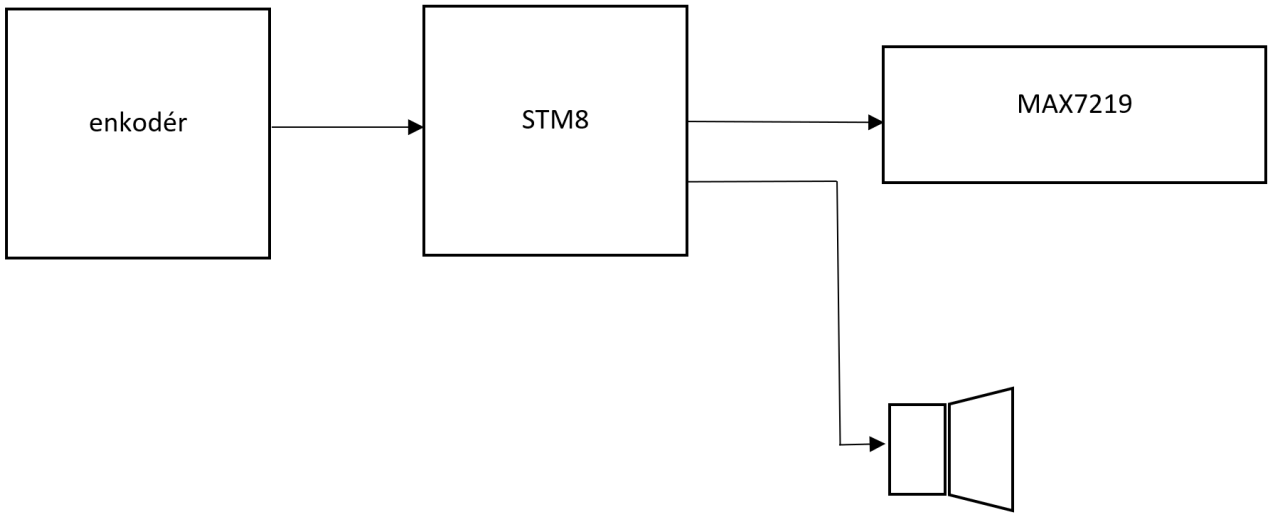
1. Blokové schéma



Mikrokontrolér

- Typ STM8S208RB

Displej

- Sedmisegmentový
- typ ovladače MAX7219
- slouží k zobrazování času

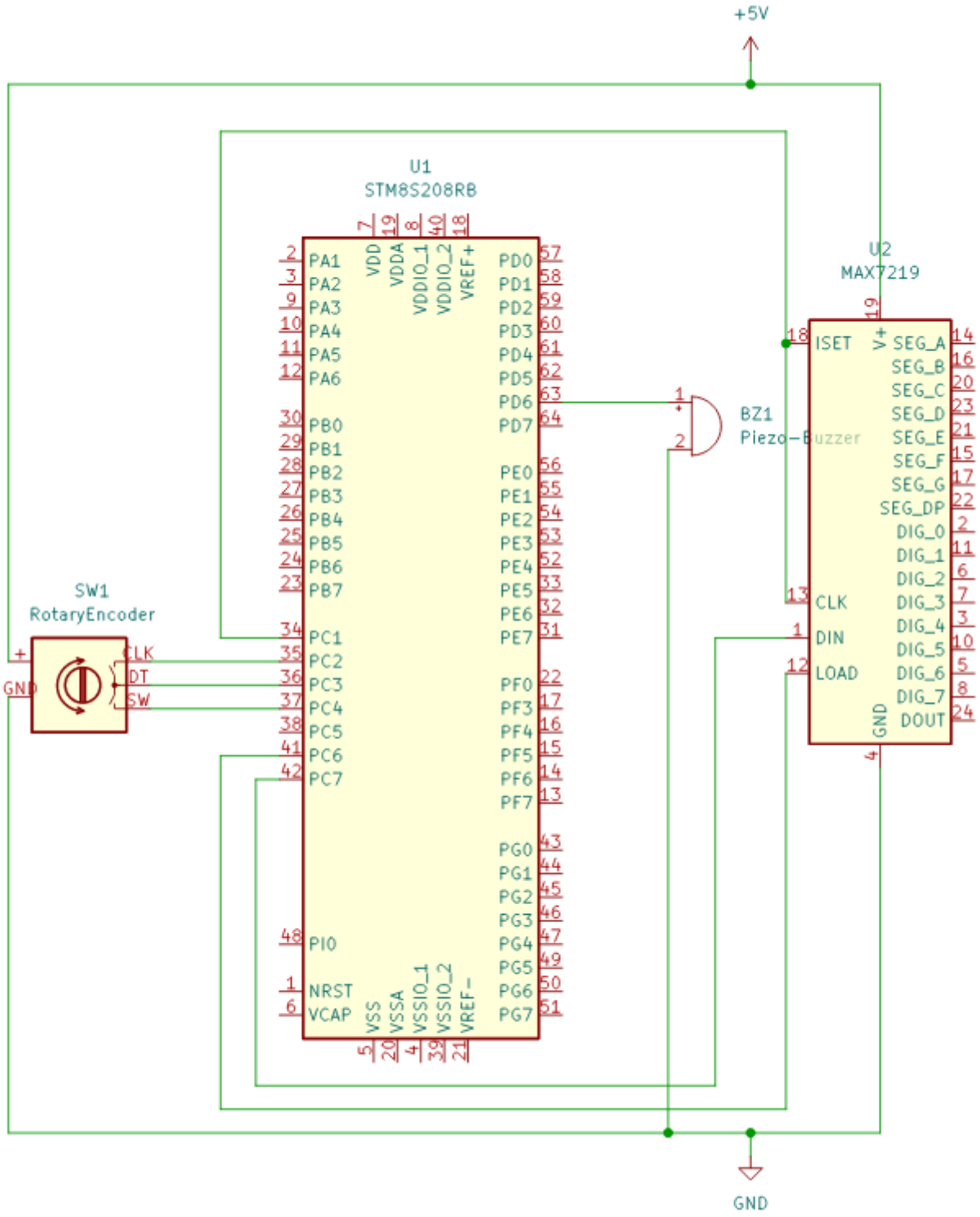
Rotační enkodér

- slouží k ovládání minutek

Siréna

- realizovaná samovybuzovacím piezo-bzučákem
- napájeno 5 V z mikrokontroléru

2. Schéma zapojení



3. Program

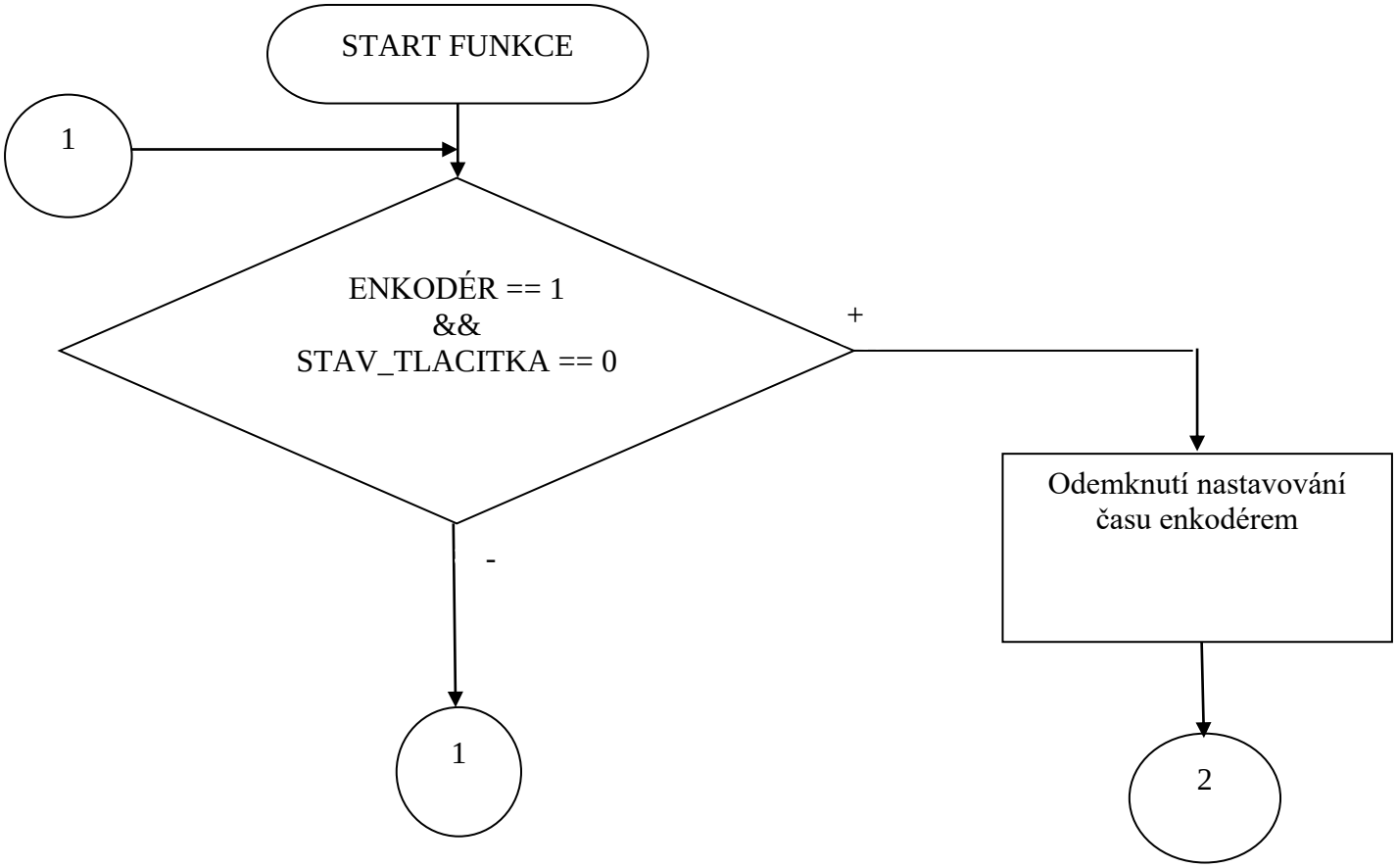
```
//knihovny
#include "stm8s.h"
#include "milis.h"
#include "stm8_hd44780.h"
#include "stdio.h"
#include "swspi.h"

#define DECODEMODE		(0x9<<8)
#define INTENSITY		(0xa<<8)
#define SCANLIMIT		(0xb<<8)
#define SHUTDOWN			(0xc<<8)
#define DTEST				(0xf<<8)

#define d3		(0x3<<8)
#define d4		(0x4<<8)
#define d5		(0x5<<8)
#define d6		(0x6<<8)

void init_enc(void);
void process_enc(void);
void init_timer(void);
void tlacitko(void);
void max7219_init(void);
void stopky_cas(void);
void stopky(void);

uint16_t sekundy=0;
uint8_t desitkyminut=0;
uint8_t minuty=0;
uint8_t desitky=0;
uint16_t minuly_cas=0;
uint16_t cas=0;
uint8_t stav_tlacitka=0;

volatile int16_t hodnota=0; // proměnná jejíž hodnotu měníme enkodérem

void main(void){
CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1); // 16MHz z interního RC
oscilátoru
init_milis(); // milis kvuli delay_ms()
init_enc();		// inicializace vstupu enkodéru
lcd_init();		// inicializace displeje
