Detektor přítomnosti

ZÁVĚREČNÝ PROJEKT DO MIT

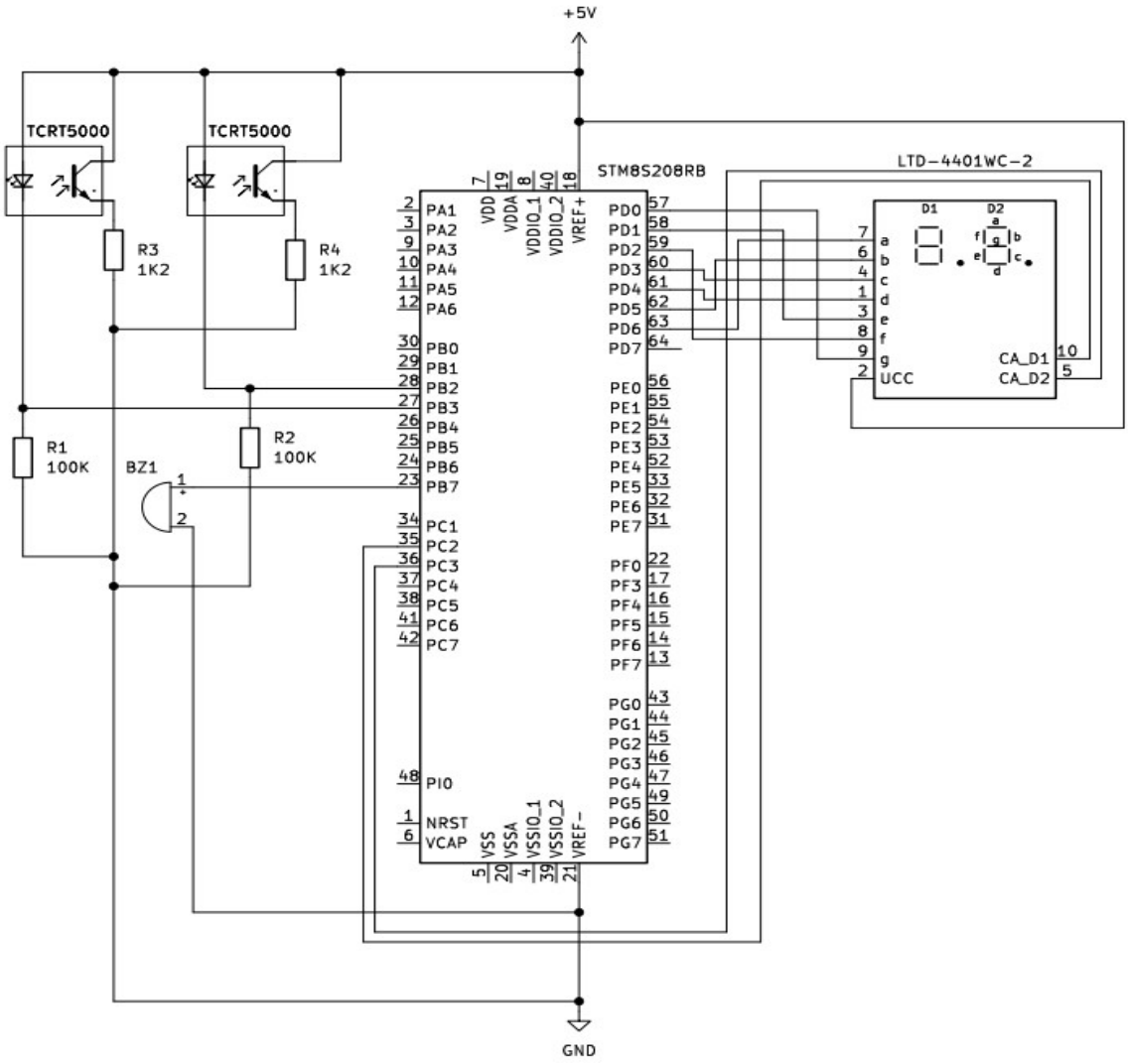
Zadání:

Zařízení reagující na počet osob v místnosti.

