Vyšší odborná škola a
Střední průmyslová škola elektrotechnická
Božetěchova 3, 772 00 Olomouc

Samostatný projekt MIT

Název projektu	Číslo projektu
HLÍDACÍ STANICE PŘÍTOMNOSTI	MIT 1

Zadání

1. Vytvořte zařízení, které bude zaznamenávat přítomnost u zařízení, dané přístupy k zařízení potom uloží v čase. Bude zde možnost prohlížet si výsledky.

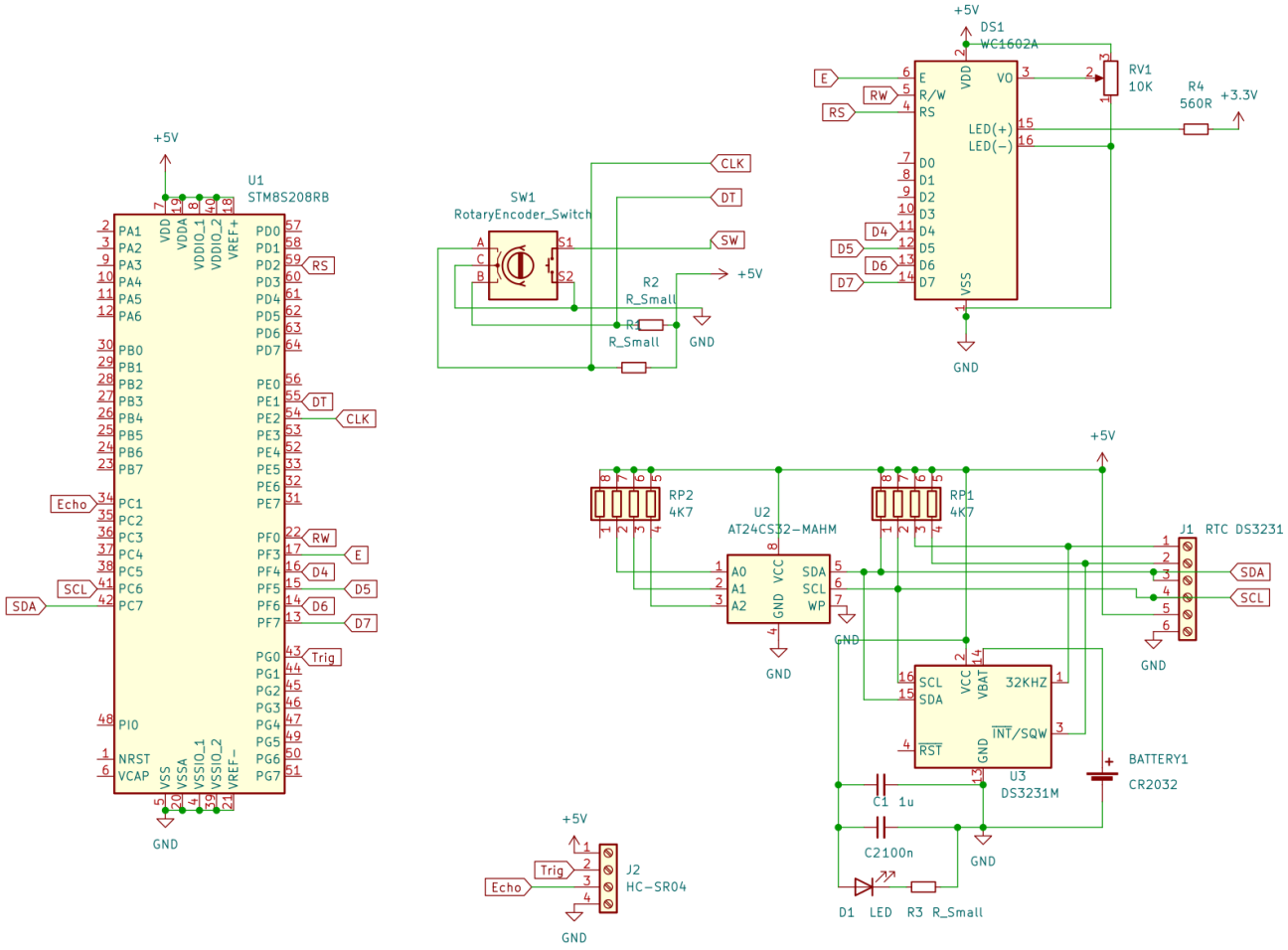
Použitý SOFTWARE: KiCAD, STVD

Celková doba vypracování: 20 hodin

Poř. č.	Příjmení a jméno	Třída	Školní rok
9	KOLÁŘ Jakub	4A	2021/22

Datum vypracování	Datum odevzdání	Počet listů	Klasifikace
20.1.2022	22.1.2022	9	

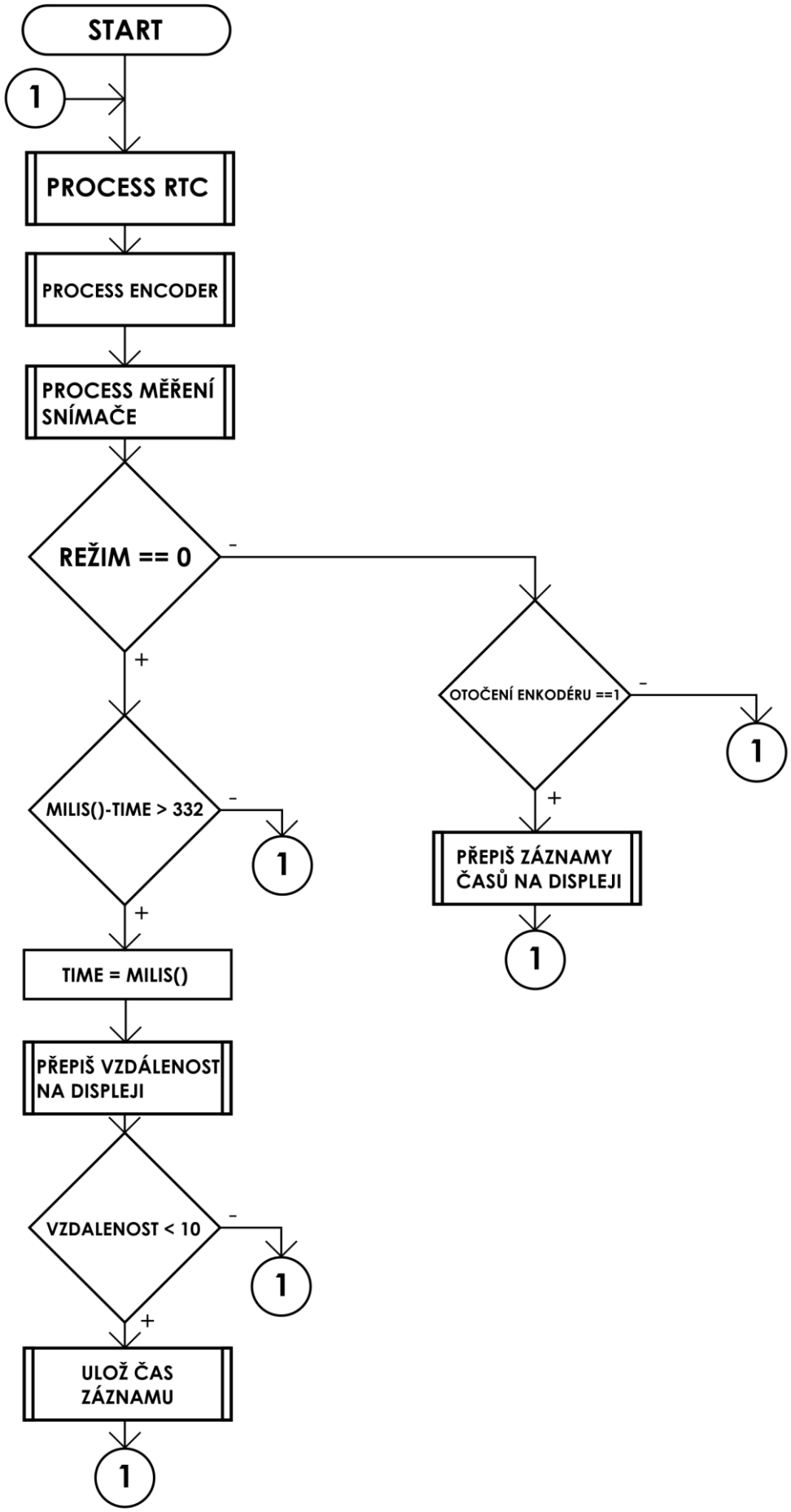
SCHÉMA ZAPOJENÍ:



