Vyšší odborná škola a Střední průmyslová škola elektrotechnická
Božetěchova 3, Olomouc
Laboratoře elektrotechnických měření

PROTOKOL MIT

Název úlohy

Joystick s bzučákem

Zadání

1. Vytvořit program, který bude vypisovat polohu joysticku v putty
2. Při stisku joysticku začne bzučák bzučet na určené frekvenci

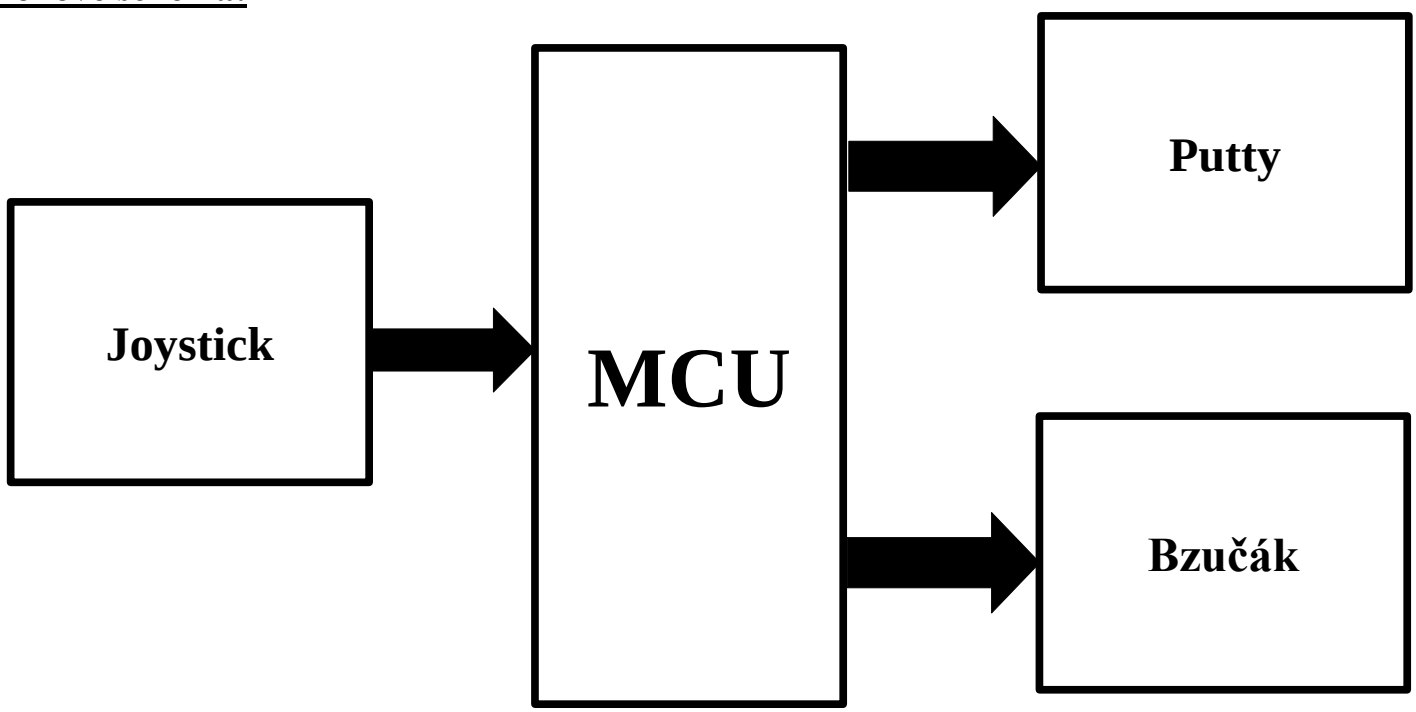
Poř. č.	Příjmení a jméno	Třída	Skupina	Školní rok
7	Kocian David	4.B	1	2021/22

Datum odevzdání	Počet listů	Klasifikace			
		příprava	měření	protokol	obhajoba
26.4.2021	5				

Protokol o měření obsahuje:

- blokové schéma
- slovní popis funkce
- schéma
- výpis programu
- vývojoví diagram
- závěr

