Tampereen ammattikorkeakoulu

Harjoitustyö - Musiikkisoitin

Sulautetun järjestelmän ohjelmointi ja mikrokontrollerit

Eino Palomäki
Juho Tiainen
Carita Lindfors

Ratakanat

HARJOITUSTYÖ
Joulukuu 2023

Tietotekniikan tutkinto-ohjelma
22TIETOB

TIIVISTELMÄ

Tampereen ammattikorkeakoulu
Tietotekniikan tutkinto-ohjelma
22TIETOB

PALOMÄKI, EINO & TIAINEN, JUHO & LINDFORS, CARITA:
Harjoitustyö - Musiikkisoitin

Harjoitustyö 25 sivua, joista lähdekoodia 10 sivua
Joulukuu 2023

Tässä raportissa esitellään Arduino Nano -mikro-ohjaimen avulla toteutettu musiikkisoitin, joka toimii Theremin-soittimen tavoin. Työssä on käytetty DFRobotin 37 Pcs Sensor Set -kittiin sisältyviä laitteita. Tavoitteena oli saavuttaa arvosana 5. Raportissa käydään läpi työn suunnittelu, toteutus, testaus, oppimiskokemukset sekä mahdolliset ongelmat.

Asiasanat: tiivistelmä

SISÄLLYS

1 JOHDANTO 4
1.1 Thermin 4
1.2 Harjoitustyön määrittely ja tavoitteet 5
2 TEKEMISEN SUUNNITTELU 6
2.1 Suunnitelman kriteerit 6
2.2 Toteutus 8
2.2.1 Toteutuksessa käytetyt komponentit: 8
2.3 Ohjelman toiminta 9
2.3.1 Pääpiirteet 9
3 TESTAUS 13
3.1 Työn toiminta 13
4 OSAAMISEN REFLEKTOINTI 14
4.1 Mitä opimme? 14
4.1.1 Toimiko työ suunnitellusti? 14
4.1.2 Arvosana 14
LIITTEET 15
Lähdekoodit 15

1 JOHDANTO

1.1 Thermin

Theremin on sähköinen musiikkisoitin, joka tunnetaan siitä, että sitä soitetaan ilman fyysistä kosketusta. Theremin-soittimen toimintaperiaate perustuu sähkömagneettiseen kenttään. Se koostuu yleensä kahdesta antennista: pystysuuntaisesta antennista, jota kutsutaan anturiantenniksi, ja vaakasuuntaisesta antennista, joka toimii äänen korkeuden (pitch) säätönä. Soittaja sijoittaa kätensä näiden antennien lähelle, mutta ei kosketa niitä. Etäisyyden muutokset käden ja antennien välillä vaikuttavat sähkömagneettiseen kenttään, mikä puolestaan vaikuttaa äänisignaaliin.

Thereminin äänentuotto perustuu osittain heterodyyniperiaatteeseen, joka luo kaksi eri taajuista oskillaattoria. Yksi oskillaattori on kiinteä, kun taas toinen muuttuu etäisyyden mukaan. Nämä kaksi oskillaattoria sekoitetaan, ja syntynyt eroaaltosignaali määrittää lopullisen äänen korkeuden.

Kuva 1. Moog Etherwave Theremin

1.2 Harjoitustyön määrittely ja tavoitteet

Harjoitustyömme tavoitteena on toteuttaa Arduino Nanolla DFRobotin 37 Pcs Sensor Set -kitin sisältämillä laitteilla toimiva musiikkisoitin, joka simuloi theremin-soittimen toimintaa. Työn toteutuksessa huomioimme erityisesti innovatiiviset ratkaisut, monipuolisuuden, ja huolellisesti tehdyn dokumentaation.

Tavoitteenamme on saavuttaa arvosana 5 seuraavilla kriteereillä:

- Käytetty neljää mikro-ohjaimen sisäistä lohkoa.
- Liitetty jokin toimiva I/O-laite mikro-ohjaimeen.
- Käytetty kolmea keskeytyspalvelua.
- IO-portteja ohjattu rekistereiden avulla.
- Käyttöliittymä on looginen ja helppokäyttöinen.
- Dokumentaatio on kattava ja yksityiskohtainen.
- Koodi on laadittu helppolukuiseksi selkeällä kommentoinnilla ja rakenteella.

