VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES

Juho Sneck

LIIKKUVA HERÄTYSKELLO

Tekniikka
2023

VAASAN AMMATTIKORKEAKOULU
Tietotekniikka

TIIVISTELMÄ

Tekijä	Juho Sneck
Opinnäytetyön nimi	Liikkuva herätyskello
Vuosi	2023
Kieli	suomi
Sivumäärä	37
Ohjaaja	Santiago Chavez

Tämän opinnäytetyön tarkoituksena oli tehdä herätyskello, joka voisi lähteä liikkeelle herätyksen alettua. Tällaisen liikkuvan herätyskellon tarkoitus olisi helpottaa käyttäjien heräämistä siten, että herätystä ei voi sammuttaa sängystä käsin. Tämä voisi parantaa nukkumistottumuksia.

Liikkuva herätyskello ei ole uusi keksintö, mutta niitä ei ole paljon markkinoilla. Tarkoituksena oli luoda tuotteen prototyyppi, josta voisi selvitä mitä kaikkea tällaisen kellon rakentamiseen vaadittaisiin. Aihe oli myös mielenkiintoinen, koska tässä on mahdollista luoda jotain toimivaa alusta alkaen. Työssä käytettiin monia erilaisia tekniikoita, kuten piirilevyn suunnittelua ja valmistamista, ohjelmointia ja 3D-tulostamista.

Lopputuloksena oli toimiva herätyskello, joka pystyy liikkumaan ja väistämään edessään olevia esteitä. Kellosta jouduttiin jättämään pois muutama ominaisuus ajan puutteen vuoksi. Näitä ylimääräisiä ominaisuuksia on kuitenkin mahdollista lisätä tuotteeseen myöhemmin.

Avainsanat herätyskello, piirilevy, 3D-tulostus, ohjelmointi

VAASAN AMMATTIKORKEAKOULU
UNIVERSITY OF APPLIED SCIENCES
Tietotekniikka

ABSTRACT

Author	Juho Sneck
Title	Moving Alarm Clock
Year	2023
Language	Finnish
Pages	37
Name of Supervisor	Santiago Chavez

The purpose of this thesis was to make an alarm clock that could go off when the alarm is started. The purpose of such a moving alarm clock would be to make it easier for users to wake up so that the alarm cannot be turned off from the bed. This could improve sleeping habits.

A moving alarm clock is not a new invention, but there are not many of them on the market. The purpose was to create a prototype of the product, from which it would be possible to find out what would be required to build such a watch. The topic was also interesting because here it is possible to create something that works from scratch. Many different techniques were used in the work, such as circuit board design and manufacturing, programming and 3D printing.

The result is a functional alarm clock that can move and dodge obstacles in front of it. A few features had to be left out of the watch due to lack of time. However, it is possible to add these additional features to the product later.

Keywords Alarm clock, circuit board, 3D printing and programming

SISÄLLYS

TAULUKKO- JA KUVALUETTELO

TERMIT JA LYHENTEET

LCD	Nestekidenäyttö.
OLED	Orgaanisia yhdisteitä sisältävä näyttö.
AVR	Mikrokontrolleriperhe .
PWM	Pulssinleveysmodulaatio.
DC	Tasavirta.
UART	Sarjaliikennepiiri, jolla voidaan siirtää dataa.
IDE	Ohjelmointiympäristö.

1 JOHDANTO

Projekti on herätyskello, joka lähtee liikkeelle herättäessään. ”Torkuttaminen” on tuttu tapa monelle ihmiselle saada hieman lisää uniaikaa, ja helpotusta heräämiseen. Tämä ei kuitenkaan ole tehokasta nukkumista, eikä siten virkistä oikeasti nukkujaa.[1] Nukkuja voi myös unissaan sammuttaa herätyskellon ilman heräämistä. Näihin ongelmiin voitaisiin hakea ratkaisua kellolla, joka lähtee karkuun herättäessään. Kellon tarkoitus on siis pakottaa herääjä nousemaan ylös sängystä torkuttamisen sijaan. Myös tarkoitukseton herätyksen sammuttaminen voitaisiin ehkäistä sillä, että herätyskello ei ole käden ulottuvilla. Heräämisen vaikeus pienentyy huomattavasti silloin, kun nukkuja on joutunut nousemaan ylös kertaalleen. Tämä tuotteen tarkoitus olisi siis pakottaa käyttäjä nousemaan, ja tällä tavalla poistaa mahdollisesti vaikea kohta heräämisessä. Tuote sopisi hyvin esimerkiksi henkilöille, jotka haluaisivat eroon torkuttamisesta tai joilla on vaikeuksia herätä herätyskelloon.

[1] McCallum, Does Hitting the Snooze Button Help or Hurt, 2021

2 VAATIMUKSET

Suunnitteluvaiheessa mietitään, että kuinka projekti tullaan toteuttamaan. Toteutustapoja voi olla monia erilaisia, eikä tämä tapa välttämättä ole paras vaihtoehto.

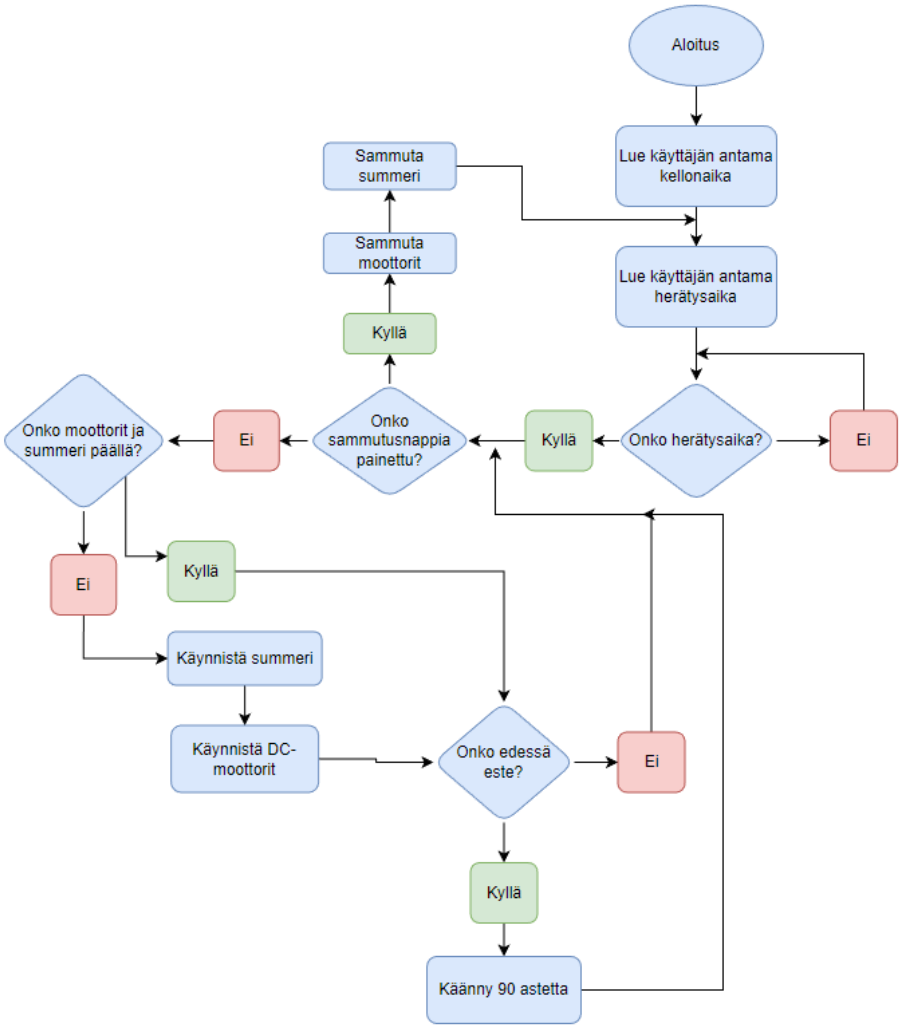
2.1 Alustava suunnittelu

Tuote on siis herätyskello, joka lähtee liikkeelle silloin, kun laite herättää käyttäjää. Liikkuminen tehtäisiin kahdella moottorilla, ja kello painotettaisiin niin, että se tulee olemaan aina ”oikein” päin. Kelloon tulisi asentaa jonkinlaiset jouset, jotta siinä tilanteessa, kun kello lähtee liikkeelle ja tippuu pöydältä, se ei menisi rikki. Kelloon tarvittaisiin näyttö. Näytön avulla tulostettaisiin käyttäjälle kellonaika, ja muu käyttöliittymä. Kellossa tulisi olla painonappeja, joilla voidaan säätää kellonaika, asettaa herätys ja sammuttaa herätys. Nämä napit voisivat tehdä myös muita toimintoja, varsinkin jos käytössä olisi OLED-näyttö. Kelloon tarvitaan etäisyysanturi, jonka avulla kello voi kääntyä, jos se havaitsee esteen edessään. Kello tarvitsee virtalähteen itselleen.

