embarca tech | Residência Tecnológica em Sistemas Embarcados

Funcionalidades do SBC Labrador

Unidade 5 | Capítulo 1

Prof. Jorge Wattes

Executores: Instituto Federal Piauí; Instituto Federal Rio Grande do Norte; Instituto Federal Maranhão; Instituto Federal Ceará; cpedi; HBr

Coordenação: Softex

Iniciativa: Governo Federal Brasil União e Reconstrução

Sumário

Unidade 5

Capítulo 1

1. Boas-vindas e Introdução

Olá, estudantes! Sejam bem-vindos à nossa aula sobre o SBC Labrador, desenvolvido pelo projeto Caninos Loucos. Nessa aula nós iremos entender um pouco mais sobre este computador de placa única e sobre seus periféricos. Iremos aprender a realizar as configurações iniciais e como utilizar os periféricos de GPIO e comunicação, aplicando-os no uso de alguns tipos de sensores.

2. Objetivos Educacionais

Ao final desta aula, você será capaz de compreender as funcionalidades do SBC Labrador e suas aplicações em projetos embarcados.

3. Configurações Iniciais do Labrador

O uso de Single-Board Computers (SBCs) é uma excelente plataforma para aprendizado e desenvolvimento de projetos de automação, IoT e robótica. A SBC Labrador, da Caninos Loucos, é uma ferramenta acessível que permite a interação direta com o hardware e o controle de dispositivos externos. Neste e-book, vamos explorar o uso inicial da SBC Labrador, com exemplos de programação em Python, desde a configuração do ambiente de desenvolvimento até o uso de entradas e saídas digitais como LEDs e botões.

3.1 Instalando o Sistema Operacional

A SBC Labrador utiliza uma versão customizada do Linux Debian 11 disponível na área de downloads da página do projeto Caninos Loucos. A instalação do sistema operacional é realizada a partir de um cartão micro SD, onde a imagem deve ser gravada utilizando softwares específicos com o Win32 Disk Imager.

- Após gravar o cartão micro SD, realize os seguintes passos:
- Desenergize a Labrador e insira o cartão micro SD;
- Mantenha pressionado o **botão recovery** na placa Labrador Core;
- Com o **botão recovery** pressionado, energize a placa.

Figura 1: Relação entre a segurança Tradicional e IoT [1].
Fonte: Practical Internet of Things Security, página 6.

O sistema operacional será inicializado pelo cartão micro SD. Abra o Terminal e execute o comando **ls**, ele irá listar todos os elementos na raiz do diretório de armazenamento, você irá visualizar um arquivo denominado **install.sh**, execute o seguinte comando para realizar a instalação do sistema operacional na memória interna do Labrador: **sudo sh instal.sh**, inserindo a senha caninos para habilitar a função administrador (sudo). Ao concluir a instalação, você poderá desligar a Labrador, remover o cartão micro SD e voltar a energizar a Labrador para usá-lo normalmente.