2 TEKEMISEN SUUNNITTELU

2.1 Suunnitelman kriteerit

1. Käytetään vähintään neljää erilaista sisäistä lohkoa.
    - 16bit T/C 1:
        - 16-bittinen ajastin/laskuri tarjoaa mahdollisuuden ajoittaa tapahtumia tietyn ajanjakson perusteella. Koodissa tämä ilmenee puolen tunnin musiikinsoiton automaattikeskeytyksenä.

    - TWI (Two-Wire Interface):
        - TWI on käytössä kommunikaatiossa I2C-protokollan avulla. Käytetään liittämään lisälaitteita, kuten LCD-näyttö.

    - A/D-muunnin (Analog-to-Digital Converter, ADC):
        - ADC mahdollistaa analogisten signaalien muuttamisen digitaalisiksi, kuten potentiometrin lukeminen. Tämä on koodin osa, joka lukee potentiometrin arvon.

    - Watchdog timer:
        - Watchdog-timer auttaa välttämään ohjelman kaatumista ja pitää sen toiminnassa. Koodissa sen on käytössä wdt_enable(WDTO_8S); rivillä, joka käynnistää vartijakellon kahdeksan sekunnin ajastimella. Watchdog-timer seuraa mikrokontrollerin toimintaa ja keskeyttää sen tarvittaessa, jottei ohjelma jumitu.

2. Luodaan 2-suuntainen käyttöliittymä.
    - 2-suuntainen käyttöliittymä mahdollistaa vuorovaikutuksen käyttäjän ja laitteen välillä kahdella suunnalla. Koodissa tämä näkyy käyttäjän valitessa sävelen ja oktaavin näppäimistöllä ja samalla näytöllä annetaan käyttäjälle ohjeita ja tietoa.

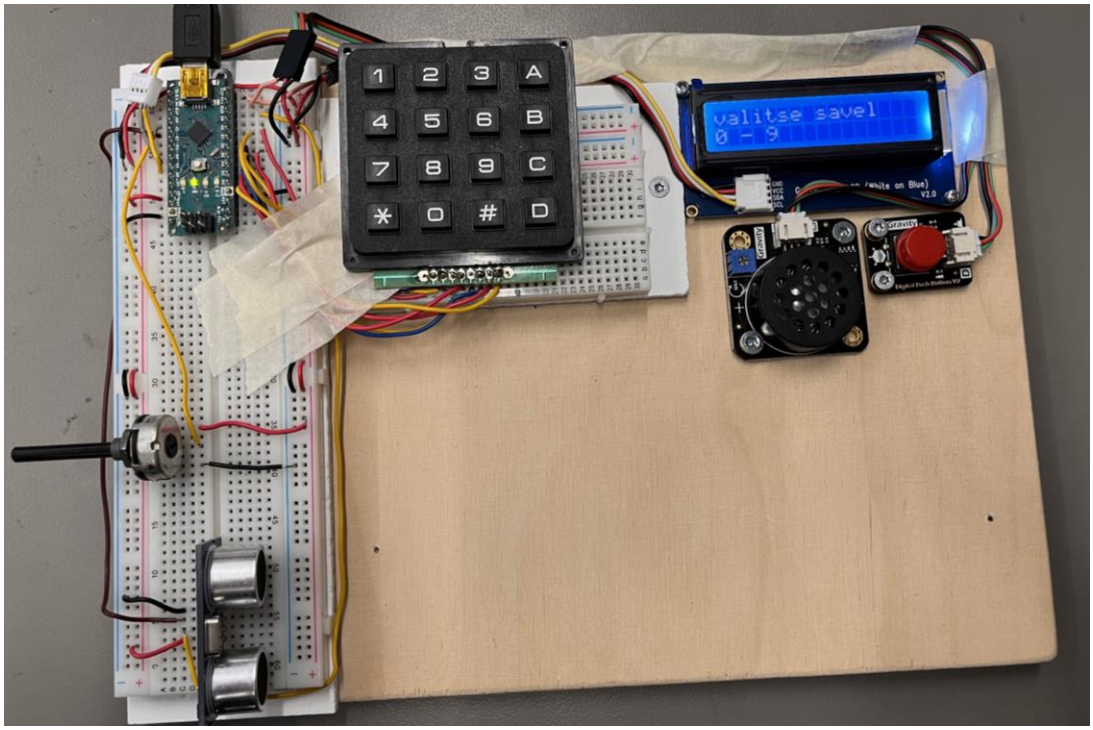
3. Käytetään keskeytyspalvelua ja ohjataan IO-portteja rekistereiden avulla.
    - Koodissa käytetään keskeytyspalvelua nappulan painalluksen havaitsemiseen, ja IO-portteja ohjataan rekistereiden avulla esimerkiksi ajastimien ja ultraäänisensorien ohjauksessa.