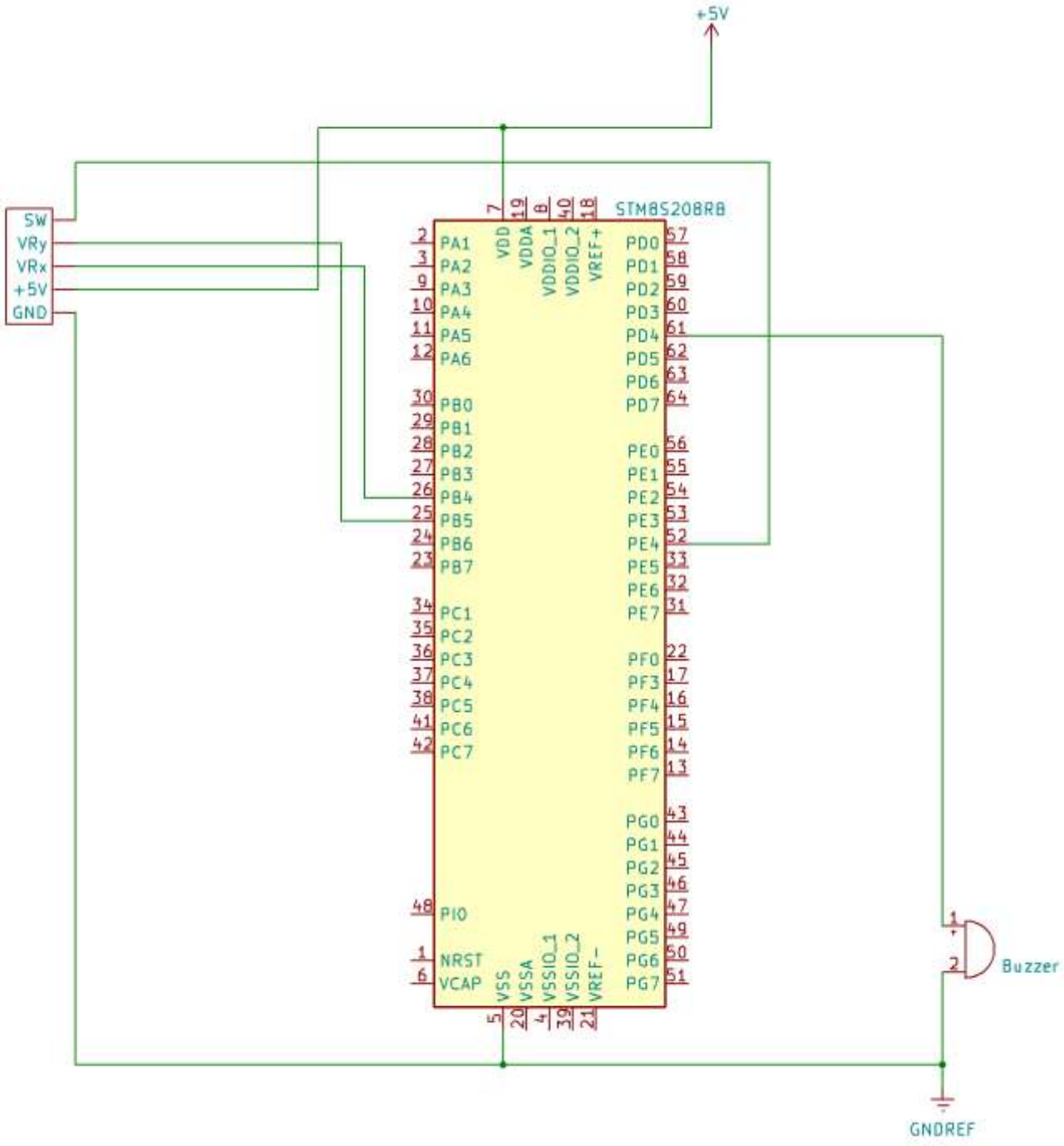
Blokové schéma:



Slovní popis funkce:

Jako výstup se do putty vypisuje aktuální poloha joystiku na osách X a Y. Po stisknutí joystiku se zapne pasivní bzučák na určené frekvenci.