Virtalähde voisi toimia paristoilla, mutta ekologisempi vaihtoehto voisi olla ladattava akku. Laadukkaasta akusta voitaisiin saada virtaa pidemmäksi ajaksi kellolle. Kello tarvitsisi kaiuttimen tai summerin, joka toimisi herättävänä komponenttina. Herätyskelloon voisi myös lisätä pehmeää valoa tuottavan lampun, minkä avulla käyttäjän heräämiskokemus tulisi olemaan vaivattomampi. Kello tulisi toimimaan AVR-mikrokontrollerilla. ATmega8-mikrokontrolleri voisi olla sopiva laitteelle, mutta laite voi myös vaatia tehokkaampaa mikrokontrolleria. Tällainen voisi olla esimerkiksi isommalla muistilla varustettu prosessori.

2.2 Lohkokaavio

Projektin toimivuuden kannalta on järkevää tehdä lohkokaavio, minkä perusteella on helpompi tutkia kellon toimintaa. Kuvassa 1 selitetään kellon toimintaperiaate.



Kuva 1. Lohkokaavio.

Ohjelma alkaa suorittamaan itseään silloin, kun laitteistoon kytketään virta. Välittömästi virtojen kytkeydyttyä pyydetään käyttäjältä kellonaika. Tämä on välttämätön toimenpide, koska mikrokontrollerin ei ole mahdollista pitää kellonaikaa yllä ilman virtaa. Kellonajan syöttämisen jälkeen kysytään käyttäjältä aika, milloinka kellon halutaan herättää. Käyttäjä antaa kellonajan ja herätysajan painamalla nappeja. Tämän jälkeen kello alkaa tulostamaan aikaa LCD-näytölle, ja tarkastamaan, että onko herätysaika. Tätä kyselyä suoritetaan niin kauan, että herätysaika tulee.

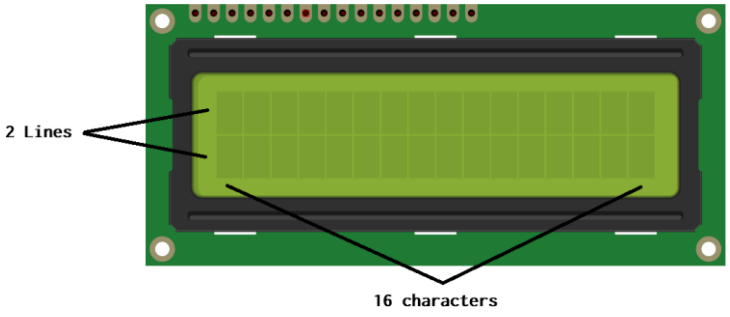
Herätysajan jälkeen tarkastetaan, että onko käyttäjä painanut herätyksen sammutusnappia. Jos nappia ei olla painettu, alkaa käyttäjän herättäminen. Tähän siis kuuluu moottorien ja summerin käynnistäminen. Moottori ja summeri pidetään niin kauan aikaa päällä, kunnes käyttäjä painaa sammutusnappia. Jos herätyksen aikana kello havaitsee edessään esteen, se kääntyy 90 astetta, ja jatkaa liikkumista. Sen jälkeen, kun sammutusnappia on painettu, palataan takaisin kysymään käyttäjältä herätysaika. Näin ohjelma on suorittanut täydellisen kierroksen, ja kierros alkaa alusta.

3 KOMPONENTIT

Suunnitteluvaiheessa päätettiin mitä komponentteja tultaisiin käyttämään opinnäytetyössä. Tähän on listattu tärkeimmät komponentit ja niiden toiminta. Näiden komponenttien lisäksi työssä tullaan käyttämään esimerkiksi vastuksia, kondensaattoreita ja diodeja, sekä muita piirin toimimisen kannalta tärkeitä komponentteja.

3.1 LCD-näyttö

LCD-näyttö eli nestekidenäyttö on yleisesti käytetty näyttötekniikka, jota käytetään monissa laitteissa, kuten tietokoneiden näytöissä, matkapuhelimissa ja digitaalisissa kelloissa. LCD-näyttö toimii käyttämällä nestekidejä, jotka reagoivat sähkökenttiin ja muuttavat valonsäteitä läpäisemättömiksi tai läpäiseviksi. Tämä mahdollistaa kuvan muodostamisen näytölle. LCD-näyttöjä on saatavana eri kokoisina ja eri resoluutioina, ja niiden etuihin kuuluu alhainen virrankulutus ja korkea tarkkuus.[2] Niitä käytetään laajalti sekä harrastusprojekteissa että ammattikäytössä. Kuvassa 2 näkyy työssä käytettävä 16x2 kokoinen LCD-näyttö.



Kuva 2. 2x16 LCD-näyttö.

[2] Tech Target, LCD (Liquid Crystal Display), 2019.

3.2 ATmega8-16PU-mikrokontrolleri.

ATmega8 on tehokas ja vähävirtainen 8-bittinen mikrokontrolleri.[3] ATmega8 kuuluu AVR-mikrokontrolleriperheeseen, mitä Atmel on alkanut kehittämään vuodesta 1996 lähtien.[4] AVR-mikrokontrollereita käytetään suositussa Arduino-kehitysalustoissa, ja ne ovat suosittuja harrastelijoiden ja sulautettujen järjestelmien opiskelijoiden projekteissa. Tällä mikrokontrollerilla tullaan ohjailemaan projektin komponentteja sekä kelloa. Se valikoitui projektiin prosessoriksi halpuutensa ja tehokkuutensa vuoksi. Kuvassa 3 näkyy tässä työssä käytettävä mikrokontrolleri.

Kuva 3. ATMEGA8-16PU.

3.3 L293D-moottoriohjain

L293D on yleisesti käytetty integroitu piiri, jota käytetään moottoreiden ohjaukseen erilaisissa sovelluksissa. Se on kaksisuuntainen moottoriohjain, joka kykenee ohjaamaan kahta DC-moottoria tai yhtä askelmoottoria.[5]

[3] Atmel, 8-bit Atmel with 8KBytes In-System Programmable Flash, 2013.
[4] Elprocus, AVR Atmega8 Microcontroller Architecture & Its Applications.
[5] Texas Instruments, L293x Quadruple Half-H Drivers, 2016.

L293D:n toimintaperiaate perustuu sen kykyyn ohjata moottoreiden nopeutta ja suuntaa käyttäen PWM-signaalia. Se vastaanottaa ohjaussignaalin mikrokontrollerilta tai muulta lähteeltä, joka määrittelee moottorin nopeuden ja suunnan.[6] L293D:n sisällä olevat transistoreilla voidaan ohjata virtaa, joka kulkee moottoreille, ja siten muuttaa niiden nopeutta ja suuntaa. L293D sisältää myös sisäänrakennetun suojaustoiminnon, joka suojaa piiriä ylikuormitukselta ja ylikuumenemiselta.[7] Lisäksi se on suunniteltu toimimaan laajalla jännitealueella, joka tekee siitä sopivan monenlaisiin sovelluksiin. Kuvassa 4 on esitelty L293D-moottoriohjain.

Moottoriohjainta käytetään tässä tapauksessa ohjailemaan moottoreita ja tuottamaan moottoreille 12V jännite. Moottoriohjaimessa on neljä sisääntuloa ja neljä ulostuloa moottoreille. Jokainen moottori tarvitsee kaksi, joten tällä moottoriohjaimella voidaan ohjata kahta eri moottoria. Moottorien suunta määrittyy taulukon 1 mukaan.