4. Huolellinen dokumentaatio ajoissa palautettuna.
    - Kaikki tarvittavat dokumentit liittyen projektiin (koodin kommentointi, käyttöohjeet jne.) on palautettu ajoissa ja ne ovat kattavia ja ymmärrettäviä. Koodi on selkeästi kommentoitu.

2.2 Toteutus

2.2.1 Toteutuksessa käytetyt komponentit:

- Arduino Nano
- Grove LCD (White on Blue) -näyttö
- Näppäimistö
- Potentiometri
- Ultra Sonic Sensor HC-SR04 (etäisyysanturit)
- Digital Speaker Module FIT0449 (kaiutin)
- Digital Push Button (red) DFR0U29-R (nappi)
- Tarvittavat johdot



2.3 Ohjelman toiminta

Tämä Arduino-koodi toteuttaa musiikkisoittimen, joka toimii Theremin-soittimen tavoin. Tässä ohjelmassa Theremin-soitin käyttää etäisyyssensoria mittaamaan käden tai muun objektin etäisyyttä laitteesta, ja sen perusteella soitetaan eri säveliä. Nuotin kestoa voidaan säätää potentiometrillä. Maksimietäisyys sensorista on 20 cm, joka on alustan mitta.

Koodi on organisoitu setup- ja loop-funktioihin. Setup-funktiossa alustetaan tarvittavat komponentit, kun taas loop-funktio sisältää pääasiallisen ohjelman silmukan.

2.3.1 Pääpiirteet

1. Nuottien määrittely:
    - Koodi alkaa määrittelemällä nuottien taajuudet ja sointuluettelon eri duuri- ja molliasteikkoihin. Tämä tehdään kovakoodaamalla nuottien taajuudet.

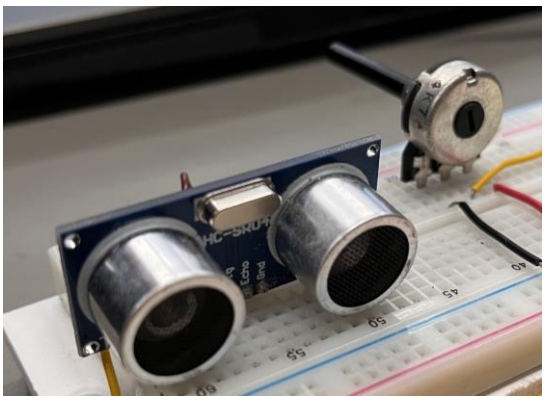
```
//määritellään kaikki tarvittavat nuotit ja niiden taajuus.
  #define A2  110
  #define AS2 117
  #define B2  123
  #define C3  131
  #define CS3 139
  #define D3  147
  #define DS3 156
  #define E3  165
  #define F3  175
  #define FS3 185
  #define G3  196
  #define GS3 208
  #define A3  220
  #define AS3 233
  #define B3  247
  #define C4  262
```

```
int notes[][7] =      //int array jossa määritetty sävellajit ja oktaavit
{ // Ensimmäinen oktaavi
  {A2, B2, CS3, D3, E3, FS3, GS3}, //a-duuri
  {A2, B2, C3, D3, E3, F3, G3}, // a-molli
  {B2, CS3, DS3, E3, F3, GS3, AS3}, // h-duuri
  {B2, CS3, D3, E3, FS3, G3, A3}, // h-molli
  {C3, D3, E3, F3, G3, A3, B3}, // c-duuri
  {C3, DS3, F3, G3, GS3, AS3, B3}, // c-molli
  {D3, E3, FS3, G3, A3, B3, CS4}, // d-duuri
  {D3, E3, F3, G3, A3, AS3, C4}, // d-molli
  {E3, FS3, GS3, A3, B3, CS4, DS4}, //e-duuri
  {E3, FS3, G3, A3, B3, C4, D4}, // e-molli

