - Servo Motor

# Labrador 32 – SPI:

```
from periphery import SPI

# Configurar o dispositivo SPI
spi = SPI("/dev/spidev0.0", 0, 500000)  # "/dev/spidev0.0" é o
dispositivo SPI, 0 é o modo, e 500000 é a frequência (500 kHz)

# Dados para enviar (deve ser um array de bytes)
data_to_send = [0x01, 0x02, 0x03]  # Exemplos de dados em hexadecimal

# Realizar a comunicação SPI (enviar e receber dados)
response = spi.transfer(data_to_send)

# Mostrar o resultado
print("Enviado:", data_to_send)
print("Recebido:", response)

# Fechar o dispositivo SPI
spi.close()
```

# Aplicações de SBCs