Enostavni avtomat za mešanje sokov z RTOS

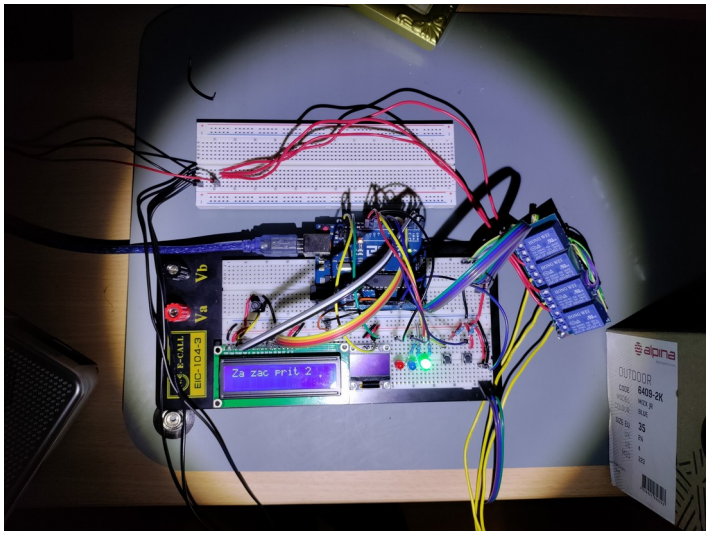
Matej Poljanšek, 64190119

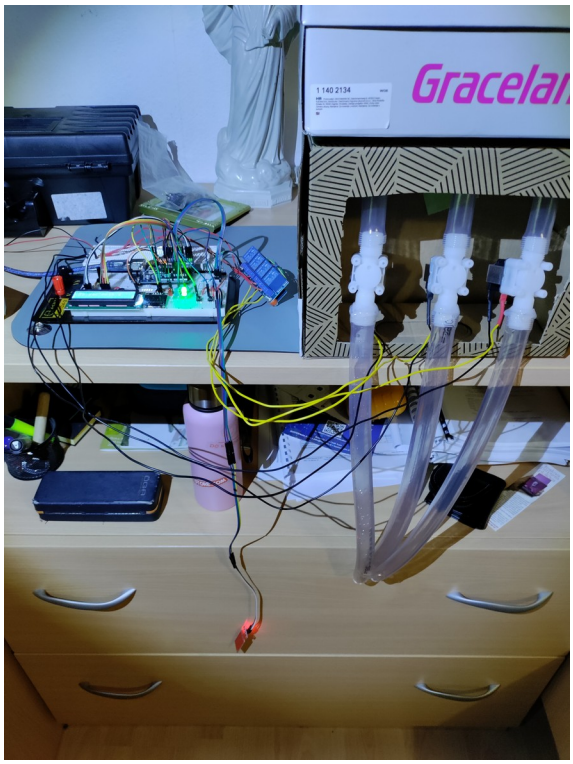
3.6.2022

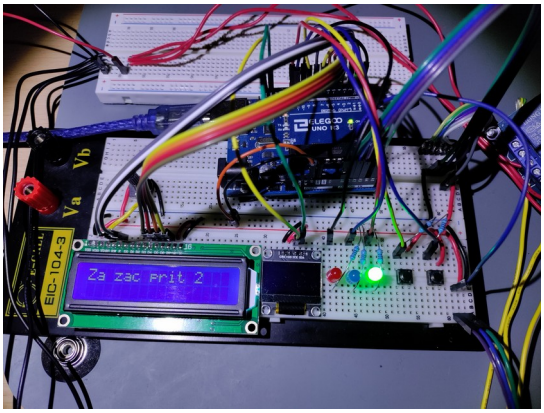
Modul B: Vgrajeni sistemi

Uvod

Pri tem projektu zasnujemo enostaven avtomat za točenje treh različnih tekočin, ki jih uporabnik ločeno določi količino. Za osnovo uporabimo mikrokrmilnik atmega328p, nameščen na razvojni ploščici Arduino Uno, za kontrolo količine tekočin pa uporabimo DC elektromagnetne ventile. Cilj projekta je delujoč sistem za relativno enostavno doziranje do treh tekočin v isti kozarec s prikazom izbranih nivojev tekočin in LED prikazom trenutnega koraka v postopku. Prav tako implementiramo sistem za zaznavanje preliva tekočine, LCD zaslon z enostavnimi navodili in prikazom stanja postopka in OLED zaslon s prikazom izbire količine posameznih tekočin.



