Minutka

Jméno studenta: Michal Kolařík

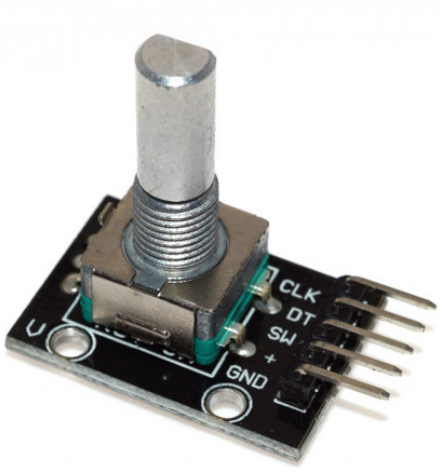
Zadání: Pomocí STM8 vytvořte minutku ovládanou enkodérem, zobrazení na displeji

Součástky:

tlačítko, bzučák, MAX7219 displej, vodiče, enkoder, nepájivé pole

Teoretický úvod:

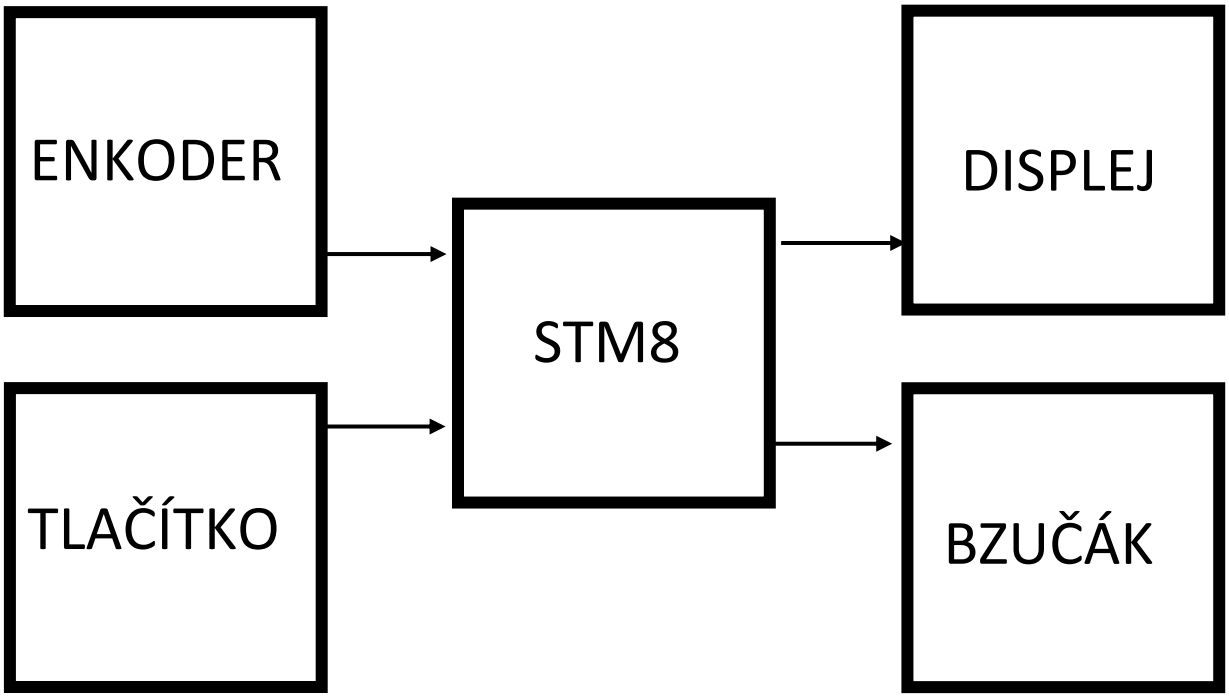
Cílem projektu je pomocí STM8 vytvořit minutku ovládanou enkodérem a zobrazením odpočtu na displeji. Použit je rotační enkodér keys KY-040 s tlačítkem. Rotační enkodér KY-040 je rotační vstupní součástka, která při otáčení její osou poskytuje informaci o této rotaci a také jejím směru. Zároveň tento typ rotačního enkodéru obsahuje i tlačítko. Dále je použit displej MAX7219, což je univerzální displej, který zahrnuje 8 sedmi segmentových prvků. Displej je vhodný především pro vypisování číslic (disponuje i desetinnou tečkou) nebo pro zobrazování jednoduššího textu. který má 8 znaků sedmisegmentového typu. Tento displej je červeně podsvícený pomocí LED. Zařízení komunikuje přes rozhraní SPI. Displej je osazen čipem MAX7219, který je využíván také v LED maticích. V mém konkrétním případě je napájen 5V ale lze ho napájet i 3,3V za cenu nižší svítivosti. Dále je použit aktivní bzučák na 5V což znamená, že jeho zapojení je velmi jednoduché – stačí přivést na 5V.