```

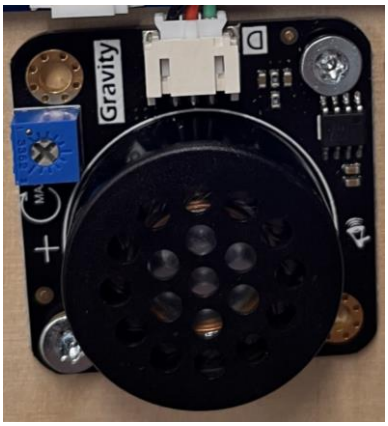
2. Näppäimistö:
    - Ohjelma käyttää neljän rivin ja neljän sarakkeen näppäimistöä (4x4-näppäimistö), joka on kytketty Arduinoon. Näppäimistön avulla käyttäjä voi valita soitettavan sävelen ja oktaavin.

3. Ultraäänisensori:
    - Koodi käyttää ultraäänisensoria (HC-SR04) mittaamaan käden tai muun objektin etäisyyttä. Näiden antamat tiedot käytetään säveltämään musiikkia.

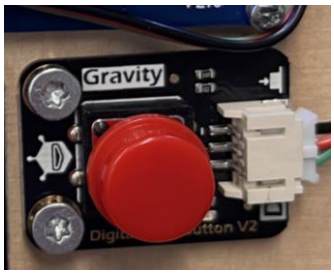


4. LCD-näyttö:
    - Koodi ohjaa LCD-näyttöä (Grove LCD), joka näyttää käyttäjälle ohjeet siitä, mitä tehdä. Näyttöön tulee viestiä, kuten "Valitse savel" ja "Valitse oktaavi" sekä lopuksi kertoo mitä säveltä ja oktaavia käytetään.

5. Äänen soitto:
   - Koodi käyttää Arduino-nanoa ja kaiutinta (Digital Speaker Module FIT0449) soittamaan valittua säveltä.



6. Keskeytykset ja vahtikoira:
   - Koodi käyttää keskeytystä nappulan painalluksen havaitsemiseen. Lisäksi siinä on vartijakello (Watchdog Timer), joka varmistaa, että ohjelma pysyy vakiona ja ei jää jumiin.




```
188     wdt_enable(WDTO_8S);    // alustetaan watchdog
189   }
```

```
// 16bit T/C 1 (ajastin/laskuri)
TCCR1A = 0;
TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10);
OCR1A = 15624;
TIMSK1 = (1 << OCIE1A);
```

```
// tarkistetaan jos käyttäjä on tehnyt nappi keskeytyksen ja onko aikakeskeytys täynnä
if (digitalRead(interruptPin) == HIGH ||  timerCounter >= 1800)
{
```

```
// Kutsutaan kun ylivuoto tapahtuu
ISR(TIMER1_COMPA_vect)
{
  timerCounter++; // inkrementoituu joka sekunti
}
```

7. Potentiometri:
   - Koodi käyttää potentiometriä (A0) määrittämään nuotin kestoa (pituutta).

3 TESTAUS

3.1 Työn toiminta

1. Valittu ääni-ala/oktaavi näppäimistöllä.
    - Koodissa on toteutettu näppäimistöllä tapahtuva valinta sävelestä ja oktaavista. Käyttäjälle esitetään ohjeet LCD-näytöllä, ja valitut äänet tallennetaan muuttujiin key ja key2. Ensimmäinen näppäimistövalinta liittyy sävelen valintaan (0-9), ja toinen liittyy oktaavin valintaan (A-D). Valinnan jälkeen käyttäjälle näytetään valitut äänet LCD-näytöllä.