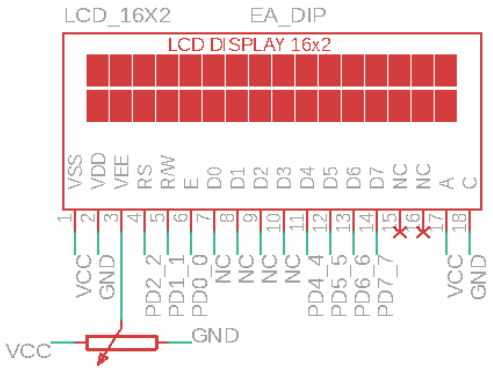


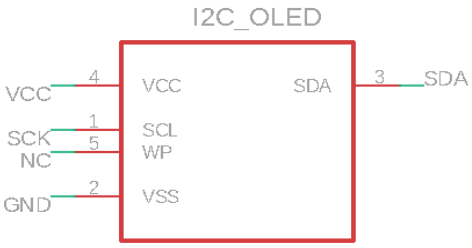


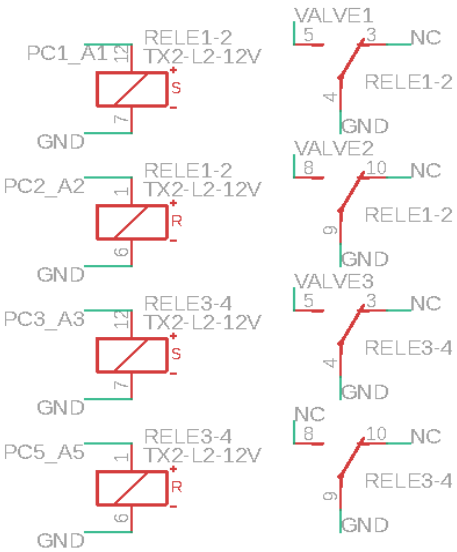
Seznam komponent

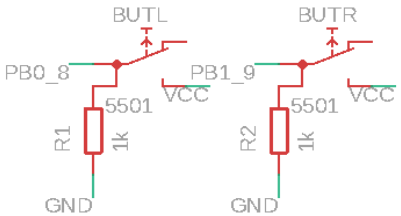
- Atmega328p razvojna plošča Arduino Uno
- rdeča LED
- modra LED
- zelena LED
- tipki 2x
- 10 kΩ potenciometer
- 1 kΩ 5x
- I2C SSD1306 12864 OLED zaslon
- 16x2 LCD zaslon s Hitachi HD44780 gonilnikom
- Hoya water level detection sensor module
- SailFlo elektromagnetni ventil DC12V 4x

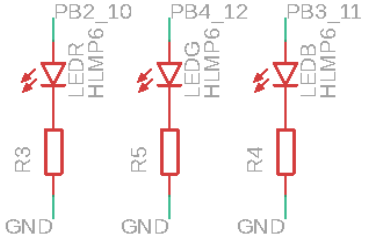
Shema elektronike









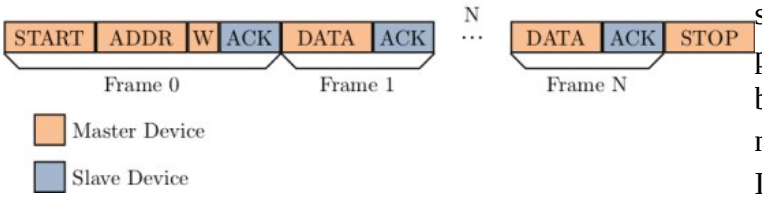


Opis komunikacije

Z vsemi perifernimi enotami razen OLED zaslonom komuniciramo preko navadnih vhodno-izhodnih enot in ni posebnih komunikacijskih protokolov, razen za LCD zaslon, kjer komuniciramo preko 4-bitnega podatkovnega in 3-bitnega kontrolnega vodila. Za komunikacijo skrbi knjižnica lcd.h.

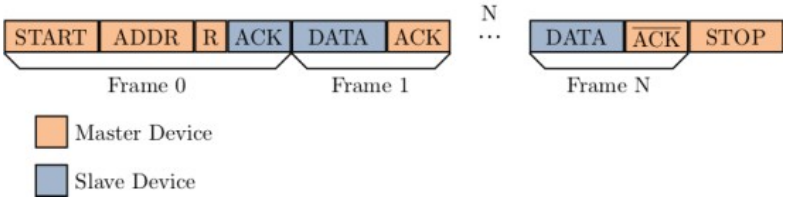
Z OLED komuniciramo preko I2C vodila. I2C je serijsko vodilo, pri katerem nastavimo mikrokrmilnik kot master in OLED zaslon kot slave napravo. Master komunicira na vodilu z naslavljanjem slave naprave in zahtevo po branju ali pisanju na napravo. Sam potek komunikacije je prikazan s spodnjima diagramoma Slika 1 in Slika 2.

Oba procesa se začneta z začetnim okvirjem, kjer master na vodilo zapiše s katero napravo komunicira in v katero smer, na kar se slave odzove. Temu sledijo podatkovni paketi oblike podatek, potrditev. Pri branju z naprave master konča proces z negiranim odzivom in stop signalom. Izmenjava podatkov pri branju poteka v paketih po 1 bajt.



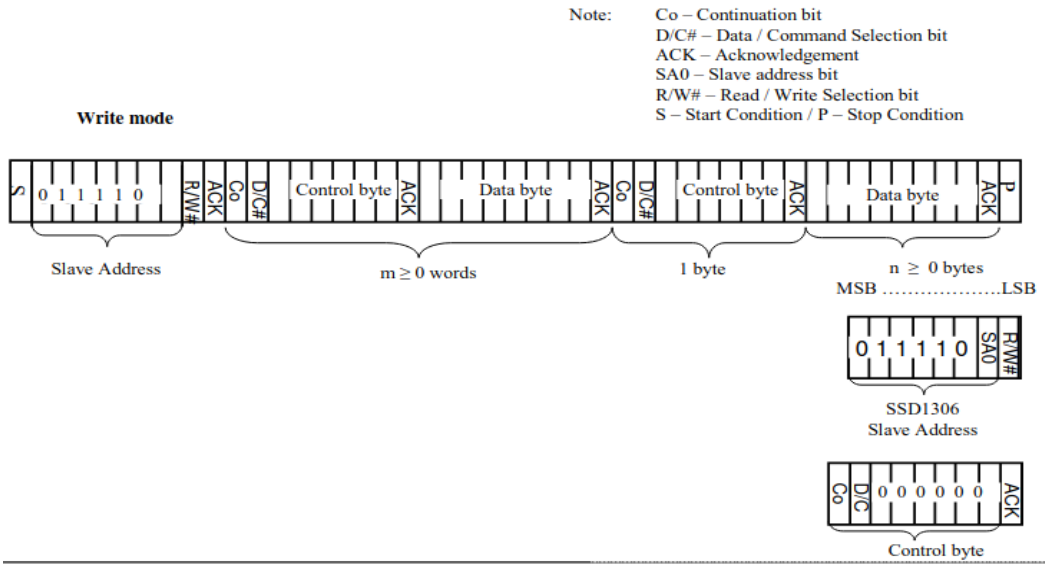
Slika 1: pisanje na napravo

OLED deluje v na principu vrstic in strani. Vsaka vrstica je razdeljena na strani s po 8 piksli. Zapisujemo stran po stran, kot je prikazano na spodnjem diagramu Slika 3. S kontrolnim bajtom povemo OLED zaslonu lokacijo pisanja, z data bajtom pa same podatke za pisanje.