Taulukko 1. Moottorin pyörimissuunta.

IN1	**IN2**	**Pyörimissuunta**
Alatila	Alatila	Moottori ei pyöri
Ylätila	Alatila	Eteenpäin
Alatila	Ylätila	Taaksepäin
Ylätila	Ylätila	Moottori ei pyöri

[6] Last Minute Engineers, Control DC Motors with L293D Motor Driver IC & Arduino.
[7] Last Minute Engineers, Control DC Motors with L293D Motor Driver IC & Arduino.

Kuva 4. L293D.

3.4 DC-moottori

DC-moottori on sähkömoottori, joka toimii suoraan tasavirralla. Moottorissa on yleensä pyörivä roottori ja kiinteä staattori.[8] Staattorin ja roottorin välissä on kommutaattori, joka kääntää virtasuunnan roottorin keloissa oikeaan aikaan, jotta se pyörii jatkuvasti.[9] DC-moottorit ovat yleisesti käytettyjä monissa sovelluksissa, kuten kulkuvälineiden käyttövoimana, teollisuudessa, kotitalouslaitteissa ja robotiikassa. Niiden suuri etu on, että ne ovat helppoja ohjata nopeuden ja suunnan suhteen käyttämällä esimerkiksi PWM-signaalia. Tässä työssä valittiin kaksi 12V jännitteellä toimivia, 100 kierrosta minuutissa pyörivää moottoria. Nämä moottorit valittiin hitaan pyörimisnopeuden vuoksi, jotta kellon liikkumisnopeutta olisi helpompi kontrolloida.

3.5 Summeri

Työssä käytetään yhtenä herättävänä komponenttina summeria. Summeri on yksinkertainen komponentti, mikä tuottaa ääntä silloin kun sille syötetään virtaa.

[8] Elprocus, What is a DC MOTOR : Basics, Types & Its Working.
[9] Elprocus, What is a DC MOTOR : Basics, Types & Its Working.

Summerissa on kondensaattori, mikä ladataan sähköä syötettäessä. Varaus purkautuu silloin, kun sähkövirtaa ei syötetä.[10]

Työssä käytetään passiivista summeria. Tämä tarkoittaa sitä, että summerille on tuotettava vaihtovirtaa mikrokontrollerilta, jotta summeri saadaan tuottamaan ääntä. Äänen taajuutta voidaan muuttaa säätelemällä vaihtovirran taajuutta. Taajuus voidaan tuottaa käyttämällä esimerkiksi mikrokontrollerin PWM-ominaisuutta. Tässä työssä kuitenkin joudutaan käyttämään delay-funktiota jännitteen vaihtelemiseen, koska mikrokontrollerin PWM-lähdöt ovat jo muussa käytössä. Kuvassa 5 on työssä käytettävä summeri.

Kuva 5. Summeri.

3.6 US-100-etäisyysmittari

US-100-etäisyysmittari on ultraäänipohjainen mittalaite, joka käyttää ääniaaltoja etäisyyden mittaukseen.

[10] APC, 2020, Piezo Buzzers vs. Magnetic Buzzers — What's the Difference?

Toimintaperiaate perustuu aikaperiaatteen käyttöön: mittari lähettää ulos lyhyitä ultraääniaaltoja, jotka heijastuvat takaisin mittauspisteeseen, kun ne osuvat esteeseen.[11] Sensori mittaa ajan, joka kuluu signaalin lähettämisestä sen vastaanottamiseen takaisin mittauspisteeseen, ja käyttää tätä tietoa etäisyyden laskemiseen. Ultraääniaallot ovat ihmiskorvalle kuulumattomia, korkeataajuisia ääniä, joita voidaan käyttää etäisyyden mittaukseen tarkasti ja nopeasti.[12] Tämä kyseinen mittari sisältää myös lämpötila-anturin, mutta tätä ominaisuutta ei käytetä työssä.[13] US-100-etäisyysmittaria olisi mahdollista käyttää myös UART-tilassa. Kuvassa 6 esitellään US-100-etäisyysmittari.

Mittari saadaan aloittamaan etäisyyden mittaaminen antamalla sen TRIG-pinniin 40µs pulssi. Tämän jälkeen mittari lähettää kahdeksan kertaa 40kHz äänipulssin, ja ECHO-pinni menee ylätilaan. Kun ääniaallot palaavat takaisin mittarille, ECHO-pinni menee alatilaan. Näin saadaan aika, mikä on kulunut äänipulssilta objektille ja takaisin.

Etäisyys voidaan muuttaa senttimetreiksi seuraavalla kaavalla.

$$etäisyys = \frac{ään nopeus * aika}{2} \quad (1)$$

Kun tiedetään äänen nopeus ilmassa (343 m/s) ja aika kauanko äänellä kestää osua objektiin ja palata takaisin, voidaan laskea etäisyys objektiin.

[11] Cook, 2018, Ultrasonic Sensors: How They Work (and How to Use Them with Arduino).
[12] Cook, 2018, Ultrasonic Sensors: How They Work (and How to Use Them with Arduino).
[13] Adafruit, US-100 Ultrasonic Distance Sensor - 3V or 5V Logic.

Kuva 6. US-100.

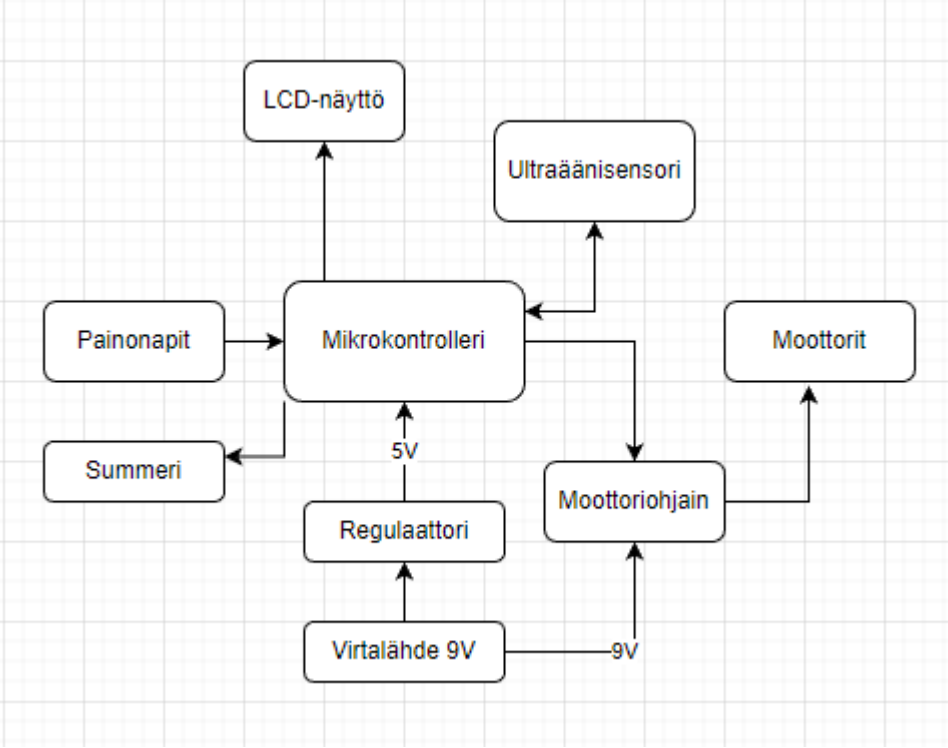
3.7 LM7805

LM7805 on regulaattori, mikä muuntaa paristolta saatavan 9V jännitteen 5V jännitteeksi. 5V jännite on tarvittava jännite mikrokontrollerille ja useille muille työssä käytettäville komponenteille. Regulaattori suodattaa ensin syöttöjännitteen, sitten säätää lähtöjännitteen ja suojaa itseään ylikuumenemiselta.[14] Näin se toimii tasajännitteen lähteenä sähköpiirissä.

[14] Texas Instruments, 2003, µA7800 SERIES POSITIVE-VOLTAGE REGULATORS.