init_timer();	// spustí tim3 s poerušením každé 2ms
enableInterrupts(); // není nutné, protože tuto funkci voláme v init_milis()

CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1);
swspi_init();
```

```
max7219_init();
while (1){
                swspi_tx16(0x010A);
                swspi_tx16(0x020A);
                swspi_tx16(0x070A);
                swspi_tx16(0x080A);

                swspi_tx16(d3 | sekundy);
                swspi_tx16(d4 | desitky);
                swspi_tx16(d5 | minuty);
                swspi_tx16(d6 | desitkyminut);
                stopky_cas();
                tlacitko();
                stopky();
}
}

// rutina poerušení od update události timeru 3 (přemístina z stm8s_it.c !)
 INTERRUPT_HANDLER(TIM3_UPD_OVF_BRK_IRQHandler, 15)
 {
        TIM3_ClearITPendingBit(TIM3_IT_UPDATE); // vyčistit vlajku (nutné vždy)
        process_enc(); // zkontrolovat stav pinu enkodéru
 }

 void stopky_cas(void){  //rozděluje čas na číslice pro jednotlivé segmenty
        if (hodnota<0){
                hodnota=0;
}
        sekundy=hodnota;
        desitkyminut=sekundy/600;
        sekundy=sekundy%600;
        minuty=sekundy/60;
        sekundy=sekundy%60;
        desitky=sekundy/10;
        sekundy=sekundy%10;
}

void stopky(void) {     //každou sekundu odečítá hodnotu celkovy_cas
        static uint16_t minuly_cas=0;
        uint16_t cas;

        cas = milis();
