**Digital Badge (DB4.0)**

**Autor: Dr. Marlon da Silva Garrido**
**Professor Associado IV (CENAMB - PPGEA)**
**Universidade Federal do Vale do São Francisco (UNIVASF)**

**Explicações Básicas das Interações e Funcionamento do código Real com Periféricos.**

**Bibliotecas Utilizadas Básicas**

- **pico/stdlib.h:**
  Fornece funções básicas para inicialização, delays, e gerenciamento do sistema no Raspberry Pico.
- **hardware/i2c.h:**
  Usada para configurar e utilizar a interface I2C, essencial para a comunicação com o display OLED e o MPU6050.
- **hardware/uart.h:**
  Responsável pela comunicação serial via UART, utilizada nos módulos GPS e LoRa.
- **hardware/gpio.h:**
  Utilizada para configurar os pinos de entrada e saída, incluindo botões, LEDs e buzzer.
- **hardware/pwm.h:**
  Caso seja necessário utilizar PWM (por exemplo, para controlar intensidade de algum sinal), embora neste exemplo o buzzer seja acionado diretamente via GPIO.

**1. Arquivo (main.c)**

**Objetivo:**
Este é o arquivo principal que orquestra a inicialização dos periféricos, a leitura dos sensores e módulos, e o controle das ações do sistema (por exemplo, enviar dados via LoRa, atualizar o display OLED, emitir alertas e monitorar botões).

**Principais responsabilidades e funções:**

- **Inclusões de Bibliotecas:**
  São incluídos os cabeçalhos do SDK do Pico (pico/stdlib.h, hardware/*), além dos arquivos de cabeçalho específicos para o display (ssd1306.h), fonte (fonte.h), GPS (gps.h), LoRa (lora.h) e acelerômetro/giroscópio (mpu6050.h).
- **Configuração dos Periféricos:**
  - **GPIO:** Configura os pinos de saída para os LEDs e buzzer; os pinos de entrada para os botões (com pull-up).

    - **I2C:** Inicializa a interface I2C para comunicação com o display OLED (SSD1306) e o MPU6050.
    - **UART:** Configura duas interfaces UART:
        - Uma para o módulo GPS (por exemplo, usando UART0 com taxa de 9600 bauds).
        - Outra para o módulo LoRa (por exemplo, usando UART1 também com 9600 bauds).
- **Inicialização dos Dispositivos:**
    - Chama a função de inicialização do display OLED (ssd1306_init).
    - Inicializa o acelerômetro/giroscópio (mpu6050_init).
    - Inicializa as UARTs para o GPS e o LoRa.
- **Loop Principal:**
  Dentro do loop, o código executa continuamente as seguintes tarefas:
    - **Leitura do GPS:** Chama gps_read() para obter dados do módulo GPS via UART.
    - **Leitura do MPU6050:** Chama mpu6050_read_accel() para ler os valores de aceleração dos eixos X, Y e Z. Se for detectado movimento (comparando com um limiar), atualiza o tempo do último movimento e desativa os alertas (LEDs, display e buzzer).
    - **Verificação de Inatividade:** Calcula o tempo decorrido desde o último movimento e, conforme o período de inatividade (5, 10 ou 15 minutos), aciona:
        - Piscar do LED azul (5 minutos).
        - Piscar do LED vermelho e exibir a mensagem "ATENCAO" no display (10 minutos).
        - Ativação do buzzer (15 minutos), que só é desativado quando o Botão A é pressionado.
    - **Transmissão via LoRa:** A cada 2 minutos, monta uma mensagem com os dados do GPS e chama lora_send() para enviá-la.
    - **Modo Emergência:** Se o Botão B for pressionado, imprime uma mensagem de emergência com a localização no monitor serial e envia a mesma mensagem via LoRa.
    - **Tratamento de Botões:** Um callback (button_a_handler) é usado para resetar alertas ao pressionar o Botão A.

**2. Arquivos (gps.h e gps.c)**

**Objetivo:**
Gerenciar a leitura dos dados do módulo GPS, que envia sentenças NMEA via UART.

**Função Principal:**

- **bool gps_read(uart_inst_t *uart, char *buffer, size_t len)**
  Essa função verifica se há dados disponíveis na UART conectada ao GPS, lê os caracteres (até encontrar uma quebra de linha ou atingir o tamanho do buffer) e armazena no buffer fornecido. Retorna true se algum dado foi lido.
    - **Bibliotecas Utilizadas:**
        - hardware/uart.h para funções de leitura da UART.
        - pico/stdlib.h para funções básicas (como delays, se necessário).

**3. Arquivos (lora.h e lora.c)**

**Objetivo:**
Gerenciar o envio de mensagens via o módulo LoRa, que utiliza uma interface UART para comunicação.

**Função Principal:**

- **void lora_send(uart_inst_t *uart, const char *message)**
  Essa função utiliza a função uart_puts() para enviar a mensagem via a UART configurada para o LoRa. Adiciona uma quebra de linha após a mensagem para facilitar a leitura no receptor.
    - **Bibliotecas Utilizadas:**
        - hardware/uart.h para as funções de envio de dados via UART.

**4. Arquivos (mpu6050.h e mpu6050.c)**

**Objetivo:**
Fornecer funções para inicialização e leitura dos dados de aceleração do sensor MPU6050, que é utilizado para detectar movimento.