Slika 2: branje z naprave

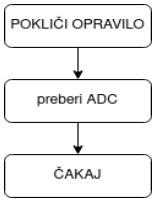
Za komunikacijo uporabimo knjižnico SSD1306.h, ki pripravi in pošlje podatke na OLED zaslon. Ima že vgrajeno funkcijo za izris grafikonov, ki je v tem primeru zelo uporabna za prikaz izbranih nivojev v stolpčnih grafikonih. Knjižnica za samo I2C komunikacijo uporablja zunanjo knjižnico I2C.h.



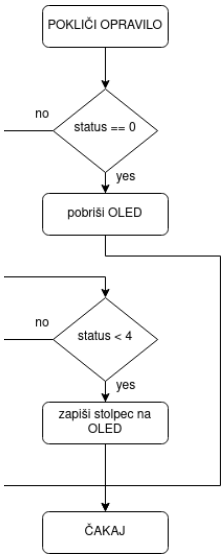
Slika 3: komunikacija z OLED

Opis programa

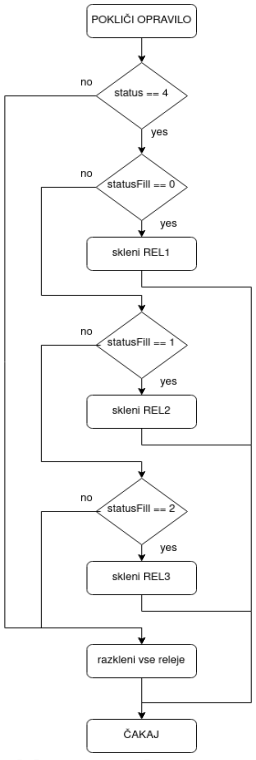
Slika 6: main.c



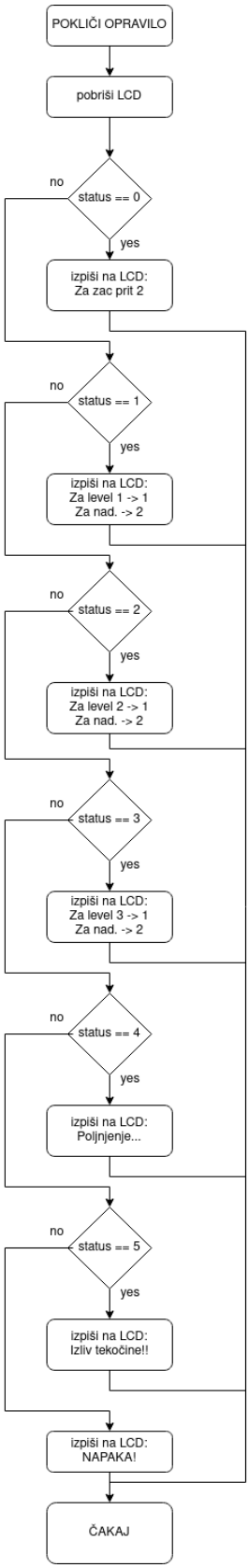
Slika 5: task4: ADC varovalo



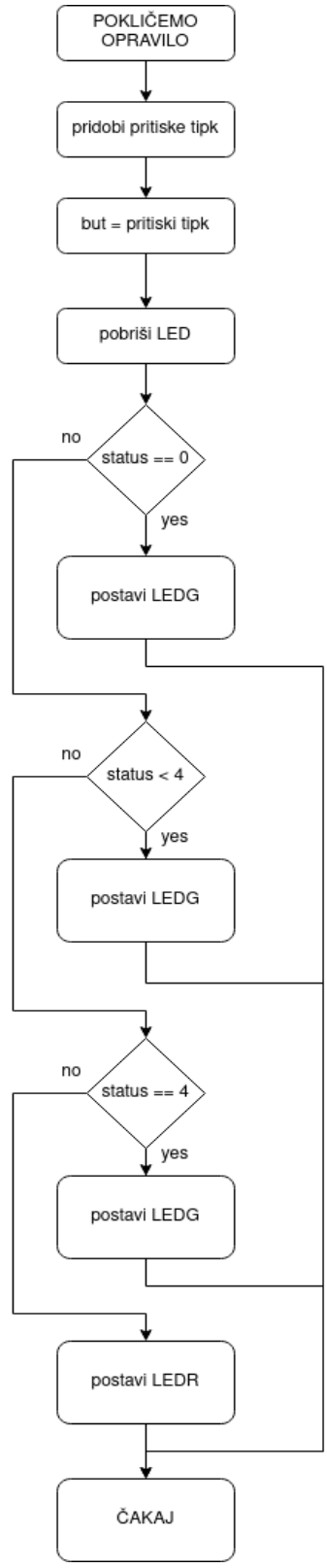
Slika 9: task2: OLED



Slika 8: task3: releji



Slika 7: task1: LCD



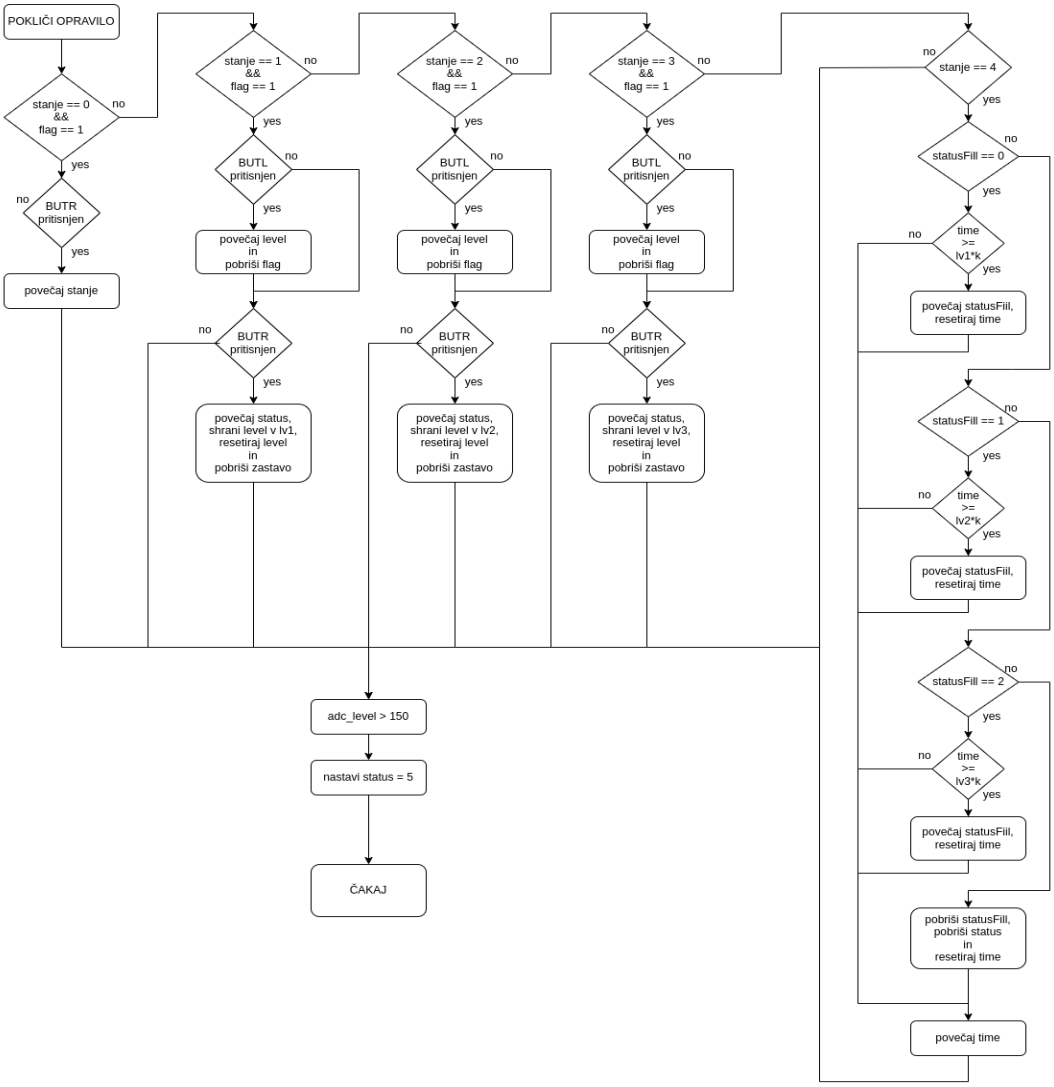
Slika 4: task0: tipke in LED

Uporabimo kooperativno ciklično razvrščanje, za kontrolo skrbi zunanja knjižnica OS.h. Uporabimo 6 opravil, ki se ciklično ponavljajo. Vsako opravilo traja 20 ms, med sabo pa komunicirajo z zunanjimi spremenljivkami, deklariranimi v globals.h. Posamezna opravila so prikazana v obliki diagramov poteka. Nastavitve posameznih perifernih komponent so izvedene s funkcijami, deklariranimi v utils.h, uporabljeni makroti pa so deklarirani v config.h, kar omogoča enostavno modifikacijo, v kolikor bi potrebovali prestaviti komponente na druge priključke.

Task5 je kontrolno opravilo, ki skrbi za preklapljanje med stanji. Koda je priložena na koncu.

Stanja so sledeča:

- začetno stanje
- izbira 1. nivoja tekočine
- izbira 2. nivoja tekočine
- izbira 3. nivoja tekočine
- polnjenje:
    - polnjenje 1. tekočine
    - polnjenje 2. tekočine
    - polnjenje 3. tekočine
- stanje preliva tekočine



Slika 10: task5: kontrolno opravilo

Izzivi

Pri sami izvedbi je največji izziv najti knjižnico za kooperativno ciklično razvrščanje. Prva knjižnica v testni uporabi namreč ni poskrbela za samodejno nastavitev naslednjega opravila in je zahtevala opravila v obliki neskončnih zank, ki so v tem primeru neustrezne zaradi adp, LCD in OLED, saj bi nastala težava, če bi prekmalu prekinili prenos podatkov..

Zato uporabimo OS, ki te pomanjkljivosti nima. Potrebno je samo urediti komunikacijo med funkcijami tipa void f(void), kar storimo z uporabo spremenljivk tipa extern.

Koeficient k, ki določa potrebno število ciklov za polnjenje, dobimo iz enačbe:

k = 9 s / 120 ms (9s je čas, potreben za polnjenje 1 enote tekočine, 120 ms je čas, v katerem je task5 spet na vrsti).