Schéma:



Program:

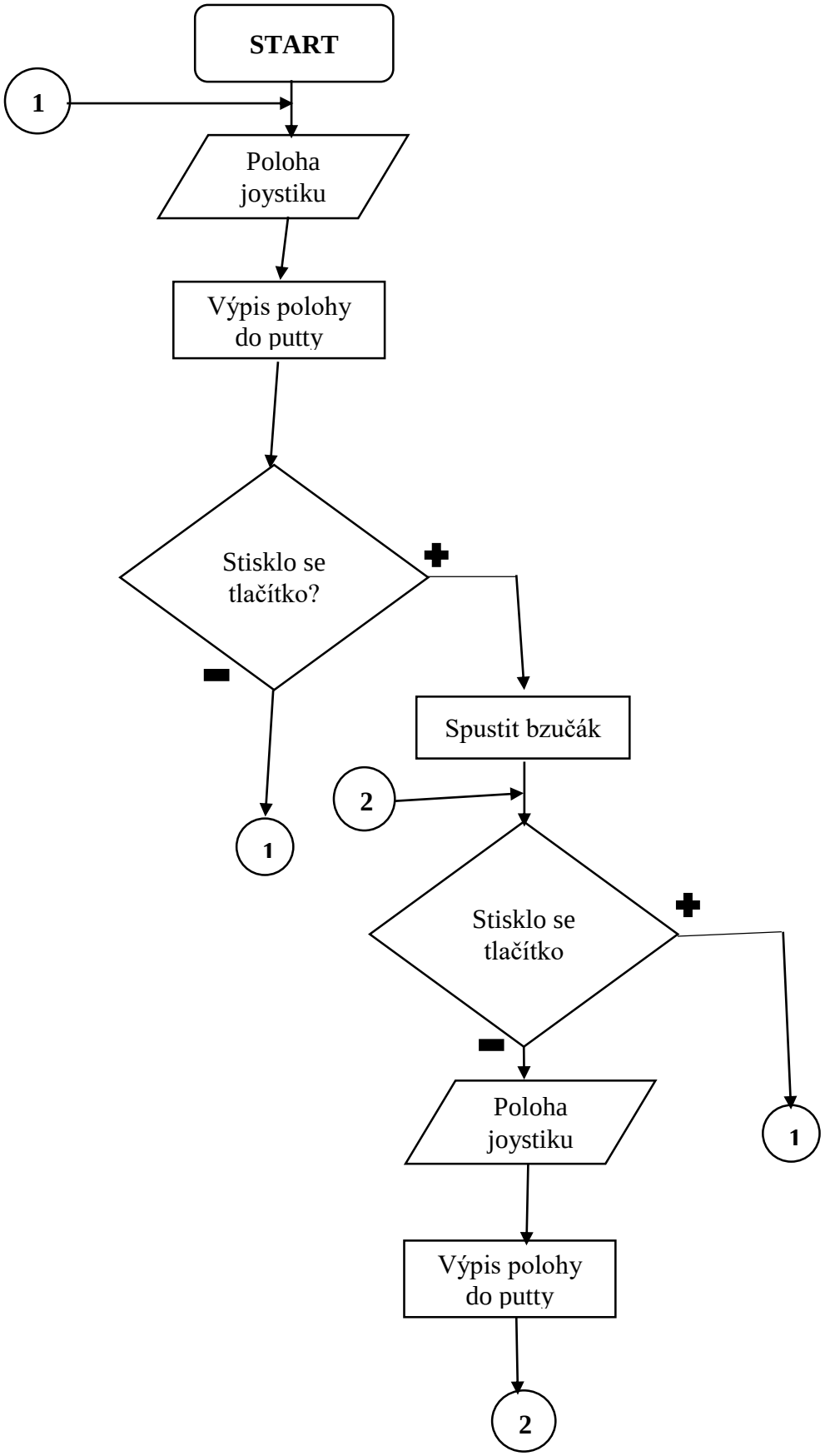
main.c:

```
#include "stm8s.h"
#include "milis.h"
#include "spse_stm8.h"

/*#include "delay.h"*/
// #include <cstdint>
#include <stdio.h>
#include "stm8s_adc2.h"
#include "uart1.h"
```

```
#define _ISOC99_SOURCE
#define _GNU_SOURCE

#define LED_PORT GPIOC
#define LED_PIN  GPIO_PIN_5
#define LED_HIGH   GPIO_WriteHigh(LED_PORT, LED_PIN)
#define LED_LOW  GPIO_WriteLow(LED_PORT, LED_PIN)
#define LED_REVERSE GPIO_WriteReverse(LED_PORT, LED_PIN)

#define BTN_PORT GPIOE
#define BTN_PIN  GPIO_PIN_4
#define BTN_PUSH (GPIO_ReadInputPin(BTN_PORT, BTN_PIN)==RESET)


void setup(void)
{
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);      // taktovani MCU na
16MHz
    GPIO_Init(LED_PORT, LED_PIN, GPIO_MODE_OUT_PP_LOW_SLOW);
    GPIO_Init(BTN_PORT, BTN_PIN, GPIO_MODE_IN_FL_NO_IT);

    init_milis();
    init_uart1();

    TIM2_TimeBaseInit(TIM2_PRESCALER_4, 7000 - 1 );

TIM2_OC1Init(TIM2_OCMODE_PWM1,TIM2_OUTPUTSTATE_ENABLE,3000,TIM2_OCPOLARITY_HI
GH);// inicializujeme kanál 1 (TM2_CH1)
    TIM2_OC1PreloadConfig(ENABLE);
    TIM2_Cmd(ENABLE);

 // inicializace ADC //
    // na pinech/vstupech ADC_IN2 (PB4) a ADC_IN3 (PB5) vypneme vstupní
buffer
    ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL4, DISABLE);
    ADC2_SchmittTriggerConfig(ADC2_SCHMITTTRIG_CHANNEL5, DISABLE);

    // při inicializaci volíme frekvenci AD převodníku mezi 1-4MHz při 3.3V
    // mezi 1-6MHz při 5V napájení
    // nastavíme clock pro ADC (16MHz / 4 = 4MHz)
    ADC2_PrescalerConfig(ADC2_PRESSEL_FCPU_D4);