        if(stav_tlacitka==1||stav_tlacitka==2){ //spuštění stopek tlačítkem
                if((cas-minuly_cas) >= 1000){
                        hodnota=hodnota-1;
                        minuly_cas = cas;
                }
        }
        if(stav_tlacitka==3){ //vyresetování stopek
                hodnota=0;
```

```
	}
}

void tlacitko(void){
	if(GPIO_ReadInputPin(GPIOC,GPIO_PIN_4)==RESET && stav_tlacitka==0){
		stav_tlacitka=1;
		hodnota++; //jednoché řešení problému s milis, kdy minutky odečtou první
sekundu hned po spuštění
	}
	if(GPIO_ReadInputPin(GPIOC,GPIO_PIN_4)!=RESET && stav_tlacitka==1){
		stav_tlacitka=2;
	}
	if(GPIO_ReadInputPin(GPIOC,GPIO_PIN_4)==RESET && stav_tlacitka==2){
		stav_tlacitka=3;
	}
	if(GPIO_ReadInputPin(GPIOC,GPIO_PIN_4)!=RESET && stav_tlacitka==3){
		stav_tlacitka=0;
	}
}


void max7219_init(void){
	swspi_tx16(DECODEMODE | 0xff);
	swspi_tx16(INTENSITY | 3);
	swspi_tx16(SCANLIMIT | 7);
	swspi_tx16(DTEST | 0);
	swspi_tx16(SHUTDOWN | 1);

}


void init_timer(void){
TIM3_TimeBaseInit(TIM3_PRESCALER_16,1999);
TIM3_ITConfig(TIM3_IT_UPDATE, ENABLE); // povolit přerušení
TIM3_Cmd(ENABLE); // spustit timer
}

void init_enc(void){
	// enkodéry jsou jen spínače, takže vstupy volíme ve stejném režimu jako pro tlaeítka
GPIO_Init(GPIOC,GPIO_PIN_3,GPIO_MODE_IN_PU_NO_IT);  // vstup, s vnitoním pullup
rezistorem
GPIO_Init(GPIOC,GPIO_PIN_2,GPIO_MODE_IN_PU_NO_IT);