S strani strojne opreme je najpočasnejši element elektromagnetni ventil, ki ni optimalen za tovrstno uporabo. Za namen projekta je zadovoljiv, vendar bi pri ponovni izbiri posvetil več časa sami pretočnosti ventila.

Vsi grafi in sheme so dostopne poleg kode na GitHub repozitoriju.

Viri

1. SOLOMON SYSTECH; SSD1306, 128 x 64 Dot Matrix OLED/PLED Segment/Common Driver with Controller; Dostopno: https://www.elecrow.com/download/SSD1306%20Datasheet.pdf, [Dostopano: 2.6.2022]
2. Mr. Dr. Prof. Bolt; Controlling the SSD1306 OLED through I2C; Dostopno: https://mrdrprofbolt.wordpress.com/2020/04/23/controlling-the-ssd1306-oled-through-i2c/; [Dostopano: 1.6.2022]
3. Arduino Uno pinnout; Dostopno: https://i.chillrain.com/wp-content/uploads/2017/08/arduino-uno-pin-1024x729.jpg; [Dostopano 15.5.2022]
4. Atmel Corporation; ATmega328P DATASHEET; Dostopno: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf; [Dostopano: 14.5.2022]
5. Ferenc Németh; avr-simple-scheduler; Dostopno: https://github.com/ferenc-nemeth/avr-simple-scheduler, [Dostopano: 20.5.2022]

Koda

Celoten projekt je dostopen na: https://github.com/matejpolj/modul-B_projket/

Spodaj je glavna projektna koda.