    // volíme zarovnání výsledku (typicky vpravo, jen vyjmečně je výhodné
vlevo)
    ADC2_AlignConfig(ADC2_ALIGN_RIGHT);

    // nasatvíme multiplexer na některý ze vstupních kanálů
    ADC2_Select_Channel(ADC2_CHANNEL_4);
    // rozběhneme AD převodník
    ADC2_Cmd(ENABLE);
    // počkáme než se AD převodník rozběhne (~7us)
    ADC2_Startup_Wait();
}

int main(void)
{
    uint32_t time = 0;
    uint8_t aktual_stav = 0;
```

```
    uint8_t minuly_stav = 0;
    uint8_t stav_bzucaku = 0;
    uint16_t ADCx;
    uint16_t ADCy;

    setup();

    while (1) {

        if (milis() - time > 50) {
            time = milis();

            ADCx = ADC_get(ADC2_CHANNEL_4);
            ADCy = ADC_get(ADC2_CHANNEL_5);

            printf("osa x = %d osa y = %d\n\r", ADCx, ADCy);
        }

        if (BTN_PUSH){
            aktual_stav = 1;
        }
        else{
            aktual_stav = 0;
        }

        if (aktual_stav == 1 && minuly_stav == 0){
            LED_REVERSE;
            if (stav_bzucaku == 0){
                TIM2_SetCompare1(4000);
                stav_bzucaku = 1;
            }
            else{
                TIM2_SetCompare1(0);
                stav_bzucaku = 0;
            }
        }

        minuly_stav = aktual_stav;



    }
}


/*------------------------------  Assert -----------------------------------
*/
#include "__assert__.h"
```

Vývojoví diagram:



Závěr:
Projekt se mi podařilo úspěšně zprovoznit a naučil jsem se pracovat s výstupem přes UART a zároveň jak funguje passivní bzučák