2. Musiikin toistuminen etäisyyden perusteella.
    - Loopissa ohjelma tarkkailee etäisyyden muutoksia ultraäänisensorin avulla. Etäisyys muunnetaan nuotin korkeudeksi, ja sen perusteella soitetaan ääniä valitusta äänialasta ja oktaavista. Nuotin kesto määräytyy myös analogisen potentiometrin avulla, joten musiikin toistossa otetaan huomioon myös dynaaminen kesto.

3. Etäisyyden mittausten perusteella säädetyt äänen korkeus ja volyymi.
    - Funktio tone_to_play huolehtii etäisyyden perusteella valittujen nuottien soittamisesta ja note_duration-funktio huolehtii nuotin kestosta. Etäisyyden perusteella valitaan oikea nuotti ja säädellään sen korkeutta. Kesto taas määräytyy potentiometristä, ja näin saadaan aikaan dynaaminen äänenvoimakkuus.

4. Musiikin keskeytys napista.
    - Koodissa on toteutettu musiikin keskeytys painikkeella. Jos tietty pinni (määritelty interruptPin-muuttujassa) saa HIGH-signaalin musiikki keskeytetään. Tämä tarjoaa käyttäjälle mahdollisuuden pysäyttää musiikin tarvittaessa.

4 OSAAMISEN REFLEKTOINTI

4.1 Mitä opimme?

Työ on tarjonnut mahdollisuuden syventyä mikro-ohjaimien sisäisten lohkojen käyttöön, keskeytyspalveluihin, ultraäänisensorin toimintaan ja käyttöliittymän suunnitteluun. Työssä päästiin hyödyntämään laajasti kurssilla opittuja aiheita.

4.1.1 Toimiko työ suunnitellusti?

Työ on toiminut suunnitellusti useimmilta osin. Ongelmia ilmeni esimerkiksi keskeytyspalveluiden hallinnassa tai käyttöliittymän logiikassa.
Jatkoa varten saimme hyvää oppia siitä, että harjoitus olisi ollut hyvä aloittaa laajemmalla suunnittelulla, joka olisi dokumentoitu.

4.1.2 Arvosana

Tavoitteemme arvosanalle harjoitustyöstä on 5 ja koemme tämän täyttyvän seuraavilla perusteilla:

- Työssä käytetään omaperäistä ja innovatiivista ratkaisua, mikä erottaa sen massasta.
- Siinä hyödynnetään vähintään neljää sisäistä lohkoa, mikä osoittaa syvällistä ymmärrystä Arduino Nanon mahdollisuuksista ja monipuolisesta käytöstä.
- Koodi on rakenteeltaan selkeä ja tarkkaan kommentoitu.
- Siinä on hyödynnetty kolmea erilaista keskeytyspalvelua.
- Työ on kokonaisuudessaan erittäin monipuolisesti ja tarkasti tehty.