Main.c

```
/**
    /mnt/a4b5d085-eea2-4a2e-847d-8e6738b9fd2c/arduino-1.8.13/hardware/tools/avr/bin/avrdude -C/mnt/a4b5d085-eea2-4a2e-847d-8e6738b9fd2c/arduino-1.8.13/hardware/tools/avr/etc/avrdude.conf -v -patmega328p -carduino -P/dev/ttyACM0 -b115200 -D -Uflash:w:/tmp/arduino_build_907704/Blink.ino.hex:i
**/

#include <avr/io.h>
#include <avr/interrupt.h>
#include "OS/OS.h"
#include "Task0/Task0.h"
#include "Task1/Task1.h"
#include "Task2/Task2.h"
#include "Task3/Task3.h"
#include "Task4/Task4.h"
#include "Task5/Task5.h"
#include "utils.h"
#include "globals.h"

/* Timer interrupt. The OS_TaskTimer() shall be here. */
ISR (TIMER1_OVF_vect)
{
    TCNT1 = 65223; /// 20 ms
    OS_TaskTimer();
}

/* Main function. */
int16_t main(void)
```

```
27 {
28    /// nastavimo vse periferne enote
29    configureLEDs();
30    configureLCD();
31    coifigureButtons();
32    cofigureWaterLevel();
33    configureReles();
34    configureOLED();
35
36    TCNT1 = 65223;   /// 20 ms
37
38     TCCR1A = 0x00;
39     TCCR1B = (1<<CS10) | (1<<CS12);;  /// casovnik s 1024 prescalerjem
40     TIMSK1 = (1 << TOIE1) ;   /// omogoci timer1 overflow interrupt(TOIE1)
41     sei();        /// omogoci globalne interpte
42
43    /// nastavimo vse taske
44    OS_TaskCreate(&Task0_LED_n_BUT, 1, BLOCKED);
45    OS_TaskCreate(&Task1_LCD, 6, BLOCKED);
46    OS_TaskCreate(&Task2_OLED, 6, BLOCKED);
47    OS_TaskCreate(&Task3_Rele, 6, BLOCKED);
48    OS_TaskCreate(&Task4_adc, 6, BLOCKED);
49    OS_TaskCreate(&Task5_statusMaker, 6, BLOCKED);
50
51    /* The infinte loop, only the OS_TaskExecution() function shall be here. */
52    while (1)
53    {
54       OS_TaskExecution();
55    }
56    return 0;
57 }
58 config.h
```

```
59 #ifndef CONFIG_H_INCLUDED
60 #define CONFIG_H_INCLUDED
61
62 #include <avr/io.h>
63
64 /// komanda: /mnt/a4b5d085-eea2-4a2e-847d-8e6738b9fd2c/arduino-1.8.13/hardware/tools/avr/bin/avrdude -C/mnt/a4b5d085-eea2-4a2e-847d-8e6738b9fd2c/arduino-1.8.13/hardware/tools/avr/etc/avrdude.conf -v -patmega2560 -cwiring -P/dev/ttyUSB1 -b115200 -D -U
65
66
67 // definiramo vse priključke za LED
68 #define LEDR PB2 //10
69 #define LEDG PB4 //12
70 #define LEDB PB3 //11
71 #define LEDsys  PB5//13
72
73 // definirajmo vse tipke
74 #define BUTL PB0 //8
75 #define BUTR PB1 //9
76
77 // definirajmo waterlevel meter
78 //#define d 1
79
80 // definiramo vse priključke za releje
81 #define REL1 PC1 //a1
82 #define REL2 PC2 //a2
83 #define REL3 PC3 //a3
84 #define REL4 PC5 //a5
85
86
87 // definirajmo vse za adc
88 #define ADCchannel 0
```

```
89 #endif // CONFIG_H_INCLUDED
90 ledNbutton.c
91 #include "ledNbutton.h"
92 #include "utils.h"
93 #include "config.h"
94
95 void ledClr(int led_p) {
96     /// pobrisi LED
97     PORTB &= ~(1<<led_p);
98 }
99
100 void ledSet(int led_p) {
101     /// postavi LED
102     PORTB |= (1<<led_p);
103 }
104
105 void ledTog(int led_p) {
106     /// togle LED
107     PORTB ^= (1<<led_p);
108 }
109
110 int getButtonSta(int btn_p) {
111     /// pridobi stanje izbrane tipke
112     int state = (PINB & (1<<btn_p)) >> btn_p;
113     return state;
114 }
115
116 int getButtonPrs(int btn_p) {
117     /// pridobi pritisk izbrane tipke
118     static int state_old = 0;
119
120     int state = (PINB & (1<<btn_p)) >> btn_p;
```

```
121
122     if (state_old != state) {
123         state_old = state;
124         if (state == 1) {
125             return 1;
126         }
127     }
128     return 0;
129
130 }
131
132 int adc_read(void) {
133     /// izberemo ADC kanal in masko
134     ADMUX = (ADMUX & 0xF0) | (ADCchannel & 0x0F);
135     /// nastavimo single conversion mode
136     ADCSRA |= (1<<ADSC);
137     /// pocakamo rezultat
138     while( ADCSRA & (1<<ADSC) );
139     return ADC;
140 }
141
142 void releClr(int rele_p) {
143     /// pobrisi (postavi v neg logiki) rele
144     PORTC &= ~(1<<rele_p);
145 }
146
147 void releSet(int rele_p) {
148     /// postavi (pobrisi v neg logiki) rele
149     PORTC |= (1<<rele_p);
150 }
151 ledNbutton.h
152 #ifndef LEDNBUTTON_H_INCLUDED
```

```
153 #define LEDNBUTTON_H_INCLUDED
154
155 void ledSet(int led_p);
156
157 void ledClr(int led_p);
158
159 void ledTog(int led_p);
160
161 int getButtonSta(int btn_p);
162
163 int getButtonPrs(int btn_p);
164
165 int adc_read(void);
166
167 void releClr(int rele_p);
168
169 void releSet(int rele_p);
170
171 #endif // LEDNBUTTON_H_INCLUDED
172 utils.c
173 #include "utils.h"
174 #include "config.h"
175 #include "lcd.h"
176 #include "SSD1306.h"
177 #include "ledNbutton.h"
178
179 void configureLEDs(void) {
180     /// nastavimo LED kot izhode
181     DDRB |= (1<<LEDB) | (1<<LEDG) | (1<<LEDR) | (1<<LEDsys);
182 };
183
184 void configureLCD(void) {
```

```
185     /// prizgemo LCD
