Microcontrollers, Jyri Määttä, 50248676

Ajastin tehtävä

Kuvaus toiminnasta

Ohessa on suunniteltu laite, joka hälyttää käyttäjän asettaman minuuttimäärän kuluttua. Siinä on kolme näppäintä käyttäjää varten. “Add minute” näppäimellä voi valita, että kuinka monen minuutin päästä halutaan summerin soivan. “Start” näppäimellä käynnistetään ajastin. Jotta käyttäjä on tietoinen siitä, että ajastin on päällä, on sitä varten LED-valo, joka on käynnissä aina kun ajastin on käynnissä. Kolmas näppäin on “Reset” näppäin, joka asettaa ajastimen alkutilaan sekä sammuttaa summerin äänen, jos se on alkanut soimaan ajan loputtua.

Kuvaus piirikaaviosta

Proteus kirjastoistani ei löytynyt näppäimiä tai lediä, joilla olisi myös PCB malli, joten piirikaaviossani on käytetty muita komponentteja mallintamaan niitä. Komponenttien otsikoista tunnistaa, mikä komponentti on kyseessä.
“Start” ja “Reset” näppäimet on valittu PD2 ja PD3 pinneihin, jotta ne voivat laukaista keskeytyksen ja joko käynnistää ajastimen tai sammuttaa sen. Summeri on yhdistetty PB3, jotta sille voi lähettää PWM kanttiaaltoja sopivalla taajudella.

Kuvaus PCB-asettelusta

Piirilevylle komponentit on aseteltu lähelle mikrokontrolleria välttääkseemme turhaa pitkiä etäisyyksiä komponenttien välillä. Näppäimet on myös aseteltu vierekkäin, jotta niiden käyttäminen olisi helpompaa.

Kuvaus ohjelmasta

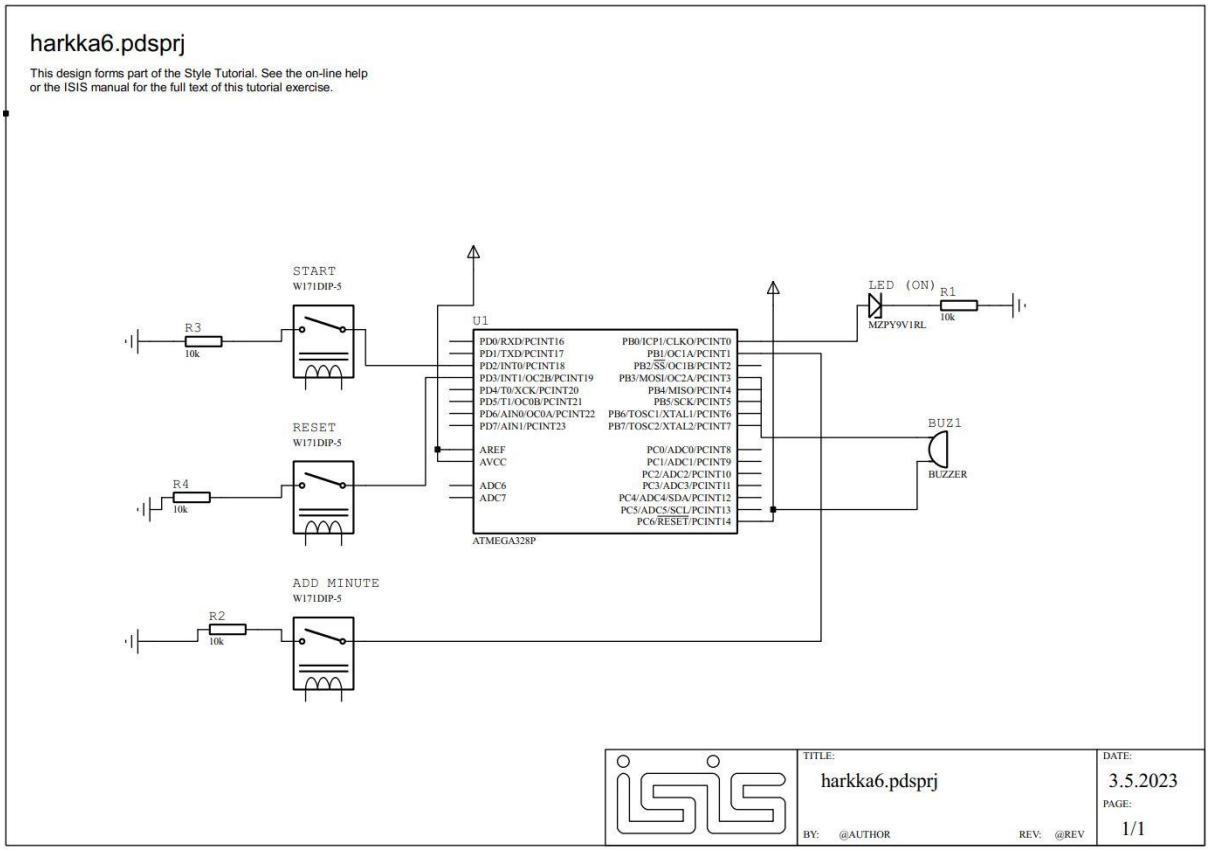
Ohjelman aluksi kutsutaan init()- funktiota, joka alustaa tarvittavat pinnit ja keskeytykset. Sen jälkeen ikuisessa luupissa pyörii loop()- funktio, joka sisältää varsinaisen toiminnallisuuden. Ohjelma on virtaa säästääkseen ns “sleep-mode”:ssa aina kun ajastinta ei käytetä. Kun “Start” näppäintä painetaan, käynnistyy keskeytys, joka saa ajastimen käynnistymään. Kun aika on loppunut, alkaa ohjelma soittamaan summeria. Summerin saa sammutettua “Reset” näppäimestä, joka asettaa ajastimen taas “Sleep-mode”:een. Ohjelman voi sammuttaa myös kesken ajastimen, jos käyttäjä niin haluaa. Aina kun ajastin on päällä, myös ledi on päällä. Ajastimen loputtua se sammuu.
Ohjelmaa on kommentoitu itse koodissa lisää (koodi löytyy alempaa).