4 TOTEUTUS

Tässä luvussa käsitellään tuotteen rakentamista ja ohjelmointia. Herätyskellossa mikrokontrolleri on tärkeimmässä roolissa. Mikrokontrolleri ohjaa kaikkia muita komponentteja suoraan paitsi moottoreita. Regulaattori muuntaa jännitteen sopivaksi mikrokontrollerille ja muille komponenteille. Moottoriohjain on ainut komponentti, mikä tarvitsee 5V jännitteen lisäksi suoraan virtaa virtalähteeltä.



Kuva 7. Funktionaalinen kuva.

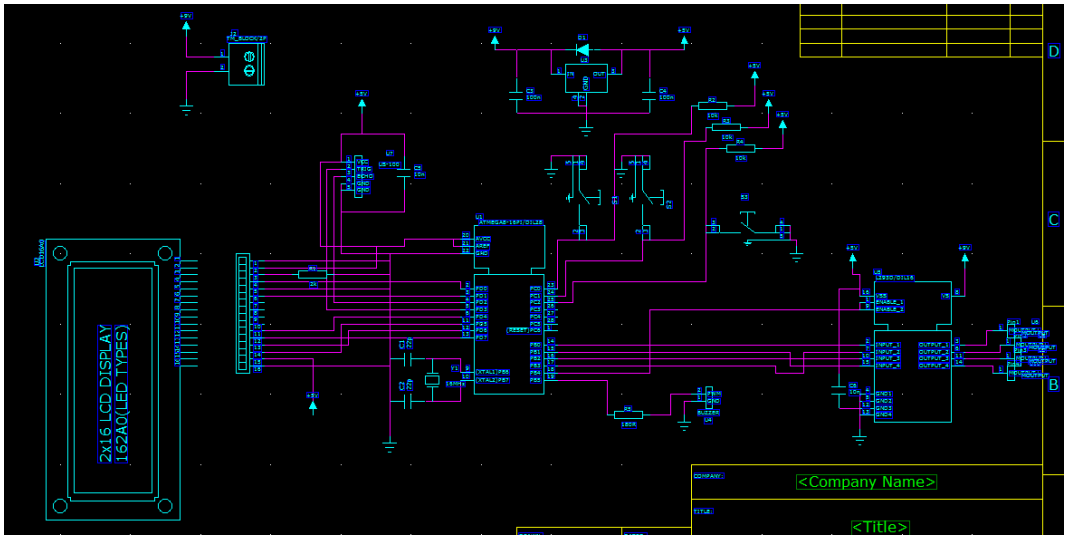
Funktionaalisesta kuvasta 7 selviää se, kuinka komponentit ovat toisissaan kiinni. Se on yksinkertaistettu kuva todellisuudesta, mutta se helpottaa tuotteen toiminnan ymmärtämistä.

4.1 Piirikaavio

Laitteen piirikaavio on suunniteltu käyttämällä PADS Logic-ohjelmaa, ja piirilevy on suunniteltu käyttämällä PADS layout-ohjelmaa. PADS logic- ja layout-ohjelmat on kehittänyt SIEMENS-osakeyhtiö, ja Vaasan ammattikorkeakoulun opiskelijat voivat

käyttää näitä ohjelmia omien projektien suunnitteluun. Näitä ohjelmia käytetään myös paljon teollisuudessa.

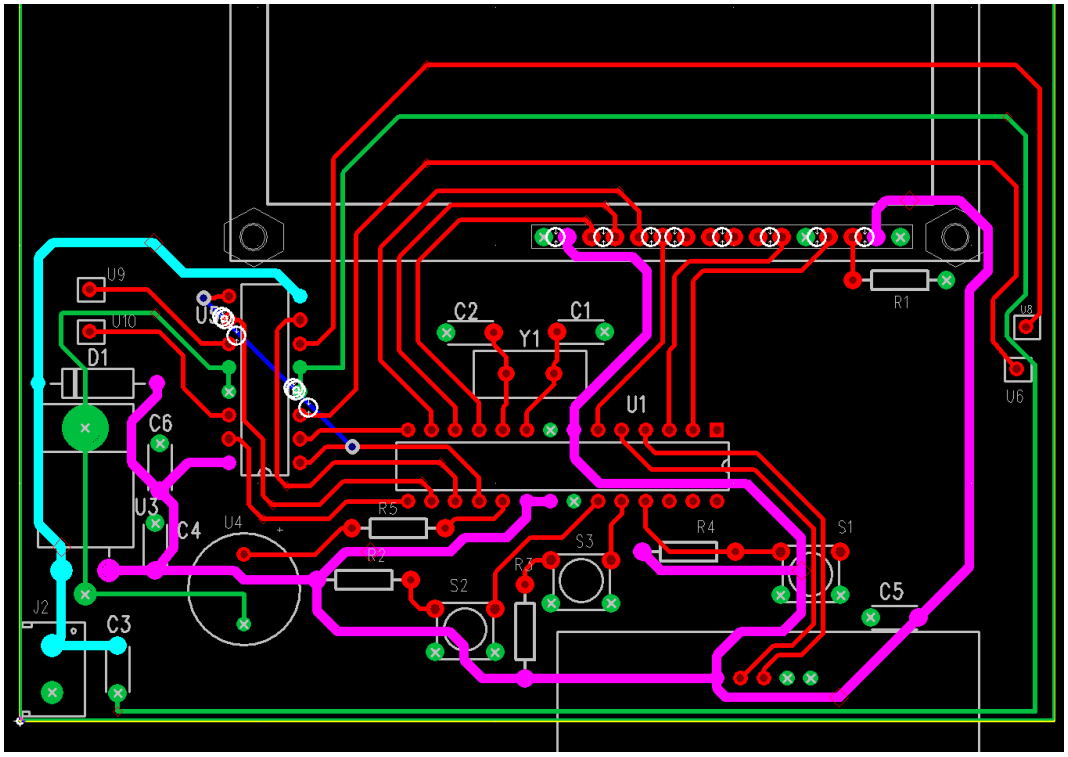
Työn piirikaavion suunnittelussa käytetään pääosin Vaasan ammattikorkeakoulun omia komponenttikirjastoja. Joitain komponentteja ei kuitenkaan ollut valmiina näissä kirjastoissa, ja ne oli luotava itse. Kuvassa 8 olevassa piirikaaviossa on kaikki komponentit mitä herätyskellossa tullaan käyttämään.



Kuva 8. Piirikaavio.

4.2 Piirilevy

Piirilevyn suunnittelussa käytettiin PADS layout-ohjelmaa, kuten aiemmin mainittiin. Tällä ohjelmalla on mahdollista tehdä fyysinen kuva piirilevystä, kun taas logic-ohjelmalla tehdään yhteydet, eikä fyysisesti oikeilla mittasuhteilla ole väliä.



Kuva 9. Piirilevyn kuva.

Kuvassa 9 on valmis piirilevy ilman kuparointia. Tässä kuvassa on vedetty kaikki tarpeelliset linjat jokaiselle komponentille. 5V ja 9V linjat on paksunnettu 1 millimetrin paksuisiksi 0,5 millimetrin sijaan. Linjojen suunnitteluehtoina on käytetty 0,5 millimetrin väliä. Tämä tarkoittaa siis sitä, että linjat eivät saa olla alle 0,5 millimetrin päässä toisistaan tai komponenttien jaloista. Piirilevyn suunnittelussa on pyritty luomaan mahdollisimman suuri maa-alue eli kuparointi mahdollisten häiriöiden vähentämiseksi. Häiriönestoon on myös käytetty kondensaattoreita mitä on sijoiteltu kriittisten komponenttien läheisyyteen. Näin voidaan minimoida häiriöiden haittaavuus piirissä.

Piirilevyn koneella valmistamisen jälkeen voidaan tulostaa kuva piirilevystä ja valmistaa se koululta saatavilla välineillä. Piirilevystä saadaan irti kupari halutuilta kohdilta erilaisten kemikaalien avulla. Tämän jälkeen piirilevyyn voidaan porata reiät kohtiin mihinkä komponentit tullaan kiinnittämään. Reikien poraamisen jälkeen voidaan aloittaa komponenttien kiinnittäminen piirilevyyn. Kun komponentit on kiinnitelty levyyn, on se valmis ohjelmointia varten.

