Tallinna Ülikool

Digitehnoloogiate instituut

KODULOOMA TOITMISE AUTOMATISEERIMINE RASPBERRY PI ABIL

Bakalaureusetöö

Autor: Sander Peerna

Juhendaja: Jaagup Kippar

Autor: ……..…."……"2017

Juhendaja: ……."……"2017

Instituudi director: ……"……"2017

Tallinn 2017

Autorideklaratsioon

Deklareerin, et käesolev bakalaureusetöö on minu töö tulemus ja seda ei ole kellegi teise poolt varem kaitsmisele eistatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on viidatud.

....

(kuupäev) (autor)

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina _____ (sünnikuupäev: _____)

(*autori nimi*)

1. annan Tallinna Ülikoolile tasuta loa (lihtlitsentsi) enda loodud teose
   _____
   _____
   _____

   (*lõputöö pealkiri*)

   mille juhendaja on _____,

   (*juhendaja nimi*)

   säilitamiseks ja üldsusele kättesaadavaks tegemiseks Tallinna Ülikooli Akadeemilise Raamatukogu repositooriumis.
2. olen teadlik, et punktis 1 nimetatud õigused jäävad alles ka autorile.
3. kinnitan, et lihtlitsentsi andmisega ei rikuta teiste isikute intellektuaalomandi ega isikuandmete kaitse seadusest tulenevaid õigusi.

Tallinnas/Haapsalus/Rakveres/Helsingis, _____

(digitaalne) allkiri ja kuupäev

Sisukord

Mõisted 5
Sissejuhatus 6
1. Lemmiklooma toitmise automatiseerimine 7
1.1. Lemmiklooma toitmise automatiseerimise vajalikkus 7
1.2. Automatiseerimisel kasutatavad seadmed 7
2. Olemasolevate lahenduste tutvustus 9
2.1. Petnet 9
2.2. Raspberry Pi Power Cat Feeder 9
2.3. Internet Enabled Raspberry Pi Pet Feeder 10
3. Lemmiklooma toitmise automatiseermise protsess 11
3.1. Kasutatud seadmed 11
3.2. Seadme loomine 11
3.3. Programmi loomise portsess 17
3.4 Seadme testimine 21
Kokkuvõte 23
Kasutatud kirjandus 24
Summary 26
LISAD 27
Lisa 1 28
Lisa 2 30

Mõisted

LCD – Vedelkristallkuvar on kuvar mille talitlus põhineb vedelkristallides ilmnevatel elektroopilistel nähtustel

GPIO – Üldise otstarbega sisend-väljund nõel, mille sisendit kasutatakse andmete saatmiseks näiteks miniarvutisse ning väljundit andmete edastamiseks järgmisesse seadmesse.

Sissejuhatus

Antud bakalaureusetöö eesmärgiks on luua aparaat, mis automatiseeriks lemmiklooma toitmise protsessi ja koostada materjal, mida järgides on võimalik luua sarnane abivahend ka oma koduloomale. Teema valik põhineb autori huvil elektroonika, miniarvutite ja automatiseerimise vastu. Samuti mängis teema valikul rolli autori hajameelsus, mille tõttu on olnud olukordi kus autor unustas õigeaegselt oma kassi toita.

Käesolev töö on suunatud inimestele, kellel on olnud sarnased juhtumeid nagu autoril ja sooviksid oma lemmikloomadele teene teha ja toita neid alati õigeaegselt. Loomulikult on töö sobilik neile, kellel on huvi automaatika vastu ja tahaksid ehitada midagi teistsugust ning huvitavat.

Töö jaotub kolmeks peatükiks. Esimeses peatükis kirjutatakse üldiselt sellest, kuidas looma toitmist automatiseerida. Räägitakse millised seadmeid on vaja aparaadi loomisel.

Teises peatükis tutvustatakse põgusalt olemas olevaid lahendusi ja kirjeldatakse kuidas antud tööde autorid oma probleemi lahendasid. Samuti uuritakse ka valmis tooteid, mida on võimalik poest osta.

Kolmandas peatükis on materjal, mille abil on võimalik ka ise oma lemmiklooma toitmist automatiseerida. Kirjeldatakse kogu ehitamise protsessi kasutades abivahenditena pilte, skeeme ja programmikoodi, mis loodi aparaadi töötamiseks.

1. Lemmiklooma toitmise automatiseerimine

Lemmiklooma toitmist on võimalik automatiseerida valmisolevate lahenduste abil, kuid üldiselt on antud valikud liiga kallid, et neid endale lubada. Tänu sellele, et viimasel ajal on väga populaarseks muutunud erinevad nii öelda krediitkaardiarvutid, on võimalik luua samaväärne abivahend palju soodsamalt ja vajadusel teha midagi, mis on suuteline tegema palju rohkemat kui valmis tooted.

1.1. Lemmiklooma toitmise automatiseerimise vajalikkus

Vajadus seadme järgi, mis automatiseeriks lemmiklooma toitmist, on inimestel, kes on sarnased autorile ja unustavad aegajalt toida oma lemmikloomi. Kasulik on ta ka neile, kes peavad aeg-ajalt pikemat aega kodust eemal olema. Kasutades antud seaded ei pea kodust lahkudes söögi kausse üleääre täis panema, vaid saab rahulikult olla kodust eemal, teades et seada toidab alati õigeaegselt kodus ootavat kodulooma.

