CÓDIGOS UTILIZADOS

RAVI DE FARIAS

Arquivo de Configuração (`prj.conf`)

```
# --- ARQUIVO: prj.conf ---

# 1. Habilita os barramentos de hardware
CONFIG_GPIO=y
CONFIG_I2C=y

# 2. Habilita o subsistema de Display
CONFIG_DISPLAY=y
CONFIG_DISPLAY_SSD1306=y

# 3. Habilita a biblioteca gráfica de texto (CFB)
# Isso permite usar fontes prontas e comandos como cfb_print()
CONFIG_CHARACTER_FRAME_BUFFER=y
CONFIG_CHARACTER_FRAME_BUFFER_SHELL=y

# 4. Habilita o subsistema de Entrada (Input) para o Teclado
CONFIG_INPUT=y

# 5. Logs para depuração (opcional, ajuda a achar erros)
CONFIG_LOG=y
CONFIG_DISPLAY_LOG_LEVEL_ERR=y
```

rquivo de Mapeamento de Hardware (`app.overlay`)

```
/* --- ARQUIVO: app.overlay --- */

/ {
    aliases {
```

```
        kpad = &keypad;
    };

    /* Configuração do Teclado Matricial 4x4 */
    keypad: keypad {
        compatible = "gpio-kbd-matrix";
        label = "Teclado 4x4";

        /* CONFIGURAÇÃO DAS LINHAS (ROWS) - L1 a L4 */
        /* Pinos configurados como Entrada com Resistor de Pull-Up interno */
        /* Ajuste os números <15, 16...> conforme sua ligação física na SAM R21 */
        row-gpios = <&gpio0 15 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>,
                    <&gpio0 16 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>,
                    <&gpio0 17 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>,
                    <&gpio0 18 (GPIO_PULL_UP | GPIO_ACTIVE_LOW)>;

        /* CONFIGURAÇÃO DAS COLUNAS (COLS) - C1 a C4 */
        /* Pinos configurados como Saída para varredura */
        col-gpios = <&gpio0 19 GPIO_ACTIVE_LOW>,
                    <&gpio0 20 GPIO_ACTIVE_LOW>,
                    <&gpio0 21 GPIO_ACTIVE_LOW>,
                    <&gpio0 22 GPIO_ACTIVE_LOW>;

        /* Tempo de debouncing (filtro de ruído do botão) */
        debounce-down-ms = <10>;
        debounce-up-ms = <10>;
    };
};
```

```
/* Configuração do Display OLED no barramento I2C (EXT1) */
&sercom1 {
    status = "okay";
    compatible = "atmel,sam0-i2c";
    clock-frequency = <I2C_BITRATE_FAST>;

    ssd1306: ssd1306@3c {
        compatible = "solomon,ssd1306fb";
        reg = <0x3c>;
        width = <128>;
        height = <32>; /* Resolução típica do OLED1 Xplained */
        segment-offset = <0>;
        page-offset = <0>;
        display-offset = <0>;
        multiplex-ratio = <31>;
        prechargep = <0x22>;
    };
};
```

Código da Aplicação (`main.c`)

```
/* --- ARQUIVO: main.c --- */
#include <zephyr/kernel.h>
#include <zephyr/device.h>
#include <zephyr/drivers/display.h>
#include <zephyr/display/cfb.h> // Biblioteca de Fontes
#include <zephyr/input/input.h> // Biblioteca de Entrada
#include <stdio.h>

// 1. Obtém a referência do display definida no .overlay
```

```
const struct device *display = DEVICE_DT_GET(DT_NODELABEL(ssd1306));

// Variáveis globais para armazenar o texto
char buffer_texto[32] = "";
int pos = 0;

// 2. Função de Callback: Executa SEMPRE que uma tecla é pressionada
static void keypad_callback(struct input_event *evt, void *user_data)
{
    // Verifica se o evento é "Pressionar" (val = 1). Ignora "Soltar" (val = 0).
    if (evt->value == 1) {

        // Lógica simples de mapeamento (ajuste conforme seu teclado real)
        // O driver envia códigos sequenciais 0-15 baseados na matriz
        char tecla = '?';
        if (evt->code >= 0 && evt->code <= 9) {
            tecla = '0' + evt->code; // Converte 0-9 para char '0'-'9'
        } else {
            tecla = 'A' + (evt->code - 10); // Simula letras para códigos acima de 9
        }

        // Adiciona o caractere ao buffer de texto
        if (pos < 16) { // Limite de caracteres da tela
            buffer_texto[pos++] = tecla;
            buffer_texto[pos] = '\0';
        } else {
            // Se encher, reinicia
            pos = 0;
            buffer_texto[0] = '\0';
```

```
        }

        // --- ATUALIZAÇÃO DO DISPLAY ---
        cfb_framebuffer_clear(display, false); // Limpa a memória de vídeo

        cfb_print(display, "Digitado:", 0, 0); // Linha superior
        cfb_print(display, buffer_texto, 0, 16); // Linha inferior (offset Y=16)

        cfb_framebuffer_finalize(display); // Envia os dados para o OLED acender
    }
}

// 3. Registra o callback para o dispositivo "keypad"
INPUT_CALLBACK_DEFINE(DEVICE_DT_GET(DT_NODELABEL(keypad)), keypad_callback, NULL);

int main(void)
{
    // Verifica se o display foi encontrado
    if (!device_is_ready(display)) {
        printk("Erro: Display não iniciado.\n");
        return 0;
    }

    // --- INICIALIZAÇÃO DO DISPLAY ---
    display_set_pixel_format(display, PIXEL_FORMAT_MONO10);
    cfb_framebuffer_init(display);
    cfb_framebuffer_clear(display, true);
    display_blanking_off(display); // Liga o display
```

```
    // Configura a fonte (índice 0 é a padrão do sistema)
    cfb_framebuffer_set_font(display, 0);

    // Escreve mensagem de boas-vindas
    cfb_print(display, "Sistema Ativo", 0, 0);
    cfb_framebuffer_finalize(display);

    printk("Aguardando entrada do teclado...\n");

    // O loop principal fica vazio pois tudo é feito via Interrupção (Callback)
    while (1) {
        k_sleep(K_FOREVER);
    }
    return 0;
}
```