186     lcd_begin();
187     lcd_set_cursor(0, 0);
188     lcd_print("Hello World!");
189 }
190 
191 void coifigureButtons(void){
192     /// nastavimo tipke kot vhodi
193     DDRB &= ~(1<<BUTL) & ~(1<<BUTR);
194 }
195 
196 void cofigureWaterLevel(void) {
197     /// nastavimo refernco Vref=AVcc
198     ADMUX |= (1<<REFS0);
199     /// nastavimo prescaler na 128 in prizgemo ADC
200     ADCSRA |= (1<<ADPS2)|(1<<ADPS1)|(1<<ADPS0)|(1<<ADEN);
201 };
202 
203 void configureReles(void) {
204     /// nastavimo releje kot izhode
205     DDRC |= (1<<REL1) | (1<<REL2) | (1<<REL3);
206     /// nastavimo jih kot zaprto, da ne porabljajo energije
207     releSet(REL1);
208     releSet(REL2);
209     releSet(REL3);
210 }
211 
212 void configureOLED(void) {
213     /// pozenemo OLED
214     OLED_Init();
215 }
216 utils.h
```

```
217 #ifndef UTILS_H_INCLUDED
218 #define UTILS_H_INCLUDED
219
220 #include "config.h"
221
222 // configuration functions
223 void configureLEDs(void);
224
225 void configureLCD(void);
226
227 void coifigureButtons(void);
228
229 void cofigureWaterLevel(void);
230
231 void configureReles(void);
232
233 void configureOLED(void);
234
235 #endif // UTILS_H_INCLUDED
236 lcd.h
237 /*
238  * lcd.h
239  *
240  * Created: 2/20/2020 1:17:15 AM
241  *  Author: Shuvangkar
242  */
243
244
245 #ifndef LCD_H_
246 #define LCD_H_
247 #define F_CPU 16000000UL
248
```

```
249 #include <avr/io.h>
250 #include "util/delay.h"
251
252 /*************************************/
253 /*                LCD Connection          */
254 /*************************************/
255 #define RS_PORT                 PORTD
256 #define RS_PIN          PD2
257
258 #define RW_PORT                 PORTD
259 #define RW_PIN                  PD1
260
261 #define EN_PORT                 PORTD
262 #define EN_PIN          PD0
263
264 #define D4_PORT                 PORTD
265 #define D4_PIN          PD4
266
267 #define D5_PORT                 PORTD
268 #define D5_PIN          PD5
269
270 #define D6_PORT                 PORTD
271 #define D6_PIN          PD6
272
273 #define D7_PORT                 PORTD
274 #define D7_PIN          PD7
275
276 /*************************************/
277 /* LCD Library Internal Functions       */
278 /*************************************/
279 void lcd_port_init_4Bit();
280 void map_data_port(unsigned char data);
```

```
281 void lcd_write_nibble(unsigned char data, uint8_t type); //type->0= cmd, type
282 void lcd_write_cmd_with_delay(unsigned char data);
283 void lcd_write_byte(unsigned char data, uint8_t type);
284 void lcd_busy_wait();
285
286 /***************************************/
287 /*                  LCD Public Functions          */
288 /***************************************/
289
290 void lcd_begin();
291 void lcd_set_cursor(char x,char y);
292 void lcd_print(char* str);
293
294 /***************************************/
295 /*                  COMMON MACRO                */
296 /***************************************/
297 #define DDR(port)  (*(&port-1))
298 #define PIN(port) (*(&port - 2))
299
300 #define setBit(port, bit) (port) |= (1 << (bit))
301 #define clearBit(port, bit) (port) &= ~(1 << (bit))
302
303 #define WRITE_PIN(PORT,PIN,STATE) ((PORT) = ((PORT&(~(1<<PIN)))|(STATE<<PIN)))
304
305 /***************************************/
306 /*                  Utility MACRO               */
307 /***************************************/
308 #define RS_HI() setBit(RS_PORT,RS_PIN)
309 #define RS_LO() clearBit(RS_PORT,RS_PIN)
310
311 #define RW_HI() setBit(RW_PORT,RW_PIN)
312 #define RW_LO() clearBit(RW_PORT,RW_PIN)
```

```
313
314 #define EN_HI() setBit(EN_PORT,EN_PIN)
315 #define EN_LO() clearBit(EN_PORT,EN_PIN)
316
317
318 #define MAP_BIT(value,pos,port,pin)  (port) = (port&(~(1<<pin)))|
(((value>>pos)&&1)<<pin)
319
320 #define CMD 0
321 #define DATA 1
322 #endif /* LCD_H_ */
323
324 /*
325 #define DATA(value)
326
327 D4_PORT |= 1<<D4_PIN
328 value = 0b0000 0111
329
330 PORTB = 0b1011 0010
331 bit0  = 0b1110 1111
332 ---------------------
333               0b101
334
335 bit0 = (value>>0)&1
336 bit1 = (value>>1)&1
337 bit2 = (value>>2)&1
338 bit3 = (value>>3)&1
339
340 D4_PORT = (D4_PORT&(~(1<<D4_PIN)))|(bit0<<D4_PIN);
341 */
342 globals.h
343 #ifndef GLOBALS_H_INCLUDED
```

```
344 #define GLOBALS_H_INCLUDED
345
346 #include <stdint.h>
347
348 /// medfunkcijske spremenljivke za komunikacijo
349
350 extern uint8_t but; /// stanje tipk
351 extern uint8_t status; /// trenutni status
352 extern uint8_t level; /// nivo trenutne izbire
353 extern uint8_t sumlevel; /// skupni nivo
354 extern uint8_t statusFill; /// status polnjenja
355 extern uint16_t adc_level; /// level za adc varovalo
356 extern uint8_t flag; /// varovalo za uporabo tipk