SLOVNÍ POPIS ZAPOJENÍ:

Zapojení se skládá z řídícího mikrokontroléru STM8S208RB, na kterém je jumper nastaven tak, aby měl 5V vstupy a výstupy. Pomocí sběrnice I2C je připojen modul reálného času s driverem DS3231. Z modulu reálného času lze získat informace o datu, hodinách, minutách a sekundách. Další periferií je LCD displej s driverem HD44780. Ten je připojen sedmi datovými vodiči. Displej musí být napájen 5 V a na kontrastní pin musí být přivedeno napětí vytvořené na potenciometru, které se dá doladit. Ve schématu je zapojeno i podsvícení pomocí LED na displeji. Ultrazvukový snímač je ovládán pomocí dvou pinů – spouštěcího a výstupního, na jehož snímání využívám timer 1. Poslední periferií je enkodér, jehož dva piny pro snímání otočení jsou připojeni na standartní GPIO. Místo tlačítka enkodéru využívám tlačítka na vývojovém kitu.

VÝVOJOVÝ DIAGRAM PROGRAMU:



SLOVNÍ POPIS FUNKCE PROGRAMU:

Při každém průchodu while(1) se spustí funkce read_RTC(), ta slouží k vyčtení času z modulu reálného času každých 100 milisekund. Funkce proces_measurment() se stará o měření vzdálenosti ultrazvukovým čidlem. Funkce proces_enc() kontroluje stav enkodéru.

Hlavní smyčka programu dále obsahuje stavový automat pro zobrazování na displeji. V režimu 0 zobrazujeme na displeji na prvním řádku vzdálenost předmětu od snímače v centimetrech, na druhém řádku zobrazujeme aktuální čas. Zároveň se v tomto režimu zaznamenávají časy, ve kterých byl předmět ke snímači blíže než 10 cm. Při změně režimu stisknutím tlačítka vypíšeme na displej záznamy těchto časů. V druhém režimu lze pomocí enkodéru mezi záznamy listovat, celkem lze zobrazit až 10 záznamů. Při dosažení 10 záznamů se přepisují nejstarší výsledky. Při opětovném stisku tlačítka změníme režim a opět zobrazujeme to, co v režimu 0.

ZÁVĚR:

Při tomto projektu jsem se naučil pracovat s modulem reálného času RTC DS3231. Obnovil jsem své znalosti z předmětu MIT z 3. ročníku. Seznámil jsem s možností programovat mikrokontroléry STM přes operační systém Linux, jelikož zde ale vzniklo mnoho problémů, navrátil jsem se k původnímu editoru STVD na Windows.

Zařízení funguje, mohlo by se využít pro bezpečnostní účely na monitorování přístupů k určitému objektu. Nedostatkem je zatím nemožnost editovat výsledky měření a fakt, že je vše na nepájivém poli. Jako pozitivní hodnotím, že i při výpadku napájení lze opětovně získat aktuální čas pomocí baterie v modulu RTC. Jako výhled do budoucna vidím montáž na samostatnou desku plošných spojů a ukládání výsledků do paměti flash.