Puutteita toteutuksessa

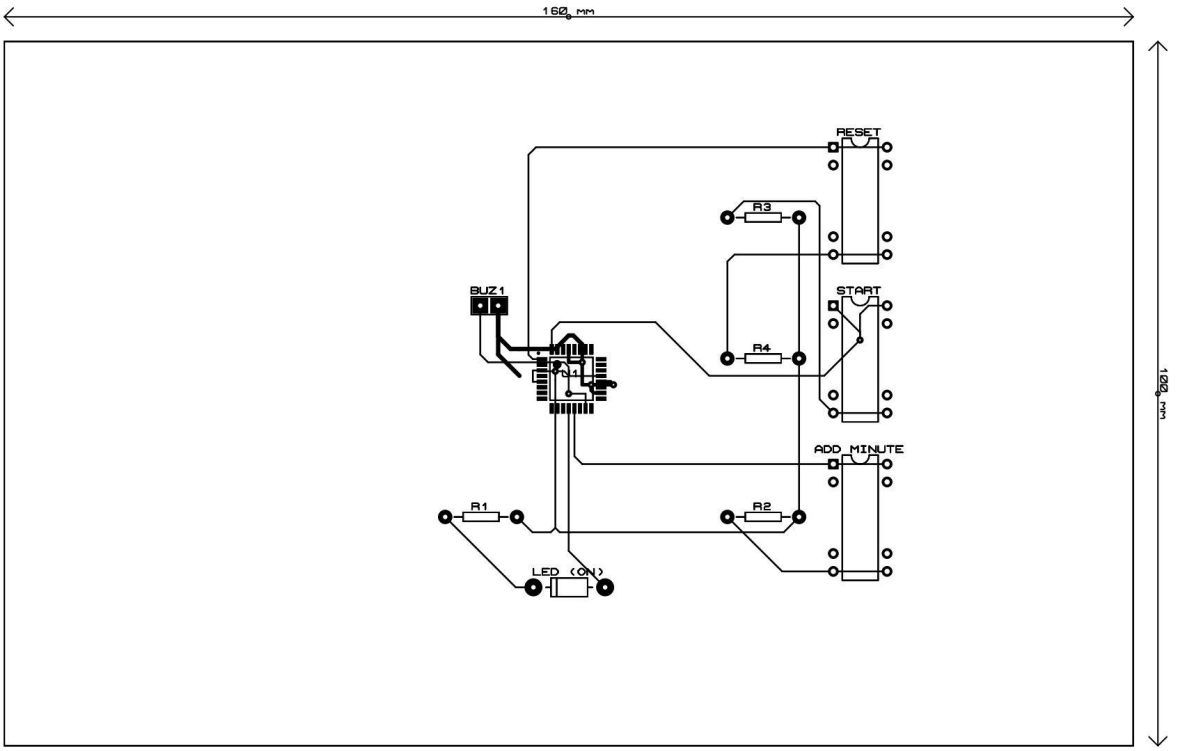
Lyhyen ajan takia toteutus ei ole mitenkään täydellinen. Jos haluttaisiin parantaa toteutusta, olisi siihen hyvä lisätä vielä esimerkiksi nappi sekunteja varten tai nappi joka vähentää aikaa. Tällä hetkellä aikaa voi vain lisätä. Myöskään käyttäjä ei tiedä montako minuuttia hän on aikaa lisännyt ellei itse laske omia napautuksiaan. Käyttäjä ei myöskään tiedä montako minuuttia on jäljellä ajasta, kun ajastin on päällä. Näitä varten jatkokehityksessä olisi oleellista lisätä LCD näyttö, joka päivittää ruudulle valitun ajan sekä näyttää montako minuuttia on aikaa jäljellä. Myöskin kun “Add minute” näppäintä on tarkoitus käyttää, taitaa ohjelma olla edelleen sleep-modessa, joten se ei välttämättä saa tallennettua käyttäjän napautuksia. Ohjelmaa tulisi ehkä muokata niin, että sleep-modesta herätään “Add minute” näppäimen napautuksesta. Rajallisen ajan takia ongelmaa ei ole kuitenkaan keretty harkita sen enempää.