GPIO_Init(GPIOC,GPIO_PIN_4,GPIO_MODE_IN_PU_NO_IT);
}

// vyhodnocuje stav enkodéru
void process_enc(void){
	static minule=1; // pamatuje si minulý stav vstupu A (nutné k detekování sestupné
hrany)
```

```
	// pokud je na vstupu A hodnota 0 a minule byla hodnota 1 tak jsme zachytili sestupnou hranu
	if(GPIO_ReadInputPin(GPIOC,GPIO_PIN_3) == RESET && minule==1){
		minule = 0; // nyní je pin v log.0
		// poeeteme stav vstupu B
		if(GPIO_ReadInputPin(GPIOC,GPIO_PIN_2) == RESET){
			// log.0 na vstupu B (krok jedním směrem)
			if(stav_tlacitka==0){hodnota=hodnota+30;} //nelze měnti čas při spuštěných stopkách
		}else{
			// log.1 na vstupu B (krok druhým směrem)
			if(stav_tlacitka==0){hodnota=hodnota-30;} //nelze měnti čas při spuštěných stopkách
		}
	}
	if(GPIO_ReadInputPin(GPIOC,GPIO_PIN_3) != RESET){minule = 1;} // pokud je vstup A v log.1
}


#ifdef USE_FULL_ASSERT
/**
  * @brief  Reports the name of the source file and the source line number
  *   where the assert_param error has occurred.
  * @param file: pointer to the source file name
  * @param line: assert_param error line source number
  * @retval : None
  */
void assert_failed(u8* file, u32 line)
{
  /* User can add his own implementation to report the file name and line number,
     ex: printf("Wrong parameters value: file %s on line %d\r\n", file, line) */

  /* Infinite loop */
  while (1)
  {
  }
}
#endif
```