MAIN.C:

```
#include "stm8s.h"
#include "milis.h"
#include "stm8_hd44780.h"
#include "stdio.h"
#include "swi2c.h"
#include "stm8s.h"

#define PULSE_LEN 2 // délka spouštěcího (trigger) pulzu pro ultrazvuk
#define MEASURMENT_PERIOD 100 // perioda měření ultrazvukem (měla by být víc jak
(maximální_dosah*2)/rychlost_zvuku)
void process_measurment(void);
void init_tim1(void);
uint16_t capture; // tady bude aktuální výsledek měření (času), v mikrosekundách us
uint8_t capture_flag=0; // tady budeme indikovat že v capture je čerstvý výsledek
char text[16];
uint32_t time2=0;
uint32_t vzdalenost=0;
uint16_t vzd1=0;
#define DETEKCE_VZDALENOSTI 10

#define RTC_ADRESS 0b11010000   //Makro pro adresu RTC obvodu
void init_tim3(void);  //funkce, která nastaví timer 3
void read_RTC(void);   //funkce, která každých 100 ms vyčítá informace z RTC
volatile uint8_t error;
volatile uint8_t RTC_precteno[7];    // pole o délce 7 bytů, kam ukládám data o čase
volatile uint8_t zapis[7];                                  //pole o délce 7 bytů, ze kterého zapisuju
data do RTC
uint16_t sec,des_sec,min,des_min,hod,des_hod,zbytek_hod;
volatile bool read_flag=0;
uint8_t stav=0;


#define ENKODER_TLAC_A_GPIO GPIOE
#define ENKODER_TLAC_B_GPIO GPIOE
#define ENKODER_TLAC_GPIO GPIOE

#define ENKODER_TLAC_A_PIN GPIO_PIN_1
#define ENKODER_TLAC_B_PIN GPIO_PIN_2
#define ENKODER_TLAC_PIN GPIO_PIN_4

bool rezim=0;
uint16_t zaznamy[10][6];
uint8_t cislo_zaznamu=0;
bool zmena_rezimu=0,zmena_zaznamu=0,sepnuto=0;
uint8_t pocet_zaznamu=0;
uint8_t i=0;
void process_enc(void);
```

```
void main(void){
CLK_HSIPrescalerConfig(CLK_PRESCALER_HSIDIV1); // 16MHz z interního RC
oscilátoru
init_milis(); // milis kvůli delay_ms()
init_tim1(); // nastavit a spustit timer
lcd_init();
lcd_clear();
GPIO_Init(GPIOG, GPIO_PIN_0, GPIO_MODE_OUT_PP_LOW_SLOW); // výstup -
"trigger pulz pro ultrazvuk"

enableInterrupts();  //globálně povolí přerušení
swi2c_init();
init_tim3(); // nastavit a spustit timer
zapis[0] = 0b00000000;
zapis[1] = 0b00110010;
zapis[2] = 0b00010010;//první půlka desítky, druhá jednotky
//swi2c_write_buf(RTC_ADRESS,0x00,zapis,3);


  while (1){

                read_RTC();
                process_measurment(); // obsluhuje neblokujícím způsobem měření
ultrazvukem
                process_enc();//obsluhuje


                switch(rezim){
                        case 0:
                                if (read_flag){//zobrazení aktuálního času
                                        read_flag=0;
                                        sprintf(text,"time:
%u%u:%u%u:%u%u",des_hod,hod,des_min,min,des_sec,sec);//
                                        lcd_gotoxy(0,1);
                                        lcd_puts(text);
                                }
                                //pocet_zazanmu++;
                                if (milis()-time2>332){//zobrazení aktuální vzdálenosti
                                        time2=milis();
                                        vzdalenost=capture/2;
                                        vzdalenost=vzdalenost*343;
                                        vzd1=vzdalenost/10000;
                                        sprintf(text,"distance: %3ucm",vzd1);
                                        lcd_gotoxy(0,0);
                                        lcd_puts(text);

                                        switch(sepnuto){
                                                case 0:
```

```
                                        if
(vzd1<DETEKCE_VZDALENOSTI){//detekce překročení minimální vzdálenosti vede k
uložení výsledku
                                                pocet_zaznamu++;

                                                if(pocet_zaznamu>10){
                                                        for(i=0;i<9;i++){