/ INÍCIO ATENÇÃO /

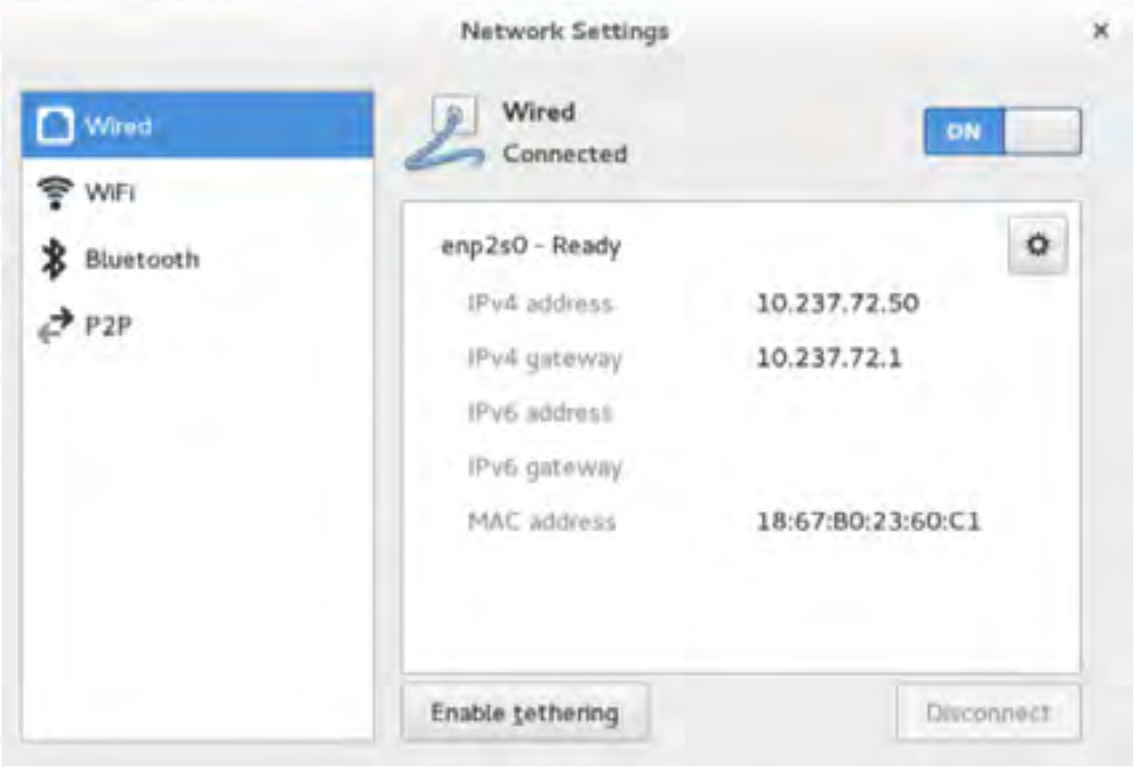
O usuário padrão é **caninos**, e as senha padrão também é **caninos**.

/ FIM ATENÇÃO /

3.2 Conectando à Rede

A Labrador pode ser conectada à rede local por cabo Ethernet ou via Wi-

Fi. Se for utilizar Wi-Fi, o programa de gerenciamento de redes pode ser acessado em **Iniciar > Preferencias > Connman Settings**. Ao executar este programa, um ícone de conexão à internet aparecerá no canto inferior direito da tela, próximo ao relógio. Com um clique duplo você acessará a interface e poderá configurar a conexão Wi-Fi desejada.



Caso queira, é possível configurar as credenciais de rede no arquivo de configuração, tipicamente localizado em /etc/wpa_supplicant/wpa_supplicant.conf. Aqui está um exemplo de configuração para Wi-Fi:

```
network={
  ssid="Seu_SSID"
  psk="Sua_Senha"
}
```

3.3 Preparando o Ambiente de Desenvolvimento

Antes de começar a programar, é importante configurar o ambiente corretamente para utilizar a SBC Labrador. Para isso, é importante garantir que a tanto a versão do Python quanto do pip estejam atualizadas. Através do pip é possível instalar facilmente algumas bibliotecas fundamentais para o uso do Labrador. Para atualizar o sistema, execute os comandos de atualização do programa apt e tente instalar o python3 e o pip3.

```
sudo apt update && sudo apt upgrade
sudo apt-get install python3 python3-pip
```

Através do pip3 é possível instalar novas bibliotecas na Labrador com o seguinte comando:

```
sudo pip3 install gpiod
```

A seguir, são apresentadas algumas das principais bibliotecas a serem utilizados na Labrador.