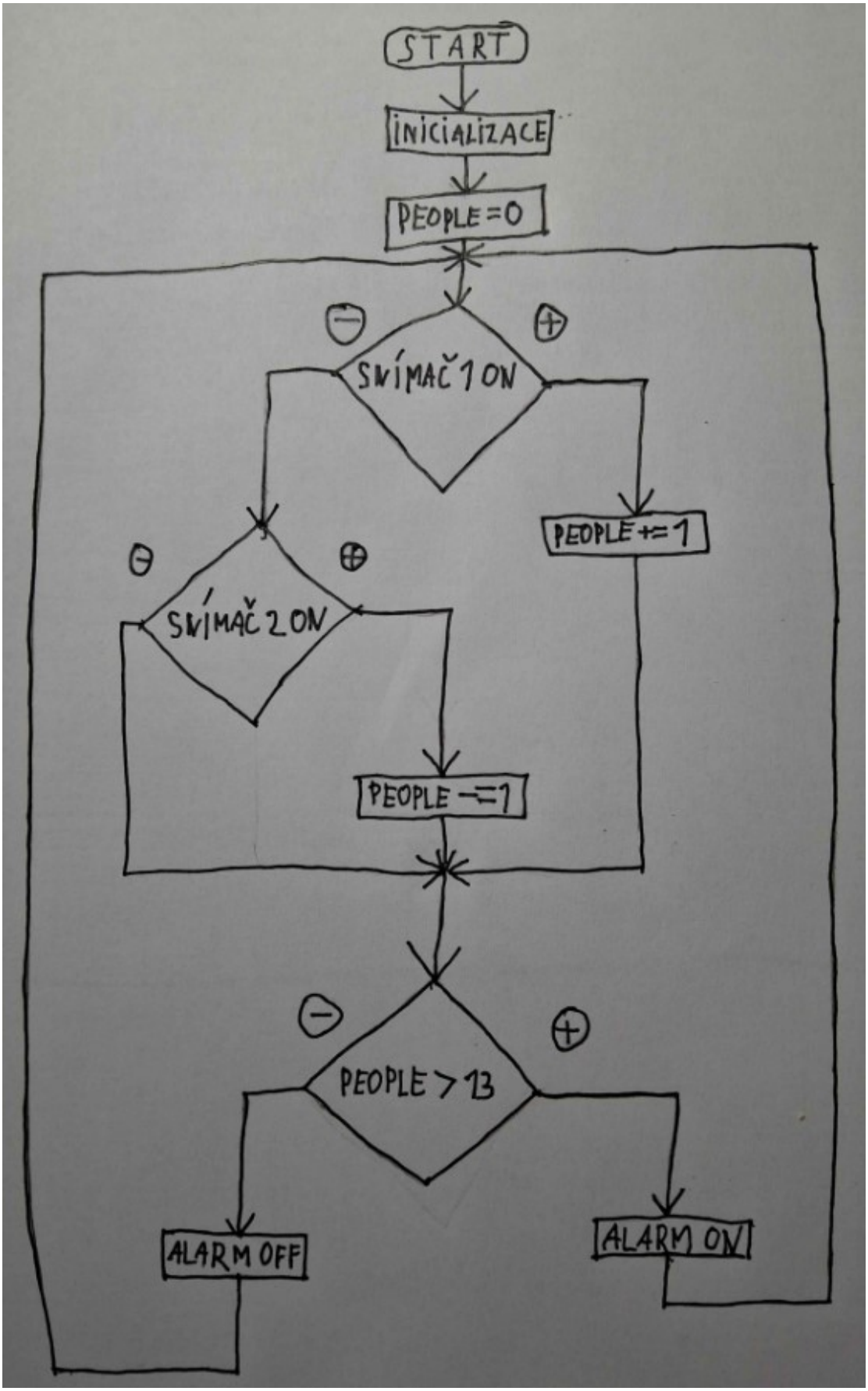
- Zařízení detekuje pohyb osob v místnosti
- Místnost bude mít svůj vstup a výstup
- Když někdo vstoupí, přičte se 1, když někdo odejde, 1 se odečte
- Když v místnosti bude 14 a více lidí, začne houkat alarm
- Alarm se vypne až ve chvíli kdy je v místnosti 13 lidí
- Celý systém se popřípadě restartuje pomocí tlačítka