        zaznamy[i][0]=zaznamy[i+1][0];

        zaznamy[i][1]=zaznamy[i+1][1];

        zaznamy[i][2]=zaznamy[i+1][2];

        zaznamy[i][3]=zaznamy[i+1][3];

        zaznamy[i][4]=zaznamy[i+1][4];

        zaznamy[i][5]=zaznamy[i+1][5];
                                                                pocet_zaznamu=10;
                                                        }
                                                }

                                                zaznamy[pocet_zaznamu-
1][0]=sec;
                                                zaznamy[pocet_zaznamu-
1][1]=des_sec;
                                                zaznamy[pocet_zaznamu-
1][2]=min;
                                                zaznamy[pocet_zaznamu-
1][3]=des_min;
                                                zaznamy[pocet_zaznamu-
1][4]=hod;
                                                zaznamy[pocet_zaznamu-
1][5]=des_hod;
                                                sepnuto=1;
                                        }
                                        break;

                                case 1:
                                        if
(vzd1>DETEKCE_VZDALENOSTI+3){
                                                sepnuto=0;
                                        }
                                        break;
                                }
                        }
```

```
                    if (zmena_rezimu){//stisk tlačítka vede program do druhého
režimu
                        zmena_rezimu=0;
                        cislo_zaznamu=0;
                        lcd_clear();

                        if(pocet_zaznamu==0){
                            sprintf(text,"EMPTY");
                            lcd_gotoxy(0,0);
                            lcd_puts(text);
                        }

                        else if(pocet_zaznamu==1){
                            sprintf(text,"%u:
%u%u:%u%u:%u%u",cislo_zaznamu+1,zaznamy[cislo_zaznamu][5],zaznamy[cislo_zaznam
u][4],zaznamy[cislo_zaznamu][3],zaznamy[cislo_zaznamu][2],zaznamy[cislo_zaznamu][1],z
aznamy[cislo_zaznamu][0]);
                            lcd_gotoxy(0,0);
                            lcd_puts(text);
                        }

                        else{
                            sprintf(text,"%u:
%u%u:%u%u:%u%u",cislo_zaznamu+1,zaznamy[cislo_zaznamu][5],zaznamy[cislo_zaznam
u][4],zaznamy[cislo_zaznamu][3],zaznamy[cislo_zaznamu][2],zaznamy[cislo_zaznamu][1],z
aznamy[cislo_zaznamu][0]);
                            lcd_gotoxy(0,0);
                            lcd_puts(text);

                            sprintf(text,"%u:
%u%u:%u%u:%u%u",cislo_zaznamu+2,zaznamy[cislo_zaznamu+1][5],zaznamy[cislo_zazna
mu+1][4],zaznamy[cislo_zaznamu+1][3],zaznamy[cislo_zaznamu+1][2],zaznamy[cislo_zazn
amu+1][1],zaznamy[cislo_zaznamu+1][0]);
                            lcd_gotoxy(0,1);
                            lcd_puts(text);
                        }
                        rezim++;
                    }
                break;




                case 1:
                    if (zmena_zaznamu){//otočení enkodéru vede ke změně hodnot
na displeji
                        lcd_clear();
                        zmena_zaznamu=0;
                        if(pocet_zaznamu>0 && pocet_zaznamu<20){
```

```
                                        sprintf(text,"%u:
%u%u:%u%u:%u%u",cislo_zaznamu+1,zaznamy[cislo_zaznamu][5],zaznamy[cislo_zaznam
u][4],zaznamy[cislo_zaznamu][3],zaznamy[cislo_zaznamu][2],zaznamy[cislo_zaznamu][1],z
aznamy[cislo_zaznamu][0]);
                                        lcd_gotoxy(0,0);
                                        lcd_puts(text);
                                    }

                                    if(cislo_zaznamu==9){