4.3 Ohjelmointi

ATmega8-mikrokontrolleri ohjelmoitiin Arduino IDEllä. Koodi ladataan mikrokontrollerille Arduino Unon avulla. Arduinoa voidaan siis käyttää niin kutsuttuna ISP-laitteena, jolla on mahdollista ladata koodia AVR-mikrokontrollerille.[15]

Tarkoituksena on siis koodata herätyskello, mikä pitää tarkasti sille asetetun ajan ja toimisi herätyskellon tapaan. Koska mikrokontrollerina käytetään ATmega8aa, ei ole mahdollista käyttää kaikkia Arduinoille suunniteltuja kirjastoja prosessorin rajallisen muistin vuoksi. Tämän vuoksi koodissa on pyritty säästämään muistia siten, että käytetään prosessorin omia rekistereitä esimerkiksi tarkan kellon luomiseen.

Ensimmäisenä koodissa määritellään eri pinneille nimiä koodaamisen helpottamiseksi, lisätään tarvittavia kirjastoja ja määritellään funktiot. Työn koodaamisen nopeuttamiseksi LCD-näytön ohjelmoimiseksi käytettiin lcd.h kirjastoa. Kirjasto on suhteellisen pieni, ja sisältää kaikki tarvittavat funktiot LCD-näytön ohjaamiseen.[16] Muut kirjastot tuovat käyttöön helpottavia funktioita, kuten delay-funktion.

[15] SM, 2023, Arduino as ISP and Arduino Bootloaders.
[16] George, 2020, Interfacing LCD with Atmega32 Microcontroller using Atmel Studio.

```
volatile uint8_t seconds = 0, minutes = 0, hours = 0, alarm_minutes = 0, alarm_hours = 0, alarm_seconds = 0;
uint16_t end = 0, start = 0;

// Katkaisurutiini, joka ajetaan joka kerta kun Timer1 on suorittanut keskeytyksen
ISR(TIMER1_COMPA_vect) {
    seconds++; // Kasvatetaan sekunti muuttujaa
    if (seconds == 60) { // Minuutti on kulunut
      seconds = 0;
      minutes++; // Kasvatetaan minuutti muuttujaa
      if (minutes == 60) { // Tunti on kulunut
        minutes = 0;
        hours++; // Kasvatetaan tunti muuttujaa
        if (hours == 24) { // Päivä on vaihtunut
          hours = 0;
        }
      }
    }
}
```

Kuva 10. Katkaisurutiinin määrittely.

Kuvassa 11 määritellään katkaisurutiini ATmega8n ajastimelle. Tämä katkaisurutiini tullaan suorittamaan aina, kun sisäisen kellon mukaan on kulunut sekunti. Katkaisurutiinin sisällä kelloon lisätään aikaa niin kauan, että 24 tuntia on kulunut. Sen jälkeen kello nollaantuu.

```
void setup() {

  cli(); // Poista kaikki keskeytykset käytöstä
  TCCR1A = 0; // Aseta Timer/Counter1:n A-rekisteri nollaksi
  TCCR1B = 0; // Aseta Timer/Counter1:n B-rekisteri nollaksi
  TCNT1 = 0; // Aseta Timer/Counter1:n arvo nollaksi
  OCR1A = 15624; // Aseta compare-arvo
  TCCR1B |= (1 << WGM12); // Aseta CTC-moodi
  TCCR1B |= (1 << CS12) | (1 << CS10); // Aseta prescaler arvoksi 1024
  TIMSK |= (1 << OCIE1A); // Aseta compare-match interrupt-pyyntö päälle
  sei(); // Salli keskeytykset
```

Kuva 11. Ajastimen asetukset.

Seuraavaksi kuvassa 12 asetetaan ajastimen asetukset. Kun on käytössä 16MHz kellotaajuudella toimiva ATmega8-mikrokontrolleri, käyttämällä 15624 compare-arvoa ja 1024 prescaler-arvoa saadaan aikaan keskeytys, mikä tapahtuu sekunnin

välein.[17] Tämä on tarpeeksi tarkka ottaen huomioon sen, että kellon aikaa voi tarvittaessa muuttaa.

Kuvassa 14 nähdään pääkoodin suoritus. Tätä koodia suoritetaan jatkuvasti. Koodissa tulostetaan näytölle kellonaika käyttäen lcd.h kirjaston funktioita. Tässä myös tarkistetaan jatkuvasti, että onko aiemmin asetettu herätys sama kuin kellonaika. Painamalla nappia 1 on myös mahdollista päästä menu-funktioon.

[17] Atmel, 2016, AVR134: Real Time Clock (RTC) Using the Asynchronous Timer.

```
void loop() {

    char buffer[3];

    Lcd4_Set_Cursor(0,1);
    Lcd4_Write_String("KLO:");

  if(hours<10){
    Lcd4_Write_String("0");
  }
    sprintf(buffer, "%d", hours);
    Lcd4_Write_String(buffer);
    Lcd4_Write_String(":");

  if(minutes<10){
    Lcd4_Write_String("0");
  }
    sprintf(buffer, "%d", minutes);
    Lcd4_Write_String(buffer);
    Lcd4_Write_String(":");

  if(seconds<10){
    Lcd4_Write_String("0");
  }
    sprintf(buffer, "%d", seconds);
    Lcd4_Write_String(buffer);
    _delay_ms(150);
    Lcd4_Clear();

    if(alarm_minutes == minutes && alarm_hours == hours && alarm_seconds == seconds){
      alarm();
    }

    if(bit_is_clear(PINC, button_1)){
      menu();
    }
```

Kuva 12. Pääkoodi.

Kuvassa 15 näkyy menu-funktion toiminta kokonaan. Napeilla 2 ja 3 voidaan käydä läpi valikon eri vaihtoehtoja. Valikossa on kolme vaihtoehtoa, aseta aika uudestaan, aseta herätys uudestaan ja poistu valikosta.

```
void menu(){
  uint8_t i = 0;
  Lcd4_Clear();
  Lcd4_Set_Cursor(1,6);
  Lcd4_Write_String("Menu");

  while(1){
    _delay_ms(100);
    if(bit_is_clear(PINC, button_2)){
      i++;
      _delay_ms(35);
      if(i==3){
        i=0;
      }
    }

    if(bit_is_clear(PINC, button_3)){
      i--;
      _delay_ms(35);
      if(i==255){
        i=2;
      }
    }


    if(i==0){
      Lcd4_Set_Cursor(2,1);
      Lcd4_Write_String("Set time        ");

      if(bit_is_clear(PINC, button_1)){
        _delay_ms(35);
        if(bit_is_clear(PINC, button_1)){
          set_time();
         return;
        }
      }
    }

    if(i==1){
      Lcd4_Set_Cursor(2,1);
      Lcd4_Write_String("Set alarm       ");

      if(bit_is_clear(PINC, button_1)){
        _delay_ms(35);
        if(bit_is_clear(PINC, button_1)){
          set_alarm();
          return;
        }

      }
    }
    if(i==2){
      Lcd4_Set_Cursor(2,1);
      Lcd4_Write_String("Exit            ");
      if(bit_is_clear(PINC, button_1)){
        _delay_ms(35);
        if(bit_is_clear(PINC, button_1)){
          return;
        }
      }
    }
  }
}
```

Kuva 13. Menu-funktion toiminta.

```
void set_time(){ //Asetetaan aika
  Lcd4_Clear();
  Lcd4_Set_Cursor(1,4);
  Lcd4_Write_String("Set time");
  _delay_ms(200);

  while(!bit_is_clear(PINC, button_1)){//Jos nappia 1 ei ole painettu, jatketaan
    char buffer[3];
    Lcd4_Set_Cursor(1,4);
    Lcd4_Write_String("Set time");
    Lcd4_Set_Cursor(2,1);
    Lcd4_Write_String("Time:");
    sprintf(buffer, "%d", hours);
    Lcd4_Write_String(buffer);
    Lcd4_Write_String(":");
    sprintf(buffer, "%d", minutes);
    Lcd4_Write_String(buffer);

    if(bit_is_clear(PINC, button_2)){ //Jos nappia 2 painetaan, lisätään aikaa kelloon
      minutes++;                      //Lisätään minuutti
      Lcd4_Clear();
      if(minutes==60){                //Lisätään tunti
        minutes=0;
        hours++;
        Lcd4_Clear();
        if(hours==24){
          hours=0;
          Lcd4_Clear();
        }
      }
      _delay_ms(50);
    }
    if(bit_is_clear(PINC, button_3)){//Jos nappia 3 painetaan, poistetaan aikaa kellosta
    minutes--;                       //Poistetaan minuutti
    Lcd4_Clear();
      if(minutes==255){             //Poistetaan tunti
        minutes=59;
        hours--;
        Lcd4_Clear();
        if(hours==255){
          hours = 23;
          Lcd4_Clear();
        }
      }
    _delay_ms(50);
    }
  }
  _delay_ms(200);
```

Kuva 14. Ajan asettamisen funktio.

