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

Samostatný projekt MIT

Název projektu	Číslo projektu
ZOBRAZOVACÍ JEDNOTKA ČASU A VZDÁLENOSTI	MIT 2

Zadání

1. Vytvořte jednotku, která se bude starat o zobrazování aktuální času a vzdálenosti od snímače.

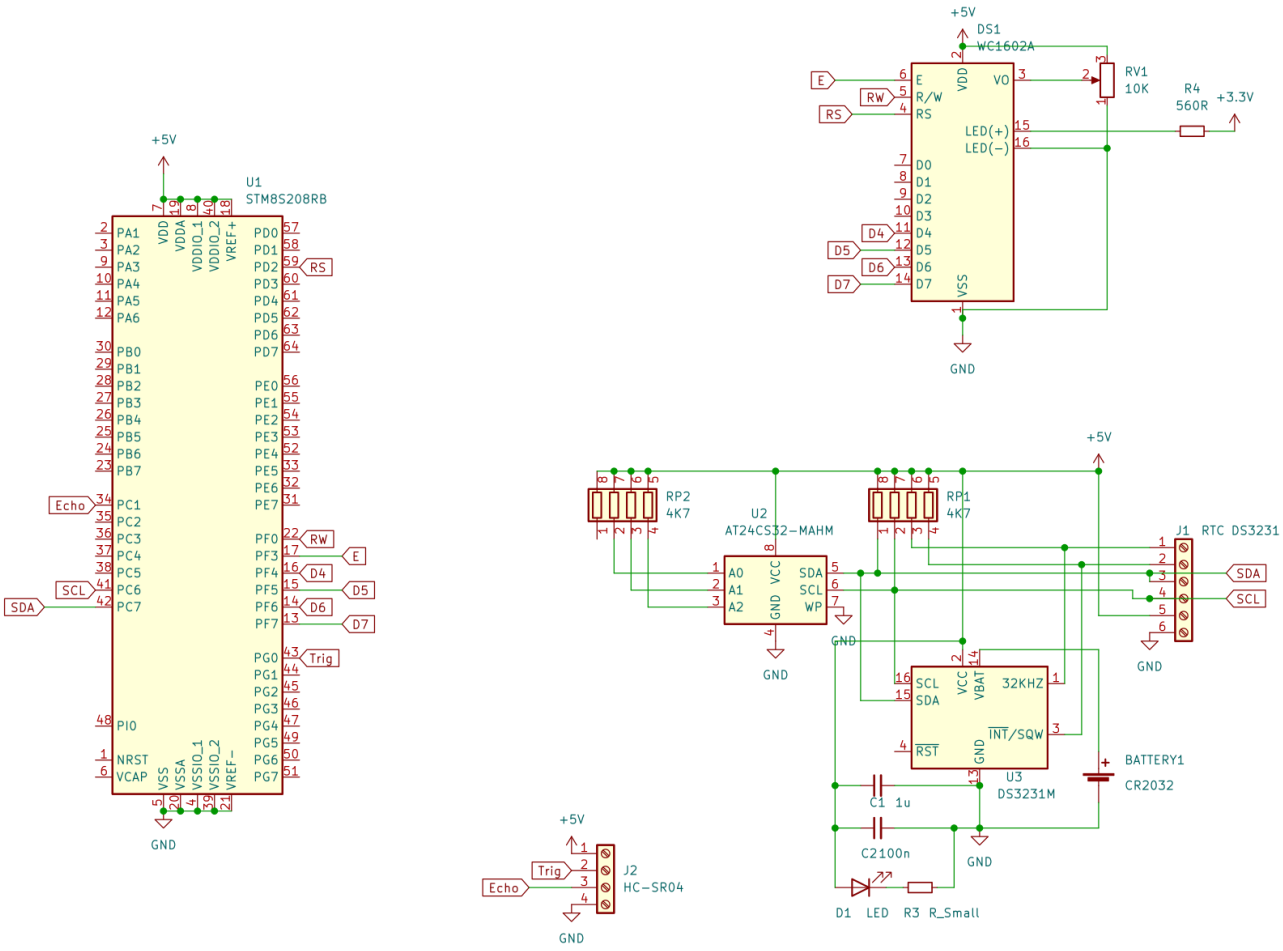
Použitý SOFTWARE: KiCAD, STVD

Celková doba vypracování: 20 hodin

Poř. č.	Příjmení a jméno	Třída	Školní rok
3	BRZOBOHATÝ Bohdan	4A	2021/22

Datum vypracování	Datum odevzdání	Počet listů	Klasifikace
23.4.2022	27.4.2022	8	

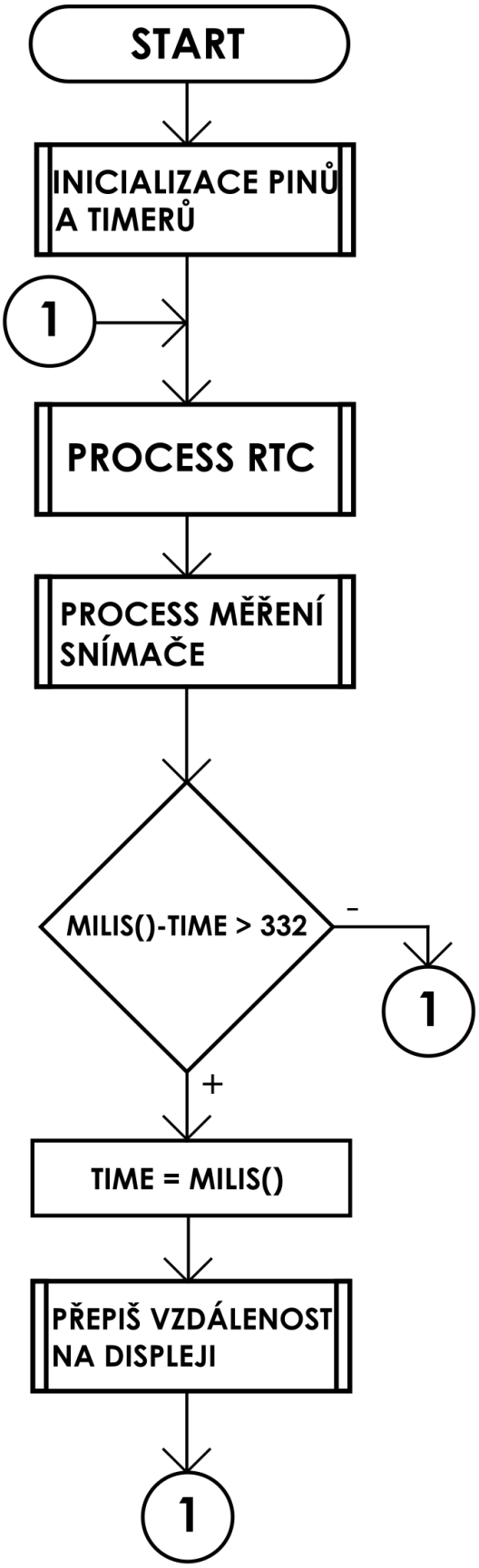
SCHÉMA ZAPOJENÍ:



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 potenciomteru, 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.

VÝVOJOVÝ DIAGRAM PROGRAMU:



SLOVNÍ POPIS FUNKCE PROGRAMU:

Na začátku programu se inicializují piny pro všechny komponenty a také timery. Při každém průchodu programem se ve funkci read_RTC zkontroluje, zda se čas aktualizoval. Jestliže ano, v hlavní smyčce se nový čas zobrazí na displeji. Při každém průchodu programem se také zkontrolují hodnoty snímače, které se potom zobrazí na displeji. Každých 333 ms dojde k přepisu vzdálenosti na displeji. Přepis času probíhá každou sekundu.

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, může se využít jako měřící stanice v různých oblastech. Nedostatkem je 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.

MAIN.C:

```
//Bohdan Brzobohatý
#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;

uint8_t i=0;


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] = 0b00001000;
zapis[2] = 0b00010101;//první půlka desítky, druhá jednotky
//swi2c_write_buf(RTC_ADRESS,0x00,zapis,3);


  while (1){

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


                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);
                }

                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);
                }


  }
}




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_D
IRECTTI,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
```