De: Ravi de Farias

Para: Prof. Carlos Henrique Barriquello

Disciplina: UFSM00292 - Projeto de Sistemas Embarcados

Data: 26 de novembro de 2025

Assunto: Anexo Técnico - Código Fonte Final da Integração HMI (SAM R21)

1. Introdução

Este documento apresenta a listagem completa dos códigos fonte utilizados para a integração bem-sucedida entre a placa **SAM R21 Xplained Pro**, o **Display OLED1 Xplained Pro** e o **Teclado Matricial 3x4**. O firmware foi desenvolvido sobre o Zephyr RTOS, utilizando uma arquitetura baseada em interrupções para garantir a eficiência do processamento.

2. Arquivo de Configuração do Kernel (prj.conf)

Localização: Raiz do projeto

Este arquivo define os subsistemas do Zephyr que devem ser compilados. A configuração foi otimizada para evitar conflitos de versões, habilitando apenas os drivers essenciais para GPIO, comunicação I2C, controle do Display e subsistema de Entrada (Input).

Properties
```
# Habilita os barramentos de hardware essenciais
CONFIG_GPIO=y
CONFIG_I2C=y

# Habilita o subsistema de Display e o driver específico do controlador OLED
CONFIG_DISPLAY=y
CONFIG_SSD1306=y

# Habilita o subsistema de Input para processamento do Teclado
CONFIG_INPUT=y

# Habilita Logs para auxílio na depuração via serial
CONFIG_LOG=y
```

3. Descrição de Hardware (app.overlay)

Localização: Raiz do projeto

Este arquivo descreve a topologia física do hardware. A principal alteração de engenharia realizada está refletida aqui: o mapeamento das linhas do teclado foi movido para os pinos **PA08-PA11** para resolver o conflito elétrico com o barramento I2C do display (que reside nos pinos PA16/PA17).

DTS

```
/* app.overlay - Configuração Final para SAM R21 (Sem Conflitos) */
/ {
    aliases {
        kpad = &keypad;
    };

    keypad: keypad {
        compatible = "gpio-kbd-matrix";
        label = "Teclado 3x4";

        /* CONFIGURAÇÃO DAS LINHAS (ROWS) */
        /* Conectadas em PA08, PA09, PA10, PA11 para liberar o barramento I2C */
        /* Configurado com Pull-Up interno para dispensar resistores externos */
        row-gpios = <&porta 8  (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>,
                    <&porta 9  (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>,
                    <&porta 10 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>,
                    <&porta 11 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>;

        /* CONFIGURAÇÃO DAS COLUNAS (COLS) */
        /* Conectadas em PA04, PA05, PA06 */
        col-gpios = <&porta 4 GPIO_ACTIVE_LOW>,
                    <&porta 5 GPIO_ACTIVE_LOW>,
                    <&porta 6 GPIO_ACTIVE_LOW>;

        /* Configuração de Debouncing para estabilidade do sinal */
        debounce-down-ms = <10>;
        debounce-up-ms = <10>;
    };
};

/* Configuração do Display OLED no conector EXT1 */
/* O driver utiliza automaticamente o SERCOM1 (I2C) nos pinos PA16 e PA17 */
&sercom1 {
    status = "okay";
    compatible = "atmel,sam0-i2c";
    clock-frequency = <I2C_BITRATE_FAST>;

    ssd1306: ssd1306@3c {
        compatible = "solomon,ssd1306fb";
        reg = <0x3c>; /* Endereço I2C físico do dispositivo */
        width = <128>;
        height = <32>;
        segment-offset = <0>;
        page-offset = <0>;
        display-offset = <0>;
        multiplex-ratio = <31>;
        prechargep = <0x22>;
```

```
    };
};
```

4. Lógica da Aplicação (src/main.c)

Localização: Pasta src/

O código principal implementa a lógica da HMI. Ele inicializa os periféricos e registra uma função de *callback* que é executada assincronamente sempre que uma tecla é pressionada, desenhando uma barra visual no display proporcional ao valor da tecla.

C

```
#include <zephyr/kernel.h>
#include <zephyr/device.h>
#include <zephyr/drivers/display.h>
#include <zephyr/input/input.h>
#include <stdio.h>
#include <string.h>

const struct device *display = DEVICE_DT_GET(DT_NODELABEL(ssd1306));
uint8_t screen_buf[512]; // Buffer de memória para os pixels da tela

// Função auxiliar: Limpa a tela preenchendo o buffer com 0x00 (preto)
void clear_screen() {
    struct display_buffer_descriptor desc;
    desc.buf_size = sizeof(screen_buf);
    desc.width = 128;
    desc.height = 32;
    desc.pitch = 128;
    memset(screen_buf, 0x00, sizeof(screen_buf));
    display_write(display, 0, 0, &desc, screen_buf);
}

// Função auxiliar: Desenha uma barra horizontal de tamanho variável
void draw_bar(int size) {
    struct display_buffer_descriptor desc;
    desc.buf_size = 128; // Define buffer para 1 linha (página do OLED)
    desc.width = 128;
    desc.height = 8;     // Altura de 8 pixels
    desc.pitch = 128;

    uint8_t line[128];
    memset(line, 0x00, 128); // Limpa a linha

    // Preenche 'size' pixels com branco (0xFF)
    for(int i=0; i<size && i<128; i++) {
        line[i] = 0xFF;
    }
```

```
    // Envia o comando de escrita para o display (coordenadas 0,0)
    display_write(display, 0, 0, &desc, line);
}

// Callback de Interrupção: Executado automaticamente no evento de tecla
static void keypad_callback(struct input_event *evt, void *user_data)
{
    // Verifica se o evento é "Pressionar" (valor 1)
    if (evt->value == 1) {
        printk("Evento de Tecla Detectado: Código %d\n", evt->code);

        // Lógica de Feedback Visual:
        // Desenha uma barra proporcional ao código da tecla pressionada.
        // Ex: Tecla 0 gera barra pequena, Tecla 9 gera barra grande.
        int bar_size = (evt->code + 1) * 10;

        // Proteção contra buffer overflow visual
        if (bar_size > 128) bar_size = 128;

        draw_bar(bar_size);
    }
}

// Macro do Zephyr para registrar o callback do dispositivo de entrada
INPUT_CALLBACK_DEFINE(DEVICE_DT_GET(DT_NODELABEL(keypad)),
keypad_callback, NULL);

int main(void)
{
    // Verificação de segurança da inicialização do hardware
    if (!device_is_ready(display)) {
        printk("Erro Crítico: Display OLED não detectado.\n");
        return 0;
    }

    // Ativação do Display
    display_blanking_off(display);

    // Sequência de animação de inicialização (Teste de sanidade)
    clear_screen();
    draw_bar(128); // Barra cheia
    k_sleep(K_MSEC(500));
    clear_screen(); // Apaga tela

    printk("Sistema HMI Inicializado. Aguardando interrupções...\n");

    // Loop principal permanece ocioso (sleep) para economizar energia,
    // pois toda a lógica é tratada via interrupções.
```

```
    while (1) {
        k_sleep(K_FOREVER);
    }
    return 0;
}
```