4. Vývojový diagram (ukázka funkce)



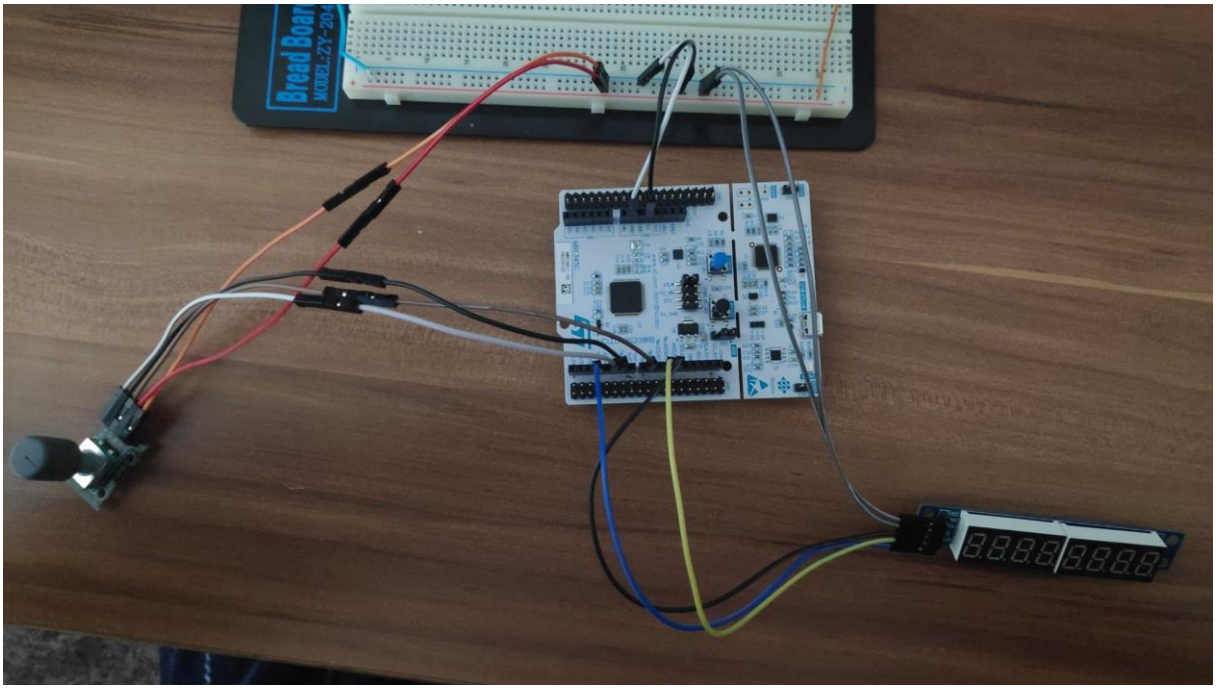


Foto konečného zapojení