Smart Crossing

Semáforo Inteligente com Acessibilidade via BLE

Integrantes: Nícolas Marçal, Vinícius E. Mantovani

Projeto: Embarcatech - Residência Tecnológica

Contexto

O Desafio da Travessia Urbana e Nossa Proposta

O Problema

- Semáforos com **tempos fixos** são ineficientes e perigosos.
- Causam atrasos e riscos desnecessários.
- Pedestres com **mobilidade reduzida** podem não ter tempo suficiente para atravessar com segurança.

A Solução: Smart Crossing

- Sistema embarcado para uma travessia **segura e adaptável**.
- Detecta a presença de pedestres para otimizar o fluxo.
- Aplicativo mobile solicita **tempo extra** de forma automática e personalizada.

Arquitetura Geral do Sistema

Dois elementos principais em comunicação sem fio.

Hardware

- **Cérebro:** Placa BitDogLab.
- **Comunicação:** Módulos Wi-Fi e Bluetooth Low Energy integrados.
- **Sensor:** VL53L0X para detectar a presença de pedestres.
- **Atores:** Controla LEDs e exibe informações em um Display OLED.

Software

- **Interface:** Aplicativo Android para o usuário.
- **Função:** Detecta a proximidade com o semáforo e envia automaticamente o comando para aumentar o tempo de travessia.

Arquitetura de Firmware

Gerenciamento com FreeRTOS para execução concorrente.

traffic_lights_task: O núcleo do sistema. Uma máquina de estados que controla a sequência e os tempos dos semáforos.

sensor_task: Lê o sensor VL53L0X e notifica a `traffic_lights_task` ao detectar um pedestre.

display_task: Gerencia todas as atualizações no display OLED.

ap_task / sta_task: Responsáveis pela comunicação Wi-Fi (UDP) e sincronização entre as unidades.

Implementando o Bluetooth (BLE)

Configurando a Pico W para atuar como um Servidor BLE (Periférico).

Para que o semáforo "converse" com o celular, utilizamos:

- **Framework:** BTstack, a pilha de software Bluetooth para sistemas embarcados.
- **Objetivo:** Fazer a placa anunciar sua presença, aceitar conexões e receber dados.
- **Arquivos-Chave:**

  `btstack_config.h` : Define funcionalidades e otimiza o uso de memória.

  `ble_connection.c` : Contém a lógica de comportamento (anúncio, conexão, etc).

Lógica de Conexão e Serviço GATT

O "coração" da lógica BLE em `ble_connection.c`

1. **Anúncio (Advertising):**

   A placa se torna detectável com o nome **"Pico-BLE"** e anuncia um Serviço Customizado para ser encontrada pelo app.

2. **Gerenciamento de Eventos (packet_handler):**

   Uma função central que reage a eventos como conexão, desconexão e status do Bluetooth.

3. **Serviço GATT (O "Contrato" de Comunicação):**

   - **Serviço Customizado (UUID 0xFFF0):** Agrupa as funcionalidades.
   - **Característica de Escrita (UUID 0xFFF1):** Canal por onde o app envia dados.
   - Quando o app escreve um valor, a função `att_write_callback` na placa é acionada.

Análise de Código: Máquina de Estados

O cérebro do sistema em `traffic_lights_task.c`

```
switch (traffic_state) {
  case PEOPLE_GREEN:
    led_green_on();
    vTaskDelay(get_time_green(PEOPLE));
    ...
  break;
  case CAR_YELLOW:
    ...
}
```

- A variável `traffic_state` armazena o estado atual.
- Funções como `led_green_on()` controlam os LEDs.
- `vTaskDelay()` controla o tempo de cada estado.

Comunicação e Sincronização

Wi-Fi/UDP para sincronia entre as placas.

Configuração da Rede

- **ap_task.c**: Configura uma placa como Access Point (SSID: "RP_AP").
- **sta_task.c**: Configura a outra placa para se conectar a essa rede.

Troca de Mensagens

- **udp_init_recv():** Inicializa um servidor UDP na porta 4444.
- **udp_send_message():** Envia mensagens de texto simples como "start", "ack", "pd".
- **udp_recv_function():** Processa mensagens recebidas e aciona ações nas tarefas correspondentes.

Arquitetura do App Smart Crossing

Projetado para ser autônomo e eficiente, operando em segundo plano.

MainActivity

Interface do usuário para definir nível de mobilidade e ativar o serviço.

ForegroundService

O cérebro do app. Roda em segundo plano para escanear e conectar via BLE.

BroadcastReceivers

"Ouvintes" que reagem a eventos do sistema para automatizar o serviço.

Lógica de Proximidade e Ação Automática

O ForegroundService age baseado na proximidade, não na interação do usuário.

Zona Longe: Continua buscando por semáforos em baixo consumo.

Zona Perto: Alerta o usuário na notificação: "Aproximando...".

Zona Imediata: A ação é disparada!

- **Automaticamente** para o scan, conecta, envia o nível de mobilidade e desconecta.
- Tudo em menos de um segundo.
- Um **cooldown** de 30s evita envios repetidos.

Smart Crossing