LIITTEET

Lähdekoodi

```
#include <Keypad.h>
#include <Wire.h>
#include <avr/wdt.h>
#include "rgb_lcd.h"

//määritellään kaikki tarvittavat nuotit ja niiden taajuus.
 #define A2  110
 #define AS2 117
 #define B2  123
 #define C3  131
 #define CS3 139
 #define D3  147
 #define DS3 156
 #define E3  165
 #define F3  175
 #define FS3 185
 #define G3  196
 #define GS3 208
 #define A3  220
 #define AS3 233
 #define B3  247
 #define C4  262
 #define CS4 277
 #define D4  294
 #define DS4 311
 #define E4  330
 #define F4  349
 #define FS4 370
 #define G4  392
 #define GS4 415
 #define A4  440
```

```
#define AS4 466
#define B4  494
#define C5  523
#define CS5 554
#define D5  587
#define DS5 622
#define E5  659
#define F5  698
#define FS5 740
#define G5  784
#define GS5 831
#define A5  880
#define AS5 932
#define B5  988
#define C6  1047
#define CS6 1109
#define D6  1175
#define DS6 1245
#define E6  1319
#define F6  1397
#define FS6 1480
#define G6  1568
#define GS6 1661
#define A6  1760
#define AS6 1865
#define B6  1976
#define C7  2093
#define CS7 2217
#define D7  2349
#define DS7 2489
#define E7  2637
#define F7  2794
#define FS7 2960
#define G7  3136
#define GS7 3322
```

```
  #define A7  3520
  #define AS7 3729
  #define B7  3951

int notes[][7] =      //int array jossa määritetty sävellajit ja oktaavit
{ // Ensimmäinen oktaavi
  {A2, B2, CS3, D3, E3, FS3, GS3}, //a-duuri
  {A2, B2, C3, D3, E3, F3, G3}, // a-molli
  {B2, CS3, DS3, E3, F3, GS3, AS3}, // h-duuri
  {B2, CS3, D3, E3, FS3, G3, A3}, // h-molli
  {C3, D3, E3, F3, G3, A3, B3}, // c-duuri
  {C3, DS3, F3, G3, GS3, AS3, B3}, // c-molli
  {D3, E3, FS3, G3, A3, B3, CS4}, // d-duuri
  {D3, E3, F3, G3, A3, AS3, C4}, // d-molli
  {E3, FS3, GS3, A3, B3, CS4, DS4}, //e-duuri
  {E3, FS3, G3, A3, B3, C4, D4}, // e-molli

// Toinen oktaavi
  {A3, B3, CS4, D4, E4, FS4, GS4}, //a-duuri
  {A3, B3, C4, D4, E4, F4, G4}, // a-molli
  {B3, CS4, DS4, E4, F4, GS4, AS4}, // h-duuri
  {B3, CS4, D4, E4, FS4, G4, A4}, // h-molli
  {C4, D4, E4, F4, G4, A4, B4}, // c-duuri
  {C4, DS4, F4, G4, GS4, AS4, B4}, // c-molli
  {D4, E4, FS4, G4, A4, B5, CS5}, // d-duuri
  {D4, E4, F4, G4, A5, AS5, C5}, // d-molli
  {E4, FS4, GS4, A4, B4, CS5, DS5}, //e-duuri
  {E4, FS4, G4, A4, B4, C5, D5}, // e-molli

// kolmas oktaavi
  {A4, B4, CS5, D5, E5, FS5, GS5}, //a-duuri
  {A4, B4, C5, D5, E5, F5, G5}, // a-molli
  {B4, CS5, DS5, E5, F5, GS5, AS5}, // h-duuri
  {B4, CS5, D5, E5, FS5, G5, A5}, // h-molli
  {C5, D5, E5, F5, G5, A5, B5}, // c-duuri
```

```
  {C5, DS5, F5, G5, GS5, AS5, B5}, // c-molli
  {D5, E5, FS5, G5, A5, B5, CS6}, // d-duuri
  {D5, E5, F5, G5, A5, AS5, C6}, // d-molli
  {E5, FS5, GS5, A5, B5, CS6, DS6}, //e-duuri
  {E5, FS5, G5, A5, B5, C6, D6}, // e-molli

// Neljäs oktaavi
  {A5, B5, CS6, D6, E6, FS6, GS6}, //a-duuri
  {A5, B5, C6, D6, E6, F6, G6}, // a-molli
  {B5, CS6, DS6, E6, F6, GS6, AS6}, // h-duuri
  {B5, CS6, D6, E6, FS6, G6, A6}, // h-molli
  {C6, D6, E6, F6, G6, A6, B6}, // c-duuri