Kuvassa 16 on esiteltynä funktio, missä voidaan asettaa aika kellolle. Aiemmin on määritelty globaaleja muuttujia, ja näitä muuttujia voidaan muokata funktioissa ilman, että niitä tarvitsee sinne erikseen viedä. Ensin tulostetaan näytölle nykyinen kellonaika. Tämän jälkeen painamalla nappia 2 voidaan lisätä aikaa kelloon 50ms

välein. Painamalla nappia 3 voidaan vähentää aikaa kellossa. Kun käyttäjä painaa nappia 1, poistutaan while-silmukasta ja suoritus poistuu funktiosta. Herätyksen asettamisen funktio on hyvin samanlainen ajan asettamisen funktion kanssa. Siinä kellon ajan muuttamisen sijaan muutetaan herätyksen aikaa. Funktion toiminnallisuus ei siis muutu merkittävästi.

```
void alarm(){
  uint8_t i;
  /Lcd4_Clear();
    Lcd4_Set_Cursor(1,0);
    Lcd4_Write_String("WAKE UP AND");
    Lcd4_Set_Cursor(2,0);
    Lcd4_Write_String("CATCH ME!!!!");

    PORTB |= (1 << motor_enable); //Moottorin enable pinni ylös

  while(!bit_is_clear(PINC, button_1)){

    PORTB |= (1 << motor_in1); //Vasen rengas eteenpäin
    PORTB |= (1 << motor_in3); //Oikea rengas eteenpäin


    for(i=0; i<200; i++){     //Summeri päälle
     PORTB |= (1 << buzzer);
    _delay_ms(1);
     PORTB &= ~(1 << buzzer);
    _delay_ms(1);
    }
    for(i=0; i<100; i++){     //Summeri päälle eri taajuudella
     PORTB |= (1 << buzzer);
    _delay_ms(2);
     PORTB &= ~(1 << buzzer);
    _delay_ms(2);
    }
    for(i=0; i<250; i++){     //Summeri päälle eri taajuudella
     PORTB |= (1 << buzzer);
    _delay_ms(0.7);
     PORTB &= ~(1 << buzzer);
    _delay_ms(0.7);
    }
```

Kuva 15. Herätysfunktion alku.

Kun herätykseen asetetun ajan luvut ovat samat kuin kellon ajat, ohjelma siirtyy suorittamaan alarm-funktiota. Kuvasta 17 nähdään, että ohjelma tulostaa ensin tekstiä näytölle. Tämän jälkeen se käynnistää moottorit ja alkaa pyörittää niitä eteenpäin.

Seuraavaksi käynnistetään summeri. Koska summeri on pietsosähköinen summeri, äänen tuottamiseen tarvitaan nopeaa signaalin muuttumista. Tämä voidaan toteuttaa lyhyellä dealy-funktiolla. Summeri ei voi olla jatkuvasti päällä tämän vuoksi, mutta ohjelma palaa aina uuden silmukan jälkeen soittamaan summeria. Summeri soi aluksi 1000Hz taajuudella. Tämä voidaan laskea kaavalla 2.

$$taajuus = \frac{1}{aika} \tag{2}$$

Seuraavaksi summeria soitetaan 500Hz taajuudella ja tämän jälkeen noin 1400Hz taajuudella.

```
PORTD |= (1 << TRIGGER_PIN); //Etäisyyden mittaaminen.
  _delay_us(10);              //Lähetetään trig pinnille signaali
  PORTD &= ~(1 << TRIGGER_PIN);

  while (!(PIND & (1 << ECHO_PIN)))  //Odota, että ECHO_PIN menee ylätilaan
    start = TCNT1;                   //Tallenna aika, kun signaali lähetettiin

  while (PIND & (1 << ECHO_PIN))     //Odota, että ECHO_PIN menee alatilaan
    end = TCNT1;                    //Tallenna aika, kun signaali palasi takaisin

  uint16_t duration = end - start;
  if(duration < 20){
    turn(duration);
  }
```

Kuva 16. Herätysfunktion loppu.

Jotta voitaisiin välttyä siltä, että herätyskello törmää seinään, tulee mitata etäisyyttä. Kuvassa 18 käytetään ultraäänisensoria etäisyyden mittaamiseen. Jos havaitaan, että edessä on objekti lähempänä kuin 20 cm, annetaan toiselle moottorille käsky pyöriä vastakkaiseen suuntaan sekunnin ajan, ja tämän jälkeen jatkaa normaalisti.

4.4 Kotelon valmistus

Herätyskellon tarvitsee jonkinlaisen kotelon, jotta renkaat ja moottorit voidaan liittää kiinni piirilevyyn. Yksittäisiin prototyyppeihin ja projekteihin on kätevää tulostaa 3D-tulostimella osia.

4.4.1 Mallinnus

Mallintaminen tehtiin Tinkercad-nimisellä sovelluksella. Tällä sovelluksella voidaan mallintaa muun muassa 3D-mallinnuksia. Ohjelma on ilmainen, ja se toimii verkkoselaimessa.

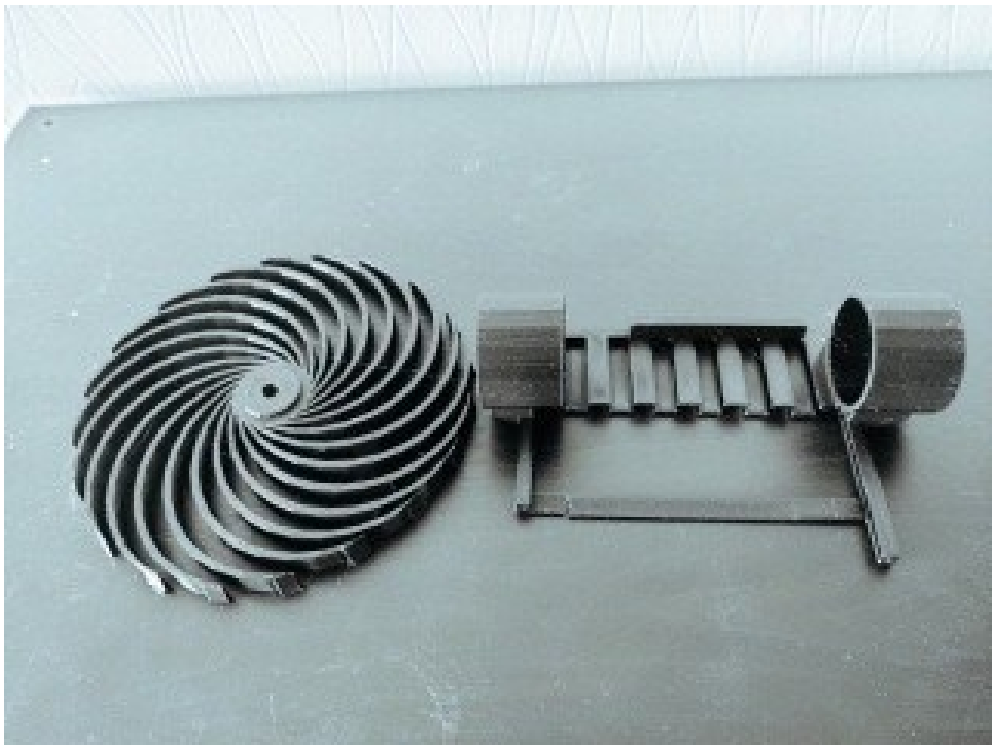
Herätyskello tarvitsee jonkinlaisen jousituksen siltä varalta, että se tulee tipahtamaan pöydältä herätyksen alkaessa. Perinteisen kierrejousituksen ja pyöräntuennan sijaan joustavuus päätettiin tehdä renkailla. Renkaista halkaisija on 15 cm, ja sen ulkoreuna ei ole kiinteä. Renkaassa on 2,5 mm paksuisia kaarevia liuskoja. Nämä liuskat joustavat ja mahdollistavat herätyskellon selviämisen pudotukselta.