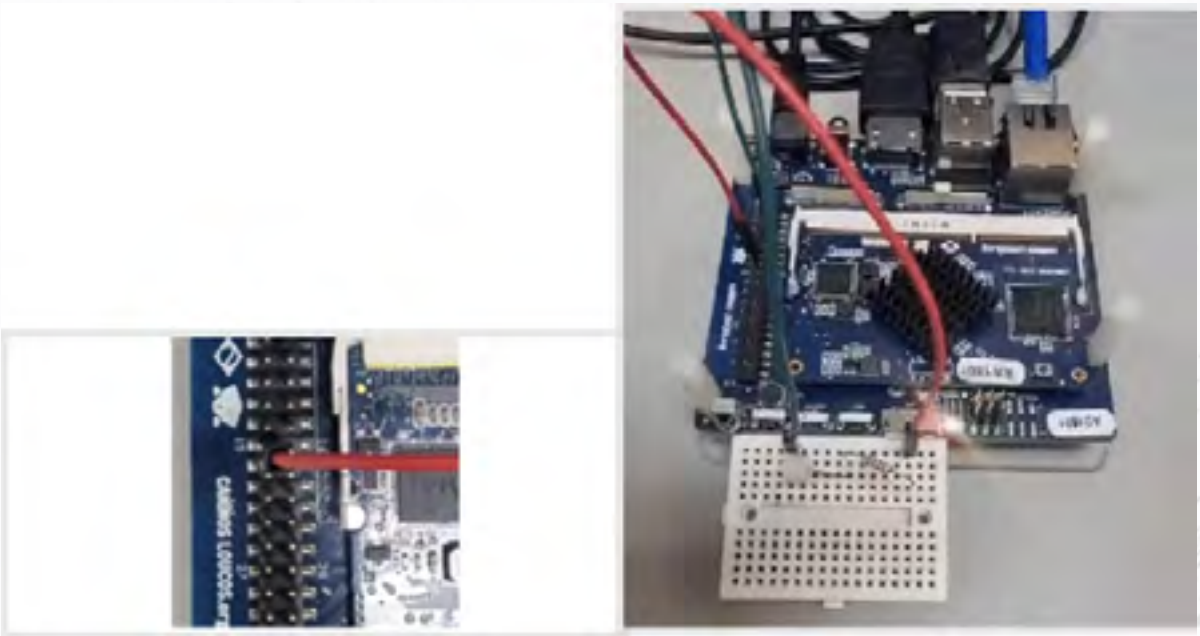
Biblioteca	Descrição	Repositório
gpiod	Biblioteca para gerenciamento genérico dos pinos de entradas e saídas digitais de hardwares Linux que possuam GPIO.	https://pypi.org/project/gpiod/
caninos-sdk	Biblioteca desenvolvida pelo projeto Caninos Loucos com o objetivo de tornar o uso das placas do projeto, incluindo a Labrador, muito mais simples e acessível.	https://github.com/caninos-loucos/caninos-sdk
OpenCV		https://pypi.org/project/opencv-python/
pylibi2c		https://github.com/amaork/libi2c

3.4 Testando a Instalação

Inicie o Terminal dentro da Labrador e, execute o comando nano para iniciar o aplicativo de escrita textual nativo do Debian 11. Redija o seguinte código no nano e em seguida pressione Ctrl + O para salvar, informando o nome teste.py e em seguida, Ctrl + Z para finalizar o nano.

```
import gpiod
#esta biblioteca permite gerar uma pausa na labrador (um delay)
import time
#criamos nossa variável chip, para uso dos recursos da placa
chip = gpiod.chip('/dev/gpiochip2')
led = chip.get_line(4)
#configurando a requisição
config = gpiod.line_request()
config.consumer = "led"
#ativamos o pino 15 (da labrador), como saída e definimos um apelido para ele
config.request_type = gpiod.line_request.DIRECTION_OUTPUT
led.request(config)
#um laço de repetição para 6 iterações
while(1):
    #na labrador, acessamos o LED habilitado como saída (led)
    #enviamos então, um sinal de 1 para o pin15
    led.set_value(1)
    #uma pausa de meio segundo
    time.sleep(0.5)
    #enviamos um sinal de 0 para o pin15
    led.set_value(0)
    time.sleep(0.5)
```

Conecte o catodo de um led ao **pino 4** do header e um resistor em série com o anodo do LED ao **pino 15**. Em seguida, execute o seguinte comando para executar o código através do Python: python3 teste.py.



Se você conseguiu fazer o LED piscar 1 vez por segundo, parabéns! Agora você pode explorar um pouco mais o código para tentar entendê-lo.

4. Utilizando as GPIO

Agora que já confirmamos que a placa Labrador está em correta operação, vamos começar a entender um pouco mais, como funcionam as GPIO dela e as funções presentes nas bibliotecas gpiod e caninos-sdk.