                                        sprintf(text,"%u:
%u%u:%u%u:%u%u",1,zaznamy[0][5],zaznamy[0][4],zaznamy[0][3],zaznamy[0][2],zaznam
y[0][1],zaznamy[0][0]);
                                        lcd_gotoxy(0,1);
                                        lcd_puts(text);
                                    }
                                    else if(pocet_zaznamu>1 && pocet_zaznamu<20){
                                        sprintf(text,"%u:
%u%u:%u%u:%u%u",cislo_zaznamu+2,zaznamy[cislo_zaznamu+1][5],zaznamy[cislo_zazna
mu+1][4],zaznamy[cislo_zaznamu+1][3],zaznamy[cislo_zaznamu+1][2],zaznamy[cislo_zazn
amu+1][1],zaznamy[cislo_zaznamu+1][0]);
                                        lcd_gotoxy(0,1);
                                        lcd_puts(text);
                                    }
                                }

                            //do something
                                if (zmena_rezimu){
                                    zmena_rezimu=0;
                                    lcd_clear();
                                    rezim++;
                                }

                            break;

                            default:
                                rezim=0;
                        }
  }
}



void process_enc(void){
    static bool minuleA=0; // pamatuje si minulý stav vstupu A (nutné k detekování
sestupné hrany)
    static bool minuleB=0; // 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
    static uint32_t pocatek_stisku=0;
```

```
	static bool minule_stisk=0,ted_stisk=0,konec_stisku=0;

		if
(GPIO_ReadInputPin(ENKODER_TLAC_GPIO,ENKODER_TLAC_PIN)==RESET){
			ted_stisk=1;
		}
		else{ted_stisk=0;}

		if((ted_stisk==1) && (minule_stisk==0)){pocatek_stisku=milis();}
		if(ted_stisk==0 && minule_stisk==1){konec_stisku=1;}

		if (konec_stisku==1 && ((milis()-pocatek_stisku)>999)){
			konec_stisku=0;
		}
		else if(konec_stisku==1 && ((milis()-pocatek_stisku)<1000)){
			zmena_rezimu=1;
			konec_stisku=0;
		}


	if((GPIO_ReadInputPin(ENKODER_TLAC_B_GPIO,ENKODER_TLAC_B_PIN) !=
RESET) && minuleB==0 &&
GPIO_ReadInputPin(ENKODER_TLAC_A_GPIO,ENKODER_TLAC_A_PIN) ==
RESET){
		zmena_zaznamu=1;
		cislo_zaznamu--;

		if (cislo_zaznamu>9){//přeteče
			if(pocet_zaznamu==1){
				cislo_zaznamu=0;
			}
			else{cislo_zaznamu=pocet_zaznamu-2;}
		}
	}
	if((GPIO_ReadInputPin(ENKODER_TLAC_B_GPIO,ENKODER_TLAC_B_PIN)
== RESET) && minuleB==1 &&
GPIO_ReadInputPin(ENKODER_TLAC_A_GPIO,ENKODER_TLAC_A_PIN) != RESET){
		zmena_zaznamu=1;
		cislo_zaznamu--;
		if (cislo_zaznamu>9){//přeteče
			if(pocet_zaznamu==1){
				cislo_zaznamu=0;
			}
			else{cislo_zaznamu=pocet_zaznamu-2;}
		}
	}

	if((GPIO_ReadInputPin(ENKODER_TLAC_A_GPIO,ENKODER_TLAC_A_PIN) !=
RESET) && minuleA==0 &&
```

```
GPIO_ReadInputPin(ENKODER_TLAC_B_GPIO,ENKODER_TLAC_B_PIN) ==
RESET){
                zmena_zaznamu=1;
                cislo_zaznamu++;
                if (cislo_zaznamu+1>(pocet_zaznamu-1)){
                        cislo_zaznamu=0;
                }
        }
        if((GPIO_ReadInputPin(ENKODER_TLAC_A_GPIO,ENKODER_TLAC_A_PIN)
== RESET) && minuleA==1 &&
GPIO_ReadInputPin(ENKODER_TLAC_B_GPIO,ENKODER_TLAC_B_PIN) != RESET){
                zmena_zaznamu=1;
                cislo_zaznamu++;
                if (cislo_zaznamu+1>(pocet_zaznamu-1)){
                        cislo_zaznamu=0;
                }

        }

        if(GPIO_ReadInputPin(ENKODER_TLAC_A_GPIO,ENKODER_TLAC_A_PIN) !=
RESET){minuleA = 1;} // pokud je vstup A v log.1
        else{minuleA=0;}
        if(GPIO_ReadInputPin(ENKODER_TLAC_B_GPIO,ENKODER_TLAC_B_PIN) !=
RESET){minuleB = 1;} // pokud je vstup A v log.1
        else{minuleB=0;}

        minule_stisk=ted_stisk;
}



void read_RTC(void){
static uint16_t last_time=0;      // každých 100ms přečte obsah RTC
  if(milis() - last_time >= 100){