**Funções Principais:**

- **bool mpu6050_init(mpu6050_t *mpu, i2c_inst_t *i2c, uint8_t addr)**
  Inicializa o MPU6050 enviando comandos via I2C para configurar o registrador de gerenciamento de energia (por exemplo, despertando o sensor). Retorna true se a escrita via I2C foi bem-sucedida.
- **bool mpu6050_read_accel(mpu6050_t *mpu, float *ax, float *ay, float *az)**
  Lê os 6 bytes de dados de aceleração dos registradores do MPU6050, converte os valores brutos em valores em “g” (usando a sensibilidade típica de 16384 LSB/g para o modo ±2g) e armazena nos ponteiros fornecidos. Retorna true se a leitura foi bem-sucedida.
    - **Bibliotecas Utilizadas:**
        - hardware/i2c.h para comunicação I2C.
        - pico/stdlib.h para funções auxiliares e delays.

**5. arquivos (ssd1306.h e ssd1306.c)**

**Objetivo:**
Gerenciar a comunicação com o display OLED SSD1306 via I2C e fornecer funções para desenhar textos e gráficos no display.

**Funções Principais:**

- **void ssd1306_init(ssd1306_t *dev, i2c_inst_t *i2c, uint8_t addr, uint8_t width, uint8_t height)**

  Inicializa o display enviando uma sequência de comandos via I2C (por exemplo, desligar o display, configurar os parâmetros de clock, multiplexação, contraste, etc.) e configura a estrutura de dados do display (incluindo o buffer de vídeo).

- **void ssd1306_clear(ssd1306_t *dev)**
  Limpa o buffer do display (preenchendo com zeros).

- **void ssd1306_show(ssd1306_t *dev)**
  Envia o conteúdo do buffer via I2C para atualizar o display.

- **void ssd1306_draw_string(ssd1306_t *dev, uint8_t x, uint8_t y, const char *str)**

  Desenha uma string no buffer do display na posição (x,y). Nesta implementação simplificada, a função pode ser um stub, dependendo de como a fonte (em fonte.h) será aplicada para desenhar cada caractere.

- **void ssd1306_draw_border(ssd1306_t *dev, int thickness)**
  Desenha uma borda de espessura definida ao redor do display, modificando o buffer de vídeo.

    - **Bibliotecas Utilizadas:**

        - hardware/i2c.h para a comunicação I2C.
        - pico/stdlib.h e funções de manipulação de memória para trabalhar com o buffer.

**6. Arquivo (fonte.h)**

**Objetivo:**
Fornecer os bitmaps de fontes para exibição de caracteres no display OLED. Aqui são definidos arrays para:

- **Caracteres maiúsculos (A-Z)**
- **Caracteres minúsculos (a-z)**
- **Números (0-9)**
- **Caracteres estendidos (acentuação e cedilha) para versões maiúsculas e minúsculas**

Esses arrays contêm os dados de 5 bytes por caractere, representando cada coluna de pixels para um caractere 5x7. Essa definição será utilizada pela função ssd1306_draw_string() para copiar os bits do caractere desejado para o buffer do display.

**Interações e Fluxo de Funcionamento Geral**

1. **Inicialização:**
    - O main.c inicia chamando stdio_init_all() e configurando os pinos (GPIO, I2C, UART).
    - O display OLED é inicializado via ssd1306_init(), e o MPU6050 é inicializado via mpu6050_init().
    - As UARTs são configuradas para o GPS e LoRa.
2. **Loop Principal (main.c):**
    - **Leitura dos Sensores:**
        - O módulo GPS é lido com gps_read(), que retorna uma sentença NMEA (ou parte dela) para ser utilizada nas mensagens.
        - O MPU6050 é lido com mpu6050_read_accel(), permitindo detectar se há movimento.
    - **Controle de Inatividade e Alertas:**
        - Se o sensor detectar ausência de movimento por períodos definidos (5, 10 ou 15 minutos), o código ativa alertas visuais (LEDs e atualização do display com "ATENCAO") e um alerta sonoro (buzzer).
        - O tempo do último movimento é atualizado sempre que o acelerômetro detecta movimento.
    - **Envio de Dados via LoRa:**
        - A cada 2 minutos, os dados do GPS são enviados através de lora_send() para outra base.
    - **Tratamento dos Botões:**
        - O Botão A é configurado com uma interrupção que chama button_a_handler(), resetando os alertas e desligando o buzzer.
        - O Botão B é verificado periodicamente para disparar o modo de emergência, que envia a localização tanto pelo monitor serial quanto via LoRa.
3. **Renderização no Display:**
    - A função ssd1306_draw_string() (em conjunto com os dados de fonte.h) é utilizada para desenhar textos (como “ATENCAO” ou os dados do GPS) no buffer do display.
    - Após as alterações no buffer, ssd1306_show() atualiza o display.
4. **Comunicação via UART:**
    - O GPS e o LoRa comunicam-se via UART. As funções gps_read() e lora_send() abstraem a leitura e o envio de mensagens, respectivamente.

A estrutura modular permite que cada componente (GPS, LoRa, MPU6050, Display) seja desenvolvido e testado individualmente, facilitando a manutenção e a expansão do projeto. As funções dos arquivos de código se comunicam por meio de chamadas diretas no main.c, criando um fluxo de dados que vai desde a captura dos sinais dos sensores até a exibição e transmissão das informações.