O barramento de pinos, também conhecido como header, presente na placa base da Labrador apresenta ao todo 40 pinos. Destes, 12 são de alimentação, sendo 8 GND, 2 de 3,3V e 2 de 5V. Essas alimentações devem ser utilizadas com cuidado, já que não possuem alta capacidade de corrente e deverão alimentar apenas pequenos sensores, **nada de ligar motores**! Por outro lado, o GND deve sempre ser utilizado na equipotencialização entre dispositivos, ou seja, ao conectar, por exemplo, duas placas para fazerem comunicação Rx/Tx, você precisará conectar também um GND para igualar a referência de 0V de ambas as placas.

Observando os pinos, vemos que eles apresentam diversas funções, por exemplo, o pino 10 pode ser utilizado tanto como o GPIO C26, quanto o Master Input Slave Output do SPI1, quanto o Sdata do I2C0, ou ainda, como o Rx da UART0. Qual dessas funções será utilizada no pino 10, depende da programação específica desse pino.

Na figura abaixo temos todos os pinos do header da Labrador, juntamente com todas as suas possíveis funções. É importante perceber que no grupo de GPIO todos os pinos são nomeados num padrão Letra + número. No exemplo do **pino 10**, o GPIO é o **C26**. A letra do GPIO indica em que grupo o pino está, e o número é o número do pino dentro daquele grupamento.

Voltando ao exemplo do item 4.4, temos que no pino 15, o GPIO é o C4. O grupo C é definido através do gpiod.chip('/dev/gpiochip2'), para cada grupo, utiliza-se um gpiochip diferente.

Grupo	gpiochip
A	gpiochip0

B	gpiochip1
C	gpiochip2
D	gpiochip3
E	gpiochip4

Por sua vez, a função, chip.get_line(), que define a variável led, estabelece que, dentro do grupo 2 (C), seja utilizado a saída 4, caracterizando led como o pino GPIOC4.

FUNC	SODIMM	HEADER		SODIMM	FUNC
3.3V	N/C	1	2	N/C	5V
GPIOI3/TWI_SDATA	41	3	4	N/C	5V
GPIOI2/TWI2_SCLK	43	5	6	N/C	GND
GPIOB18/DAP/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/RS_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/SPI0_MISO
GPIOC25/TWI3_SDATA/PCM0_SYNC/SPI0_MOSI	53	19	20	N/C	GND
GPIOC24/I2S_MCLK1/PCM0_IN/SPI0_MISO	55	21	22	169	GPIOC5/LCD0_D0/DSI_CN/SD1_D0
GPIOC22/TWI3_SCLK/PCM0_CLK/SPI0_SCLK	47	23	24	51	GPIOC23/I2S_LRCLK1/PCM0_OUT/SPI0_SS
GND	N/C	25	26	136	GPIOB19/OAN/TS_START/LCD0_D15
GPIOB16/DBP/TS_IN1/LCD0_D21	140	27	28	146	GPIOB14/DCP/TS_IN0/LCD0_D23
GPIOE15/OCN/TS_IN2/LCD0_D22	148	29	30	N/C	GND
GPIOB10/ODP/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	11	35	36	7	GPIOA28/PCM0_IN/I2S_BCLK0
GPIOB2/PCM0_SYNC/I2S_MCLK1	15	37	38	5	GPIOA31/I2S_D1
GND	N/C	39	40	3	GPIOA27/I2S_D0

5. LEDs como Saídas Digitais

5.1 Programando um LED com a Labrador

Agora que o ambiente está configurado, vamos aprender a programar um LED simples com Python3 utilizando a biblioteca gpiod.

5.2 Entendendo os Pinos GPIO

A SBC Labrador possui pinos GPIO (General Purpose Input/Output) que permitem a comunicação com dispositivos externos. Para controlar um LED, utilizaremos um desses pinos para enviar sinais digitais (ligado ou

desligado) ao LED.