Moottorit yhdistetään piirilevyyn yksinkertaisen kotelon avulla. Koteloon voidaan pujottaa piirilevy sisään. Piirilevy itsessään tukee koteloa, joten kotelon ei tarvitse olla umpinainen, ja näin voidaan säästää muovia. Moottorit pujotetaan kotelon rei’istä sisään, ja ne liimataan kiinni.

4.4.2 Tulostus

Tulostamista varten 3D-mallit tulee valmistella tulostusohjelmalla. Valmistelussa käytettiin Ultimaker Cura-ohjelmaa. Ohjelma on ilmainen, ja sen voi ladata Ultimaker valmistajan verkkosivuilta. Valmistelun jälkeen kappaleet voidaan tulostaa 3D-tulostimella. Tulostamiseen käytetään Creality Ender 3 V2 merkkistä tulostinta. Tällä tulostimella on mahdollista tulostaa esimerkiksi PLA-muovia, mikä on yksi

suosittu biohajoava muovi.[18] Kuvassa 19 on tulostettuna yksi rengas ja kotelo, mikä yhdistää kaikki osat toisiinsa.



Kuva 17. Tulostettu rengas ja kotelo.

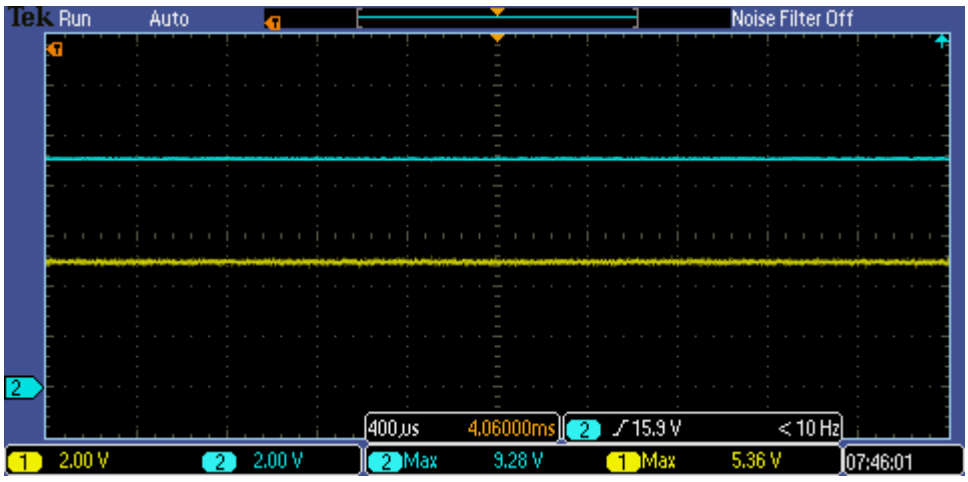
[18] Syvänne, 2020, BIOPOHJAISET JA BIOHAJOAVAT MUOVIT.

5 TESTAUS

Testauksen avulla voidaan todentaa herätyskellon toimiminen ja löytää mahdolliset virheet työstä. Testaus aloitettiin testaamalla piirilevyn signaalien johtavuus. Tämä testi suoritettiin yksinkertaisesti käyttämällä yleismittaria ja tarkistamalla että johtimissa ei ole liikaa vastusta.

5.1 Regulaattorin testaus

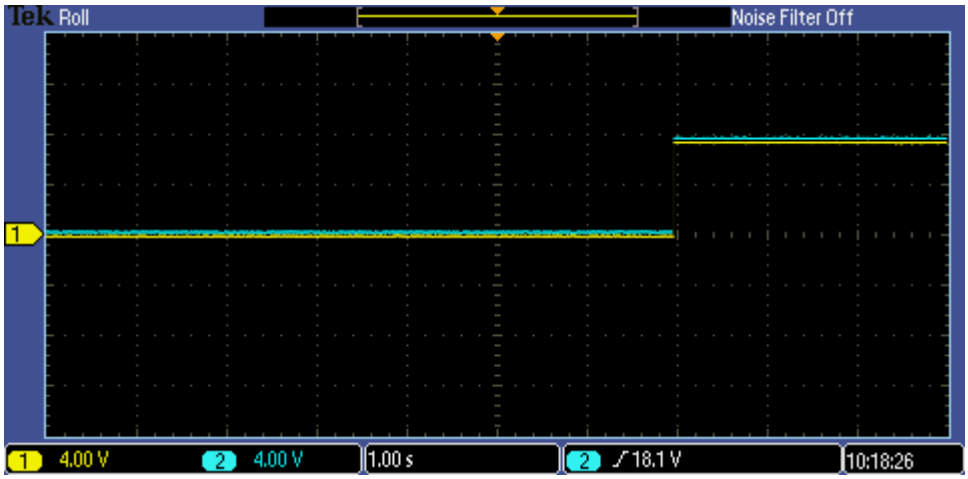
Tämän jälkeen testattiin LM7805-regulaattorin toimivuutta. Tässä työssä regulaattorin tulisi muuntaa virtalähteestä saatava jännite 5V tasajännitteeksi piirille. Testaus suoritettiin käyttämällä oskilloskooppia. Oskilloskoopilla mitattiin virtalähteen jännite ja sitä verrattiin regulaattorin muuntamaan jännitteeseen. Oskilloskoopin kuvassa keltainen viiva kuvastaa regulaattorin muuntamaa jännitettä ja sininen viiva kuvastaa virtalähteen jännitettä. Samasta kuvasta 20 voidaan havaita, että regulaattori muutti jännitteen halutulla tavalla noin 9V jännitteestä 5V tasajännitteeksi.



Kuva 18. Regulaattorin muuntama jännite.

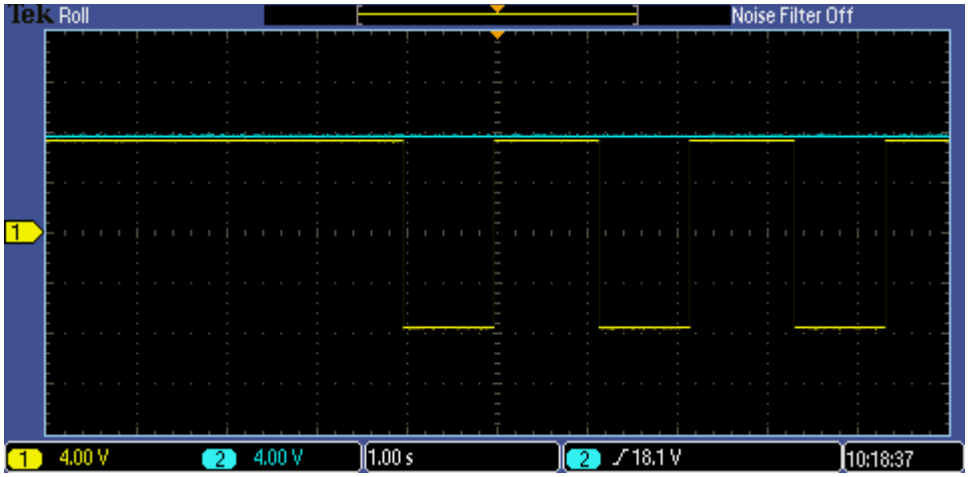
5.2 Moottoriohjaimen testaus herätysfunktiossa

Moottoriohjaimen signaalien testaaminen tehtiin myös oskilloskoopilla. Tarkoituksena oli varmistua siitä, että mikrokontrolleri lähettää oikeat signaalit moottoriohjaimelle. Samalla testataan, että koodin herätysfunktio ja etäisyysmittari toimivat halutulla tavalla. Testi suoritettiin siten, että oskilloskooppi kytkettiin moottoriohjaimen ulostuloihin 1 ja 3. Nämä ulostulot menevät moottoreiden positiivisiin napoihin. Tämän jälkeen herätyskellosta laitetiin herätysfunktio päälle, ja mitattiin tuloksia oskilloskoopista. Kuvasta 21 nähdään, että kun herätysfunktio käynnistyy, moottoreiden tulot 1 ja 3 nousevat noin 9V jännitteeseen.