357
358
359 #endif // GLOBALS_H_INCLUDED
360 task0.c
361 #include "Task0.h"
362 #include "utils.h"
363 #include "ledNbutton.h"
364 #include "globals.h"
365 #include "config.h"
366
367 uint8_t but;
368 uint8_t flag;
369
370 void Task0_LED_n_BUT(void)
371 {
372     /// pridobi pritisk tipk
373     but = getButtonPrs(BUTR);
374     but |= (getButtonPrs(BUTL) << 1);
375     flag = 1;
```

```
376
377     /// pobrišemo ledice
378     ledClr(LEDR);
379     ledClr(LEDB);
380     ledClr(LEDG);
381     ledTog(LEDsys);
382
383     if (status == 0) {
384         ledSet(LEDG); /// stanje pripravljenosti
385     } else if (status < 4) {
386         ledSet(LEDB); /// priprava na polnjenje
387     } else if (status == 4) {
388         ledSet(LEDG); /// polnjenje
389     } else {
390         ledSet(LEDR); /// napaka
391     }
392
393 }
394 task0.h
395 #ifndef TASK0_H_
396 #define TASK0_H_
397
398 #include <avr/io.h>
399
400 void Task0_LED_n_BUT(void);
401
402 #endif /* TASK0_H_ */
403 task1.c
404 #include "Task1.h"
405
406 #include "utils.h"
407 #include "ledNbutton.h"
```

```
408 #include "globals.h"
409 #include "lcd.h"
410 #include <stdio.h>
411
412
413 void Task1_LCD(void)
414 {
415     lcd_set_cursor(0,0);
416     lcd_print("                ");
417     lcd_set_cursor(0,1);
418     lcd_print("                ");
419
420     if (status == 0) {
421         lcd_set_cursor(0,0);
422         lcd_print("Za zac prit 2");
423     } else if (status == 1) {
424         lcd_set_cursor(0,0);
425         lcd_print("Za level 1 ->1");
426         lcd_set_cursor(0,1);
427         lcd_print("Za nad. ->2");
428     } else if (status == 2) {
429         lcd_set_cursor(0,0);
430         lcd_print("Za level 2 ->1");
431         lcd_set_cursor(0,1);
432         lcd_print("Za nad. ->2");
433     } else if (status == 3) {
434         lcd_set_cursor(0,0);
435         lcd_print("Za level 3 ->1");
436         lcd_set_cursor(0,1);
437         lcd_print("Za nad. ->2");
438     } else if (status == 4) {
439         lcd_set_cursor(0,0);
```

```
440         lcd_print("                ");
441         lcd_set_cursor(0,0);
442         lcd_print("Polnjenje...");
443     } else if (status == 5) {
444         lcd_set_cursor(0,0);
445         lcd_print("Izliv tekocine!!");
446     } else {
447         lcd_set_cursor(0,0);
448         lcd_print("NAPAKA!");
449     }
450 }
451 task1.h
452 #ifndef TASK1_H_
453 #define TASK1_H_
454
455 #include <avr/io.h>
456
457 void Task1_LCD(void);
458
459 #endif /* TASK1_H_ */
460 task2.c
461 #include "Task2.h"
462
463 #include "utils.h"
464 #include "ledNbutton.h"
465 #include "globals.h"
466 #include "SSD1306.h"
467
468 void Task2_OLED(void)
469 {
470     /// pobrisi oled
471     if (status == 0) {
```

```
472         OLED_Clear();
473     } else if (status < 4) {
474         /// prikazi grafe s stanji polnjenja
475         OLED_VerticalGraph(status - 1, level);
476         OLED_VerticalGraph(3, sumlevel);
477     }
478 
479 }
480 task2.h
481 #ifndef TASK2_H_
482 #define TASK2_H_
483 
484 #include <avr/io.h>
485 
486 void Task2_OLED(void);
487 
488 #endif /* TASK2_H_ */
489 task3.c
490 #include "Task3.h"
491 
492 #include "utils.h"
493 #include "ledNbutton.h"
494 #include "globals.h"
495 
496 void Task3_Rele(void)
497 {
498     /// v stanju polnjenja nastavimo ustrezne rele
499     if (status == 4) {
500 
501         if (statusFill == 0) {
502             releClr(REL1);
503             releSet(REL2);
```

```
504             releSet(REL3);
505         } else if (statusFill == 1) {
506             releSet(REL1);
507             releClr(REL2);
508             releSet(REL3);
509         } else if (statusFill == 2) {
510             releSet(REL1);
511             releSet(REL2);
512             releClr(REL3);
513         } else {
514             releSet(REL1);
515             releSet(REL2);
516             releSet(REL3);
517         }
518     } else  {
519         releSet(REL1);
520         releSet(REL2);
521         releSet(REL3);
522     }
523 }
524 task3.h
525 #ifndef TASK3_H_
526 #define TASK3_H_
527
528 #include <avr/io.h>
529
530 void Task3_Rele(void);
531
532 #endif /* TASK2_H_ */
533 task4.c
534 #include "Task4.h"
535
```

```
#include "utils.h"
#include "ledNbutton.h"
#include "globals.h"

uint16_t adc_level;

void Task4_adc(void)
{
    adc_level = adc_read(); /// samo preberemo adc,    e to dovolj traja
}
task4.h
#ifndef TASK4_H_
#define TASK4_H_

#include <avr/io.h>

void Task4_adc(void);

#endif /* TASK2_H_ */
task5.c
#include "Task5.h"

#include "utils.h"
#include "ledNbutton.h"
#include "globals.h"

/// statusne spremenljivke
uint8_t status = 0;
uint8_t statusFill = 0;
uint8_t level;
uint8_t sumlevel;