  {C6, DS6, F6, G6, GS6, AS6, B6}, // c-molli
  {D6, E6, FS6, G6, A6, B6, CS7}, // d-duuri
  {D6, E6, F6, G6, A6, AS6, C7}, // d-molli
  {E6, FS6, GS6, A6, B6, CS7, DS7}, //e-duuri
  {E6, FS6, G6, A6, B6, C7, D7}, // e-molli

};

//näppäimistön määritys
const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3','A'},
  {'4','5','6','B'},
  {'7','8','9','C'},
  {'*','0','#','D'}
};

//näppäimistön pinnien määritys
byte rowPins[ROWS] = {5, 4, 3, 2};
byte colPins[COLS] = {9, 8, 7, 6};

//näppäimistön alustus
```

```
Keypad keypad = Keypad( makeKeymap(keys), rowPins, colPins, ROWS, COLS
);

// lcd näytön määritys
rgb_lcd lcd;
const int colorR = 255;
const int colorG = 0;
const int colorB = 0;

// käytettyjen pinnien määritys, ensimmäinen pinni on rekistereillä
const int trigPin = B00000100;
const int echoPin = 16;
const int interruptPin = 11;
const int potPin = A0;

// napin painalluksille varatut merkit
char key = NULL;
char key2 = NULL;

//käytetyt muuttujat, pituus ja aika
long cm, t;

// aika keskeytyksen counter
volatile uint16_t timerCounter = 0;



void setup()
{
  //LCD näytön alustus
  lcd.begin(16, 2);
  lcd.setRGB(colorR, colorG, colorB);

  // 16bit T/C 1 (ajastin/laskuri)
  TCCR1A = 0;
  TCCR1B = (1 << WGM12) | (1 << CS12) | (1 << CS10);
```

```
  OCR1A = 15624;
  TIMSK1 = (1 << OCIE1A);

  // määritetään pinniasiat
  DDRB = trigPin;
  pinMode(echoPin, INPUT);
  pinMode(potPin, INPUT);

   // Aseta ADEN päälle, otetaan käyttöön ADC
  ADCSRA |= (1 << ADEN);
  // Aseta jännitereferenssi AVcc:ksi
  ADMUX |= (1 << REFS0);
  // Tyhjennä ADLAR, jotta tulokset ovat oikein perustasolla
  ADMUX &= ~(1 << ADLAR);

  //hiljennetään kaiutin
  noTone(12);

  // alustetaan watchdog
  wdt_enable(WDTO_8S);
}

// Funktio näppäimistön syötteen ottoon.
void check_key()
{
  lcd.clear();
  lcd.print("valitse savel");
  lcd.setCursor(0, 2);
  lcd.print( "0 - 9");

  // odotetaan että nappia painetaan oikealla arvolla.
  while (!key)
  {
    key = keypad.getKey();
    // tarkistetaan, että vääriä nappeja ei ole käytetty
```

```
    if (key =='A' ||key =='B' ||key =='C' ||key =='D' || key == '*' || key == '#')
      key = NULL;
  }

  lcd.clear();
  lcd.print("Valitse oktaavi");
  lcd.setCursor(0, 2);
  lcd.print("A - D");
  // odotetaan että nappia painetaan oikealla arvolla.
  while (!key2)
  {
    key2 = keypad.getKey();
    // tarkistetaan, että vääriä nappeja ei ole käytetty
    if (key2 =='0' || key2 =='1' || key2 =='2' || key2 =='3' || key2 =='4' || key2 =='5' ||
key2 =='6' || key2 =='7' || key2 =='8' ||key2 =='9' ||  key2 == '*' || key2 == '#')
      key2 = NULL;
  }
  lcd.clear();
  lcd.print("Savel ");
  lcd.print(key);
  lcd.setCursor(0, 2);
  lcd.print("Oktaavi ");
  lcd.print(key2);
}

// Pääohjelma
void loop()
{
  //jos näppäimistöllä ei ole annettu vielä arvoja kutsutaan check_key funktiota
  if (!key2 && !key)
    check_key();

  // tarkistetaan jos käyttäjä on tehnyt nappi keskeytyksen ja onko aikakeskeytys
täynnä