Použité součástky: Detekce překážky pomocí IR 2x, LED display, piezo reproduktor

Schéma zapojení:



Vývojový diagram:



Popis činnosti programu:

Princip programu má sloužit jako detektor pohybu v místnosti. Místnost má svůj vstup a výstup. Místnost má však svůj omezený počet osob, který v ní může být. Když je v místnosti víc osob než má, spustí se alarm a je zapnutý až do doby dokud není v místnosti povolený počet lidí.

Po zapojení k napájení je vždy na displeji zobrazena 0. Když senzor 1 detekuje pohyb, přičte na displej 1. Displej může počítat až do 99. Když senzor 2 detekuje pohyb, odečte se 1. Na displeji se nemůže zobrazit záporné číslo, čili nejmenší zobrazitelné číslo je 0. Pokud je na displeji číslo vyšší jak 13, spustí se piezo reproduktor. Ten je aktivní až do doby dokud není na displeji číslo menší jak 14, nebo je odpojen od napájení. Celé zařízení je aktivní až do doby, dokud není vypnuto napájení.

Zdrojový kód:

```
#include <main.h>
#include <stm8s.h>
#include "adc_helper.h"

void rx_action(void) // Kompilace programu
{
    char c = UART1_ReceiveData8();
}


#define DISPLAY_DATA_PORT GPIOD

#define DISPLAY_A_PIN GPIO_PIN_3
#define DISPLAY_A_PORT GPIOC
#define DISPLAY_B_PIN GPIO_PIN_2
#define DISPLAY_B_PORT GPIOC

#define BUZZER_PORT GPIOB
#define BUZZER_PIN GPIO_PIN_7

uint8_t nums[10] = {0b10000000, 0b11110010, 0b01001000, 0b01100000,
0b00110010, 0b00100100, 0b00000100, 0b11110000, 0b00000000, 0b00100000}; //
Číslo definované hodnotou portu

INTERRUPT_HANDLER(EXTI_PORTD_IRQHandler, 6)
{
}

void setup(void)
{
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1); // Nstavíme clock

    /*Inicializace displejů*/
```

```
    GPIO_Init(DISPLAY_DATA_PORT, GPIO_PIN_1, GPIO_MODE_OUT_PP_LOW_SLOW); //
PP- Push/Pull - může být nastaven na 0 i 1, LOW - počátek výstupu je 0, SLOW -
nízká rychlost pinu
    GPIO_Init(DISPLAY_DATA_PORT, GPIO_PIN_2, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(DISPLAY_DATA_PORT, GPIO_PIN_3, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(DISPLAY_DATA_PORT, GPIO_PIN_4, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(DISPLAY_DATA_PORT, GPIO_PIN_5, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(DISPLAY_DATA_PORT, GPIO_PIN_6, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(DISPLAY_DATA_PORT, GPIO_PIN_7, GPIO_MODE_OUT_PP_LOW_SLOW);

    GPIO_Init(DISPLAY_A_PORT, DISPLAY_A_PIN, GPIO_MODE_OUT_PP_LOW_SLOW); //
PP- Push/Pull - může být nastaven na 0 i 1, LOW - počátek výstupu je 0, SLOW -
nízká rychlost pinu
    GPIO_Init(DISPLAY_B_PORT, DISPLAY_B_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);

    GPIO_Init(BUZZER_PORT, BUZZER_PIN, GPIO_MODE_OUT_PP_LOW_SLOW); // Nastaví
pin pieza na nízkou úroveň
    GPIO_WriteLow(BUZZER_PORT, BUZZER_PIN);

    init_time();
    init_uart1();

    ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL14, DISABLE);
    ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL15, DISABLE);