```

```
568 /// parametri polnjenja posameznih relejev
569 uint8_t lv1 = 0;
570 uint8_t lv2 = 0;
571 uint8_t lv3 = 0;
572
573 uint16_t time = 0;
574 uint8_t k = 75; /// koeficient časa glede na tick speed (skupaj traja 90 s polnjenje
575
576 void Task5_statusMaker(void)
577 {
578   if ((status == 0)&&(flag == 1)) {
579     /// prestavimo na naslednji status
580     if (but & (1<<BUTR)) {
581       status ++;
582       flag = 0;
583       sumlevel = 0;
584     }
585   } else if ((status == 1)&&(flag == 1)) {
586     /// povečamo level
587     if (but & (1<<BUTL)) {
588       if (level < 100) level += 10;
589       flag = 0;
590     }
591
592     if (but & (1<<BUTR)) {
593       /// nastavimo vse statusne spremenljivke pred premikom
594       status ++;
595       lv1 = level/10;
596       sumlevel = level;
597       level = 0;
598       flag = 0;
599     }
```

```
600     } else if ((status == 2)&&(flag == 1)) {
601         if (but & (1<<BUTL)) {
602             /// povačamo level do skupnega levela max 100
603             if ((sumlevel + level) < 100) {
604                 level += 10;
605             }
606             flag = 0;
607         }
608 
609         if (but & (1<<BUTR)) {
610             /// nastavimo vse statusne spremenljivke pred premikom
611             status++;
612             sumlevel += level;
613             lv2 = level/10;
614             level = 0;
615             flag = 0;
616         }
617     } else if ((status == 3)&&(flag == 1)) {
618         /// enako kot pri 2
619         if (but & (1<<BUTL)) {
620             if ((sumlevel + level + 10) < 101) {
621                 level += 10;
622             }
623             flag = 0;
624         }
625 
626         if (but & (1<<BUTR)) {
627             status++;
628             if (lv1 == 0) statusFill++;
629             sumlevel += level;
630             lv3 = level/10;
631             level = 0;
```

```
632             flag = 0;
633         }
634         //flag = 0;
635     } else if (status == 4) {
636         /// stanje polnjenja
637         /// za vsak rele posebej polnimo zadevo in preverjamo, ce pretece dovolj casa
638         if (statusFill == 0) {
639             if (time >= (k*lv1)) {
640                 statusFill++;
641                 if (lv2 == 0) statusFill++;
642                 if (lv3 == 0) statusFill++;
643                 time = 0;
644             }
645         } else if (statusFill == 1) {
646             if (time >= (k*lv2)) {
647                 statusFill++;
648                 if (lv3 == 0) statusFill++;
649                 time = 0;
650             }
651         } else if (statusFill == 2) {
652             if (time >= (k*lv3)) {
653                 statusFill++;
654                 time = 0;
655             }
656         } else {
657             /// pobrisemo parametre in resetiramo stanje
658             statusFill = 0;
659             status = 0;
660             time = 0;
661         }
662         time++;
663     }
```

```
664
665     /// varovalo
666     if (adc_level > 150) {
667         status = 5;
668     }
669 }
```