  if (digitalRead(interruptPin) == HIGH ||  timerCounter >= 1800)
```

```
  {
    lcd.clear();
    noTone(12);
    delay(500);
    key = NULL;
    key2 = NULL;
    timerCounter = 0;
  }

  //rekistereillä laitetaan trigPinnin tila alas
  PORTB ^= trigPin;
  delayMicroseconds(4);
  //rekistereillä laitetaan trigPinnin tila ylös ja ultraääni sensori lähettää ultraääntä
  PORTB = trigPin;
  delayMicroseconds(10);
  //rekistereillä laitetaan trigPinnin tila alas
  PORTB ^= trigPin;

  // lasketaan kuinka kauan ultraäänellä kesti palauta sensorille
  t = pulseIn(echoPin, HIGH);

  //muutetaan mitattu aika senttimetreiksi
  cm = t / 29 / 2;

  //kutustaan funktiota, joka lukee potenttiometrin arvon, josta saadaan nuotin
kesto
  int dur = note_duration();

  // kutsutaan funktiota, joka soittaa nuottia
  tone_to_play(cm,(int)key -48, dur);

  // nollataan watchdog
  wdt_reset();

}
```

```
//funktio, jolla tehdään AD - muunnos ja saadaan potikan arvo
int readPotentiometer(int pin)
{
  // Aseta ADC-muunnos
  ADCSRA |= (1 << ADSC);  // Aloita muunnos

  // Odota muunnoksen valmistumista
  while (ADCSRA & (1 << ADSC))
  ;

  // Palauta luettu arvo
  return ADC;
}

int   note_duration()
{
  // Tyhjentää kolme alinta bittiä ADMUX-rekisterissä
  ADMUX &= 0xF8;
  // Asettaa jännitereferenssin AVcc:ksi ja asettaa kolme alinta bittiä potPin:stä riippuen
  ADMUX |= (1 << REFS0) | (potPin & 0x07);

  //kutsutaan funktiota, joka lukee potikan arvon
  int val = readPotentiometer(potPin);

  // muutetaan luettu arvo nuotin kestoksi 2 - 16 sekunnin osaa
  int dur = map(val, 150, 1000, 2, 16);

  return (dur);
}

// funktiot joka määrittää mitä nuottia soitetaan mistäkin etäisyydestä
void  tone_to_play(long cm, int mod, int dur)
{
```

```
  int noteplay = 0;
  int oct;

  // lisätään mod:ia, joka vaihtaa arraysta seuraavan oktaavin
  if (key2 == 'A')
    oct = 0;
  else if (key2 == 'B')
    oct = 10;
  else if (key2 == 'C')
    oct = 20;
  else if (key2 == 'D')
    oct = 30;

  // etäisyyden tarkistus ja oikean nuotin asetus. mod muuttaa säveltä ja oct oktaavia
  if (cm >= 2 && cm < 5)
    noteplay = notes[oct + mod][0];
  if (cm >= 5 && cm < 8)
    noteplay = notes[oct + mod][1];
  if (cm >= 8 && cm < 11)
    noteplay = notes[oct + mod][2];
  if (cm >= 11 && cm < 14)
    noteplay = notes[oct + mod][3];
  if (cm >= 14 && cm < 17)
    noteplay = notes[oct + mod][4];
  if (cm >=  17 && cm < 20)
    noteplay = notes[oct + mod][5];
  if (cm >= 20 && cm < 23)
    noteplay = notes[oct + mod][6];
  if (cm > 21 && dur == 0)
    noTone(12);
  // jos etäisyys on löytynyt soitetaan ääni
  else
  {
    //määritetään nuotin pituus sekuntti/dur
```

```
    int notedur = 1000/dur;
    //syötetään nuotti kaijuttimeen
    tone(12, noteplay, notedur);
    // nuotin pituuden 1.3 kertainen viive, jotta lyhyimmät nuottivälit erottuvat
    delay(notedur* 1.3);
  }
}

// Kutsutaan kun ylivuoto tapahtuu
ISR(TIMER1_COMPA_vect)
{
  timerCounter++; // inkrementoituu joka sekunti
}
```