Rotační enkoder KY-040

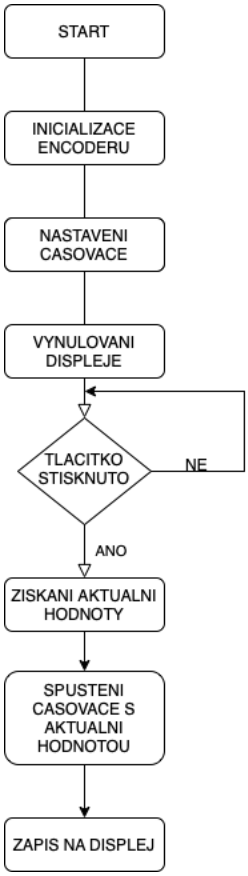
Displej MAX7219

Schéma zapojení:



Slovní popis zapojení:
V zapojení se nachází MCU STM8, displej MAX 7219 zapojený následovně: CLK na PD4, CS na PD2, DIN na PD3, napájen je přímo z STM8. Enkodér je zapojen: CLK na PC2, DT na PC1, SWITCH na PE4, napájeno z STM8. Zem bzučáku je zapojena na PG2.

Vývojový diagram:



Ukázka kódu:

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

#include "delay.h"
#include "max7219.h"


void delay_ms(uint16_t ms) {            //fce umožnující delay v ms
    uint16_t  i;
    for (i=0; i<ms; i = i+1){
        _delay_us(250);
        _delay_us(248);
        _delay_us(250);
        _delay_us(250);
    }
}


void ncoder_init(void){
    TIM1_DeInit();
    TIM1_TimeBaseInit(8, TIM1_COUNTERMODE_UP, 60, 8); //inicializace enkoderu
```

```
    TIM1_EncoderInterfaceConfig(TIM1_ENCODERMODE_TI12,
                                TIM1_ICPOLARITY_FALLING,
                                TIM1_ICPOLARITY_FALLING);
    TIM1_Cmd(ENABLE);

}

void timer(uint16_t time){
    uint8_t temp = 1;
    uint16_t temp2 = 1000;
    uint32_t time2 = milis();
    while(temp){
        if(milis() > time * 1000 + time2){
            temp = 0;
            max7219_posli(DIGIT0,0);
            max7219_posli(DIGIT1,0);
            GPIO_WriteReverse(GPIOG,GPIO_PIN_2);
            delay_ms(2000);
            GPIO_WriteReverse(GPIOG,GPIO_PIN_2);
        }
        else{
            max7219_posli(DIGIT0,((time * 1000) - (milis() - time2)) / 1000 %10);
            max7219_posli(DIGIT1,((time * 1000) - (milis() - time2)) / 1000 /10);
        }
    }
}


void setup(void)
{
    CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);      // taktovani MCU na 16MHz

    max7219_init(); //inicializace max displeje

    ncoder_init(); //inicializace enkoderu

    init_milis(); //inicializace milis

    GPIO_Init(GPIOE, GPIO_PIN_4,GPIO_MODE_IN_FL_NO_IT); // nastavime PE4 jako vstup
(button)

    GPIO_Init(GPIOC, GPIO_PIN_1,GPIO_MODE_IN_PU_NO_IT); // nastavime PC1 pro
enkoder (clk)
    GPIO_Init(GPIOC, GPIO_PIN_2,GPIO_MODE_IN_PU_NO_IT); // nastavime PC2 pro
enkoder

    GPIO_Init(GPIOG, GPIO_PIN_2,GPIO_MODE_OUT_PP_LOW_SLOW); // nastavime PG2 jako
výstup pro buzzer
    GPIO_WriteHigh(GPIOG,GPIO_PIN_2); // zapsani high na buzzer aby nebzucel po
spusteni programu
```