Kuva 19. Herätysfunktion aikana moottoreiden tulot 1 ja 3.

Tämä tarkoittaa sitä, että moottorit pyörivät eteenpäin ja näin ollen herätyskello liikkuu eteenpäin. Samalla kun herätyskellon herätysfunktio on päällä, testataan etäisyysmittarin toiminta. Mittarin eteen laitettiin esine, ja kuvasta 22 nähdään, että toinen ulostuloista muuttui negatiiviseksi jännitteeksi. Tämä tarkoittaa sitä, että toinen moottoreista pyörii nyt vastakkaiseen suuntaan. Sekunnin päästä moottori palaa pyörimään eteenpäin. Kuvasta nähdään, että estettä pidettiin jonkin aikaa sensorin edessä, mikä aiheutti monta moottorin suunnan vaihdosta.



Kuva 20. Moottoreiden ulostulot, kun havaitaan este.

6 YHTEENVETO JA JATKOKEHITYS

Herätyskello toimii halutulla tavalla, mutta se ei ole täydellinen. Kellossa on monia kehityksen kohteita, mitkä voisivat huomattavasti parantaa käyttäjäkokemusta ja kellon luotettavuutta.

6.1 Parannukset

Herätyskelloon olisi mahdollista lisätä valonlähde. Valo alkaisi kirkastua hitaasti ennen asetettua herätysaikaa, mikä helpottaisi heräämistä etenkin pimeinä vuodenaikoina. Tämä ominaisuus toisi käyttäjälle mukavuutta, ja voisi tehdä tuotteesta mielenkiintoisemman. Kelloon voisi myös lisätä esimerkiksi bluetooth-vastaanottimen, minkä avulla käyttäjä voisi asettaa kellonajan ja herätysajan käyttämällä esimerkiksi puhelinta. Tämä ominaisuus olisi nykypäivänä houkutteleva ja helpottaisi kellon käyttämistä.

Kellosta tuli suhteellisen iso ottaen huomioon sen toiminnan. Kokoa olisi mahdollista pienentää suunnittelemalla piirilevyn paremmin ja käyttämällä SDM-komponentteja. Kellon kokoon myös vaikuttaa sen moottorit. Tässä kellossa moottorit ovat turhan suuret. Pienemmät moottorit käyttäisivät vähemmän energiaa ja mahdollistaisivat kevyemmän ja pienemmän kellon suunnittelemisen. Energialähteenä kellolle toimii 9V paristo, minkä voisi päivittää suuremman kapasiteetin ladattavaan akkuun. Näin käyttäjän ei tarvitsisi ostaa aina uusia paristoja vanhojen kuluessa loppuun.

6.2 Loppupäätelmä

Tämän projekti on ollut mielestäni sopivan haastava ja monipuolinen. Olen saanut käyttää monia eri tekniikoita, kuten piirilevyn suunnittelua, koodausta ja 3D-mallintamista ja -tulostamista. Tämän vuoksi projekti on ollut mielenkiintoinen ja uskon, että tulen kehittämään tätä projektia opinnäytetyöprosessin jälkeenkin. Mielestäni projekti oli onnistunut ja tulokset olivat toivottuja. Kaikki alussa määritellyt tavoitteet eivät täyttyneet, mutta tärkeimmät funktiot ovat kuitenkin kellossa.

Summerin toiminnassa on jonkinlaista häiriötä. Se ei toista ääntä tasaisesti, vaan saattaa joissain tilanteissa toistaa vääriä taajuuksia, tai lakata soimasta kokonaan. Tämä on kuitenkin harvinaista, ja korjaantuu yleensä itsestään. Myös HC-100 sensorin mittauksessa oli ongelmia. Kun kello keinuu, se saattaa lukea etäisyyden lattiasta, ja näin ollen virheellisesti uskoa, että este olisi edessä. Tämä ei kuitenkaan vaikuta kellon toimintaan kriittisesti, ja on korjattavissa esimerkiksi ylimääräisellä renkaalla.

Haasteellisinta projektissa oli saada sisäinen kello toimimaan niin, että se ei jätä. Ajastimen määritteleminen oli mielestäni vaikeaa, ja siinä tuli ottaa huomioon monia eri rekistereitä. Tämä kuitenkin onnistui lopulta pitkän tiedonhankkimisen jälkeen.

LÄHTEET

Adafruit. US-100 Ultrasonic Distance Sensor - 3V or 5V Logic. Viitattu 20.3.2023. https://media.digikey.com/pdf/Data%20Sheets/Adafruit%20PDFs/4019_Web.pdf

APC. 2020. Piezo Buzzers vs. Magnetic Buzzers — What's the Difference? Viitattu 14.4.2023. https://www.americanpiezo.com/blog/piezo-buzzers-vs-magnetic-buzzers/

Atmel. 2013. 8-bit Atmel with 8KBytes In-System Programmable Flash. Viitattu 12.3.2023. https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-2486-8-bit-AVR-microcontroller-ATmega8_L_datasheet.pdf

Atmel. 2016. AVR134: Real Time Clock (RTC) Using the Asynchronous Timer. Viitattu 20.4.2023. https://ww1.microchip.com/downloads/en/Appnotes/Atmel-1259-Real-Time-Clock-RTC-Using-the-Asynchronous-Timer_AP-Note_AVR134.pdf

Cook, J. 2018. Ultrasonic Sensors: How They Work (and How to Use Them with Arduino). Viitattu 20.4.2023. https://www.arrow.com/en/research-and-events/articles/ultrasonic-sensors-how-they-work-and-how-to-use-them-with-arduino

Elprocus. AVR Atmega8 Microcontroller Architecture & Its Applications. Viitattu 12.3.2023. https://www.elprocus.com/avr-atmega8-microcontroller-architecture-applications/

Elprocus. What is a DC MOTOR : Basics, Types & Its Working. Viitattu 14.4.2023. https://www.elprocus.com/dc-motor-basics-types-application/

George, L. 2020. Interfacing LCD with Atmega32 Microcontroller using Atmel Studio. Viitattu 20.4.2023. https://electrosome.com/interfacing-lcd-atmega32-microcontroller-atmel-studio/

Last Minute Engineers. Control DC Motors with L293D Motor Driver IC & Arduino. Viitattu 12.3.2023. https://lastminuteengineers.com/l293d-dc-motor-arduino-tutorial/

McCallum, K. 2021. Does Hitting the Snooze Button Help or Hurt? Viitattu 14.5.2023. https://www.houstonmethodist.org/blog/articles/2021/dec/does-hitting-the-snooze-button-help-or-hurt/

SM. 2023. Arduino as ISP and Arduino Bootloaders. Viitattu 17.5.2023. https://docs.arduino.cc/built-in-examples/arduino-isp/ArduinoISP

Syvänne, J. 2020. BIOPOHJAISET JA BIOHAJOAVAT MUOVIT. Viitattu 20.4.2023. https://www.muoviyhdistys.fi/2020/03/03/biopohjaiset-ja-biohajoavat-muovit/

Tech Target. 2019. LCD (Liquid Crystal Display). Viitattu 12.2.2023. https://www.techtarget.com/whatis/definition/LCD-liquid-crystal-display

Texas Instruments. 2003. µA7800 SERIES POSITIVE-VOLTAGE REGULATORS. Viitattu 30.4.2023. https://www.sparkfun.com/datasheets/Components/LM7805.pdf

Texas Instruments. 2016. L293x Quadruple Half-H Drivers. Viitattu 14.4.2023. https://www.ti.com/lit/ds/symlink/l293d.pdf?ts=1684557149212&ref_url=https%253A%252F%252Fwww.ti.com%252Fproduct%252FL293D