Resumo da Atividade Prática

Parte 1: Python e RPi.GPIO
A linguagem **Python** é amplamente usada em sistemas embarcados devido à sua simplicidade e poder de abstração. A biblioteca **RPi.GPIO** oferece uma interface para controle dos pinos GPIO (General Purpose Input/Output) do Raspberry Pi, permitindo configurá-los como entrada ou saída para a leitura de sensores e controle de atuadores, como LEDs e motores. Essa interface facilita a comunicação com o hardware, possibilitando o desenvolvimento de sistemas interativos e automatizados.

Parte 2: GPIO Zero, RPi.GPIO e Controle de Periféricos
A biblioteca **GPIO Zero** foi desenvolvida para simplificar o uso dos GPIOs em Python, abstraindo operações mais complexas que a biblioteca RPi.GPIO necessita. Na prática, controlamos dispositivos externos conectados aos pinos GPIO, como LEDs e servomotores, por meio de sinais digitais e PWM. Além disso, exploramos a comunicação por protocolos de interface como **SPI**, **I2C** e **UART**, que são amplamente usados em sistemas embarcados para a comunicação com periféricos como displays, sensores e módulos de comunicação.

Definições:

- **PWM (Pulse Width Modulation)**: Técnica de controle que varia a largura de pulso de um sinal digital para ajustar a potência fornecida a um dispositivo (por exemplo, controlar a velocidade de um motor).
- **SPI (Serial Peripheral Interface)**: Protocolo de comunicação síncrona usado para transferir dados entre microcontroladores e periféricos.
- **I2C (Inter-Integrated Circuit)**: Protocolo de comunicação que permite conectar vários dispositivos usando apenas dois fios.
- **UART (Universal Asynchronous Receiver/Transmitter)**: Protocolo de comunicação serial usado para transmissão de dados assíncronos.
- **GPIO.BCM/GPIO.BOARD**: Modo de numeração dos pinos da Raspberry Pi. O BCM segue a numeração do processador Broadcom, enquanto o BOARD segue a numeração física dos pinos no conector.

Parte 3: Computação Paralela em Sistemas Embarcados
Nesta parte, estudamos **computação paralela**, que permite que várias tarefas sejam executadas simultaneamente, aproveitando os múltiplos núcleos da CPU. Utilizamos conceitos de **concorrência** e **escalonamento** de processos em um sistema operacional Linux embarcado no Raspberry Pi. Isso é fundamental em sistemas de tempo real ou de alta performance, onde é necessário executar várias operações ao mesmo tempo, como controlar múltiplos sensores ou realizar cálculos complexos.

Definições:

- **Processos**: qualquer programa rodando em um sistema operacional possui um mais processos associados
- **Threads**: Unidades menores dentro de um processo que podem ser executadas em paralelo.
- **Concorrência**: Capacidade de um sistema de lidar com várias tarefas ao mesmo tempo, alternando entre elas de forma eficaz.
- **Escalonamento**: Mecanismo do sistema operacional para decidir qual processo ou thread será executado em cada núcleo de processamento.
- **Multicore**: CPUs com mais de um núcleo de processamento, permitindo a execução real de várias threads simultaneamente.