5.3 Ligando o LED

- **Conectando o LED ao Pino GPIO**: Para o exemplo, conecte o anodo do LED ao pino GPIO 17 e o catodo ao GND. É recomendável utilizar um resistor de 330 ohms em série com o LED para evitar danos ao componente.
- Controlando o LED com Python:

Aqui está um exemplo simples de código Python para acender e apagar o LED:

```
import gpiod
import time

# Acessa o chip GPIO 0
chip = gpiod.Chip('gpiochip0')

# Configura o pino 17 como saída
line = chip.get_line(17)
line.request(consumer='led', type=gpiod.LINE_REQ_DIR_OUT)

# Liga o LED
line.set_value(1)
time.sleep(1)

# Desliga o LED
line.set_value(0)
time.sleep(1)

# Limpa a configuração do pino
chip.close()
```

5.4 Explicação do Código

- **gpiod.Chip('gpiochip0')**: Acessa o chip GPIO. O gpiochip0 é o nome do chip utilizado, mas em alguns casos pode variar.
- **line = chip.get_line(17)**: Obtém o controle sobre o pino 17.
- **line.request(...)**: Solicita o uso do pino, configurando-o como saída.
- **line.set_value(1)**: Liga o LED (sinal de 3,3V).
- **time.sleep(1)**: Aguarda 1 segundo.
- **chip.close()**: Libera o controle do chip GPIO.

6. Botões como Entradas Digitais

Agora que sabemos controlar um LED, vamos aprender a controlar esse LED com a ajuda de um botão.

6.1 Ligando o Botão

Para controlar o LED com o botão, conecte o botão entre o pino GPIO 27 e o GND. É importante utilizar um resistor de pull-down para garantir que o pino seja lido como LOW quando o botão não estiver pressionado.

6.2 Detectando o Pressionamento do Botão

No código a seguir, vamos ler o estado do botão e acionar o LED quando o botão for pressionado.

```
import gpiod
import time

# Acessa o chip GPIO 0
chip = gpiod.Chip('gpiochip0')

# Configura os pinos do LED e do botão
led_line = chip.get_line(17)
button_line = chip.get_line(27)

# Solicita o uso dos pinos
led_line.request(consumer='led', type=gpiod.LINE_REQ_DIR_OUT)
button_line.request(consumer='button', type=gpiod.LINE_REQ_DIR_IN, default_
val=0)

try:
  while True:
    # Lê o estado do botão
    button_state = button_line.get_value()

    if button_state == 1:
      # Se o botão for pressionado, liga o LED
      led_line.set_value(1)
    else:
      # Se o botão não for pressionado, desliga o LED
      led_line.set_value(0)
    time.sleep(0.1)
except KeyboardInterrupt:
  # Limpa a configuração dos pinos
  chip.close()
```

6.3 Explicação do Código

- **button_line = chip.get_line(27)**: Obtém o controle sobre o pino 27 para o botão.
- **button_line.request(...)**: Solicita o uso do pino 27 como entrada, com valor padrão 0 (quando o botão não for pressionado).
- button_state = button_line.get_value(): Lê o estado do botão.
- **led_line.set_value(1)**: Liga o LED no pino 17 quando o botão for pressionado.

7. Conclusão

Neste e-book, exploramos os primeiros passos no uso da SBC Labrador com Python3 e a biblioteca gpiod, abordando desde a preparação do ambiente até a programação de um LED e controle via botão. A SBC Labrador oferece uma plataforma simples e poderosa para quem deseja aprender sobre hardware e software, permitindo que os desenvolvedores criem projetos interativos com facilidade.

Com as informações e exemplos aqui fornecidos, você pode expandir seus conhecimentos e começar a desenvolver projetos mais complexos, envolvendo sensores, atuadores e outros componentes.

Referências

CANINOS LOUCOS. Guia 2 - Programando um LED com a Labrador. Disponível em: https://wiki.caninosloucos.org. Acesso em: 03 dez. 2024.

https://github.com/caninos-loucos/caninos-sdk

https://github.com/caninos-loucos/caninos-sdk/tree/main/examples

https://wiki.caninosloucos.org/index.php/Guia_4_-_Controlando_um_LED_por_PWM

https://wiki.caninosloucos.org/index.php/Guia_5_-_Enviando_dados_da_Labrador_para_o_Arduino_com_I2C


embarca
tech
Residência
Tecnológica
em Sistemas
Embarcados