    last_time = milis();
    error=swi2c_read_buf(RTC_ADRESS,0x00,RTC_precteno,7);

                sec = (RTC_precteno[0] & 0b00001111);            //sekundy
                des_sec = ((RTC_precteno[0] >> 4) & 0b00001111);                 //desítky
sekund
                min = (RTC_precteno[1] & 0b00001111);                          //minuty
                des_min = ((RTC_precteno[1] >> 4) & 0b00001111);   //desítky minut
                hod = (RTC_precteno[2] & 0b00001111);
        //hodiny
                des_hod = ((RTC_precteno[2] >> 4) & 0b00000011);  //desítky hodin
                zbytek_hod = ((RTC_precteno[2] >> 4) & 0b00001111);   //zbytek dat hodin
        }
}

void init_tim3(void){
```

```
TIM3_TimeBaseInit(TIM3_PRESCALER_16,1999); // clock 1MHz, strop 5000 => perioda přetečení 5 ms
TIM3_ITConfig(TIM3_IT_UPDATE, ENABLE); // povolíme přerušení od update události (přetečení) timeru 3
TIM3_Cmd(ENABLE); // spustíme timer 3
}


INTERRUPT_HANDLER(TIM3_UPD_OVF_BRK_IRQHandler, 15){    //funkce pro obsluhu displejů
  TIM3_ClearITPendingBit(TIM3_IT_UPDATE);
        read_flag=1;
}


void process_measurment(void){
        static uint8_t stage=0; // stavový automat
        static uint16_t time=0; // pro časování pomocí milis
        switch(stage){
        case 0: // čekáme než uplyne  MEASURMENT_PERIOD abychom odstartovali měření
                if(milis()-time > MEASURMENT_PERIOD){
                        time = milis();
                        GPIO_WriteHigh(GPIOG,GPIO_PIN_0); // zahájíme trigger pulz
                        stage = 1; // a bdueme čekat až uplyne čas trigger pulzu
                }
                break;
        case 1: // čekáme než uplyne PULSE_LEN (generuje trigger pulse)
                if(milis()-time > PULSE_LEN){
                        GPIO_WriteLow(GPIOG,GPIO_PIN_0); // ukončíme trigger pulz
                        stage = 2; // a přejdeme do fáze kdy očekáváme echo
                }
                break;
        case 2: // čekáme jestli dostaneme odezvu (čekáme na echo)
                if(TIM1_GetFlagStatus(TIM1_FLAG_CC2) != RESET){ // hlídáme zda timer hlásí změření pulzu
                        capture = TIM1_GetCapture2(); // uložíme výsledek měření
                        capture_flag=1; // dáme vědět zbytku programu že máme nový platný výsledek
                        stage = 0; // a začneme znovu od začátku
                }else if(milis()-time > MEASURMENT_PERIOD){ // pokud timer nezachytil pulz po dlouhou dobu, tak echo nepřijde
                        stage = 0; // a začneme znovu od začátku
                }
                break;
        default: // pokud se cokoli pokazí
        stage = 0; // začneme znovu od začátku
        }
}
```

```
void init_tim1(void){
GPIO_Init(GPIOC, GPIO_PIN_1, GPIO_MODE_IN_FL_NO_IT); // PC1 (TIM1_CH1) jako vstup
TIM1_TimeBaseInit(15,TIM1_COUNTERMODE_UP,0xffff,0); // timer necháme volně běžet (do maximálního stropu) s časovou základnou 1MHz (1us)
// Konfigurujeme parametry capture kanálu 1 - komplikované, nelze popsat v krátkém komentáři
TIM1_ICInit(TIM1_CHANNEL_1,TIM1_ICPOLARITY_RISING,TIM1_ICSELECTION_DIRECTTI,TIM1_ICPSC_DIV1,0);
// Konfigurujeme parametry capture kanálu 2 - komplikované, nelze popsat v krátkém komentáři
TIM1_ICInit(TIM1_CHANNEL_2,TIM1_ICPOLARITY_FALLING,TIM1_ICSELECTION_INDIRECTTI,TIM1_ICPSC_DIV1,0);
TIM1_SelectInputTrigger(TIM1_TS_TI1FP1); // Zdroj signálu pro Clock/Trigger controller
TIM1_SelectSlaveMode(TIM1_SLAVEMODE_RESET); // Clock/Trigger má po příchodu signálu provést RESET timeru
TIM1_ClearFlag(TIM1_FLAG_CC2); // pro jistotu vyčistíme vlajku signalizující záchyt a změření echo pulzu
TIM1_Cmd(ENABLE); // spustíme timer ať běží na pozadí
}
#ifdef USE_FULL_ASSERT
void assert_failed(u8* file, u32 line)
{
  while (1){}
}
#endif
```