Liitteet

Piirikaavio



PCB-asettelu



Ohjelmakoodi

```
/*
 * exam.c
 *
 * Created: 3.5.2023 13.42.47
 * Author : jyri_
 */
#include <avr/io.h>
#include <util/delay.h>
#include <avr/interrupt.h>
#include <avr/sleep.h>


#define BUZZER_PIN PIND3
#define LED_PIN PINB0
#define ADD_BUTTON_PIN PINB1
#define START_BUTTON_PIN PIND2
#define RESET_BUTTON_PIN PIND3

bool timer_on = false;
int minutes;

void init(){
```

```
      // Function to initialize the buzzer and LED pins as outputs
      // and button pins as inputs. Also enables interrupts to call
      // reset_timer interrupt to shut off the buzzer.
}

void counting_time(int minutes){

      // Function that delays the program with the given minutes value.
}

reset_timer(INT1_vect){

      // Interrupt handler that ends the timing and shuts the buzzer and
the LED off
      // when it receives reset signal.

      timer_on = false;

}

start_timer(INT0_vect){

      // Interrupt handler that starts the timer loop by setting the
timer_on to true
      // when the start button has been pressed

      timer_on = true;

}

void buzzer_on (){

      // Uses the PWM pin to send proper square waves to the buzzer to
make sound.

}

int count_minutes(){

      // Counts how many times the user has pushed the "Add minute"
button.

      return minutes
}
```

```
void LED_off (){
      // Turns off the LED light to show that the timing is off
}

void LED_on (){
      // Turns on the LED light to show that the timing is on
}

void loop(){
      // The main loop of the program
      if(timer_on){

            // The start button has been clicked, starting the timer
            LED_on();

            while(timer_on){
                  counting_time(count_minutes());
                  buzzer_on();
                  // Buzzing stops when reset button is pushed and
                  // timer_on becomes false.
            }
            LED_off();
            minutes = 0;

      } else{
            // The timer is off, entering sleep mode
            set_sleep_mode(SLEEP_MODE_PWR_DOWN); // set the sleep-mode
            sleep_enable(); // set SE-bit
            sleep_cpu(); // SLEEP-instruction
            sleep_disable(); // reset SE-bit
      }

}

int main(void)
{
      init();

    while (1)
    {
            loop();
    }
}
```