    // nastavíme clock pro ADC2a (16MHZ / 4  = 4MHz)
    ADC2_PrescalerConfig(ADC2_PRESSEL_FCPU_D4);
    // volíme zarovnání výsledku -- typicky do prava
    ADC2_AlignConfig(ADC2_ALIGN_RIGHT);
    // nastavíme multiplexer na **některý** kanál
    ADC2_Select_Channel(ADC2_CHANNEL_14);
    // rozběhnemen ADC
    ADC2_Cmd(ENABLE);
    // počkáme až se rozběhne ADC  (~7us)
    ADC2_Startup_Wait();
}
uint8_t people = 0;
int main(void)
{
    setup();
    uint32_t timestamp = 0;
    int i = 0;
    uint8_t old_a_stav = 0;
    uint8_t old_b_stav = 0;
    while (1)
    {
        if (milis() - timestamp >= 300)
        {
            timestamp = milis();
```

```
            if (i >= 9) {   // 9 - max. číselná hodnota zobrazitelná na 7-
segment
                i = 0;
            } else {
                i = i + 1;
            }

            printf("%d, %d\n", ADC_get(ADC2_CHANNEL_2),
ADC_get(ADC2_CHANNEL_3)); // hodnoty na displeji
        }
        if(ADC_get(ADC2_CHANNEL_2) > 100){  //Pokud hodnota z ADC kanálu 2
překročí 100 a předchozí stav (old_a_stav) byl 0, zvýší počet lidí
            if(old_a_stav == 0){
                people++;
            }
            old_a_stav = 1;
        }else{
            old_a_stav = 0;
        }
        if(ADC_get(ADC2_CHANNEL_3) > 100){  //Pokud hodnota z ADC kanálu 3
překročí 100 a old_b_stav je 0 (což znamená, že předchozí stav byl neaktivní),
dojde k dalšímu testu
            if(old_b_stav == 0){
                if(people >=1 ) people--; else people = 0;   //Pokud proměnná
people je větší nebo rovna 1, sníží se její hodnota o 1, Pokud proměnná people
je menší než 1, nastaví se na 0
            }
            old_b_stav = 1;
        }else{
            old_b_stav = 0;
        }
        if(people > 13) GPIO_WriteHigh(BUZZER_PORT, BUZZER_PIN); else
GPIO_WriteLow(BUZZER_PORT, BUZZER_PIN); //Alarm na 14+ lidí, čísle můžeme
změnit počet


        uint8_t desitky = people / 10; //Rozdělení desítek a jednotek na
displeji
        uint8_t jednotky = people - (desitky * 10);

        GPIO_WriteHigh(DISPLAY_B_PORT, DISPLAY_B_PIN); // vypne společnou
katodu jednotek, což umožňuje zobrazování hodnoty pro desítky na displeji.
        GPIO_WriteLow(DISPLAY_A_PORT, DISPLAY_A_PIN);  // zapne společnou
katodu desítek, čímž umožňuje zobrazování hodnoty pro desítky na displeji.

        DISPLAY_DATA_PORT->ODR = nums[jednotky]; // Vypíše jednotky
        delay_ms(10);   // delay pro rozpoznání
```

```
        GPIO_WriteHigh(DISPLAY_A_PORT, DISPLAY_A_PIN); // vypne společnou
katodu desítek, což umožňuje zobrazování hodnoty pro jednotky na displeji.
        GPIO_WriteLow(DISPLAY_B_PORT, DISPLAY_B_PIN);  // zapne společnou
katodu jednotek, což umožňuje zobrazování hodnoty pro jednotky na displeji.
        DISPLAY_DATA_PORT->ODR = nums[desitky]; // Vypíše desítky
        delay_ms(10);   // delay pro rozpoznání
    }
    return 0;
}
```