1.2. Automatiseerimisel kasutatavad seadmed

Lemmiklooma toitmiseks loodava aparaadi jaoks on vaja erinevaid seadmeid. Esiteks on vajalik seade, mis juhiks kogu tööd. Antud juhul on kasutusele võetud Raspberry Pi 2. Kuna tegemist on automaatse toitjaga, siis on vaja anumat, mis oleks võimeline hoiustama toitu. Vaja on ka erinevaid väiksemaid elektroonikakomponente näiteks: mootor, nupud ja käesolevas apraadis ka 16*2 LCD ekraan.

Raspberry Pi 2-te kasutame, sest see on küllaltki odav lahendus võrreldes täiskomplektse arvutiga. Selle abil on lihtne kätte saada täpsed kellajad ja määratleda need kellaajad, mil seade peaks töötama. Samuti on Raspberry Pi hea kuna sellel on GPIO (Raspberry Pi Foundation, kuupäev puudub), mida on lihtne juhtida kasutades programeerimiskeelt Python. Küll aga ei kasuta autor Raspberry Pi-d, et ühendada aparaati internetiga, kuna autor on arvamusel, et kõik asjad siiski ei peaks internetiga ühenduses olema.

Elektroonikakomponentitest on kasutusel 16*2 LCD ekraan, mis kuvab kellaaega ja järgimist toitmise aega. Samuti kasutatakse LCD ekraani, et muuta sätteid. Sätete muutmiseks on kasutusel kaks nuppu, millega on võimalik muuta aega, millal seade väljastab toitu.

2. Olemasolevate lahenduste tutvustus

Antud peatükis tutvustatakse olemas olevaid lahendus, mis on masstootmises ning ka lahendusi, mida inimesed on loonud oma loomade toitmiseks.

2.1. Petnet

Pealkiri	Petnet
Autor	Petnet Inc.
Aadress	http://www.petnet.io/

Antud seadme puhul on tegemist valmistootega, millel on kasutusel ka oma mobiilne rakendus. Tänu internetiühendusele ja rakendusele on aparaat võimeline teavitama millal väljastati toitu ja samuti lihtsustab see ka seadistamist. Rakendusest on võimalik märkida oma lemmiklooma vanus, kaal ja ka aktiivsus, mille järgi seade omakorda hakkab väljastama täpselt nii palju toitu kui vaja. Seade suudab mahutada ligikaudu 3kg toitu. Aparaat töötab kas läbi seina adapteri või kui on hädaolukord, näiteks elektrikatkestus, siis on olemas aku.

2.2. Raspberry Pi Power Cat Feeder

Pealkiri	Raspberry Pi Power Cat Feeder
Autor	David Bryan
Aadress	http://drstrangelove.net/2013/12/raspberry-pi-power-cat-feeder-updates/

Käesoleva seadme puhul on tegemist Raspberry Pi baasil loodud aparaadiga. Seadme jaoks on modifitseeritud kuivtoidu anumat, mille küljes on rullik, mida keerates saab väljastada anumas hoitud sööki. Automatiseerimise käigus on rulliku külge monteeritud servomootor, mis määratud kellaaegadel liigub, et anumast toitu väljastada. Samuti on aparaadil võimalus teavitada e-maili abil, millal väljastati toitu.

2.3. Internet Enabled Raspberry Pi Pet Feeder

Pealkiri	Internet Enabled Raspberry Pi Pet Feeder
Autor	krishsdoit
Aadress	http://www.instructables.com/id/Internet-Enabled-Raspberry-Pi-Pet-Feeder/

Käesoleva seadme puhul on samuti tegemist Raspberry Pi baasil loodud seadmega, kuid selle esineb eelnevaga mõningad erinevused. Anumana on kasutatud sarnast lahendust eelmise seadega, kuid mootorina on kasutatud servomootori asemel võimsamat 12 voltilist alalisvoolumootorit, mis omakorda tähendab, et mootorit pole võimalik vaid Raspberry Pi-ga tööle panna. Selle jaoks on seadmes kasutusel võetud 12-voldist toiteplokk, mis suudaks antud mootorit liigutada. Toitmiseks on seadmel üks nupp, mida vajutades aparaat väljastab toidu. Peale nupu vajutust ei luba seada järgmised 8 tundi enam toitu väljastada. Aparaadil on ka 16*2 LCD ekraan, mis kuvab hetke kellaaega, aega järgmise korrani, mil saab toitu väljastada ja kui puudub luba toidu väljastamiseks, teavitatakse kasutajat sellest LCD ekraani kaudu. Sarnaselt eelnevale lahendusele, on ka antud varjandis võimalik e-maili teel näha millal viimati toitu väljastati ja lisaks sellel on võimalik e-maili saates anda seadmele käsklus toidu väljastamiseks.

3. Lemmiklooma toitmise automatiseermise protsess

Töö automatiseerimise protsess koosneb üldiselt kahest osast: aparaadi loomise osa ja programmi loomise osa. Aparaadi loomise osas on juttu erinevate seadmete kasutamise põhjustest ja ka selle ehitamisest ning programmi loomise osas kirjeldatakse lahti, mida erinevad osad programmikoodist teevad.

3.1. Kasutatud seadmed

Seadmete valikutel lähtuti enamasti sellest, mis parasjagu kättesaadav oli, küll aga mõne asja puhul ka otsiti välja konkreetne seade mida kasutati aparaadi loomisel. Järgnevad on seadmed mida aparaadi ehitamisel kasutati:

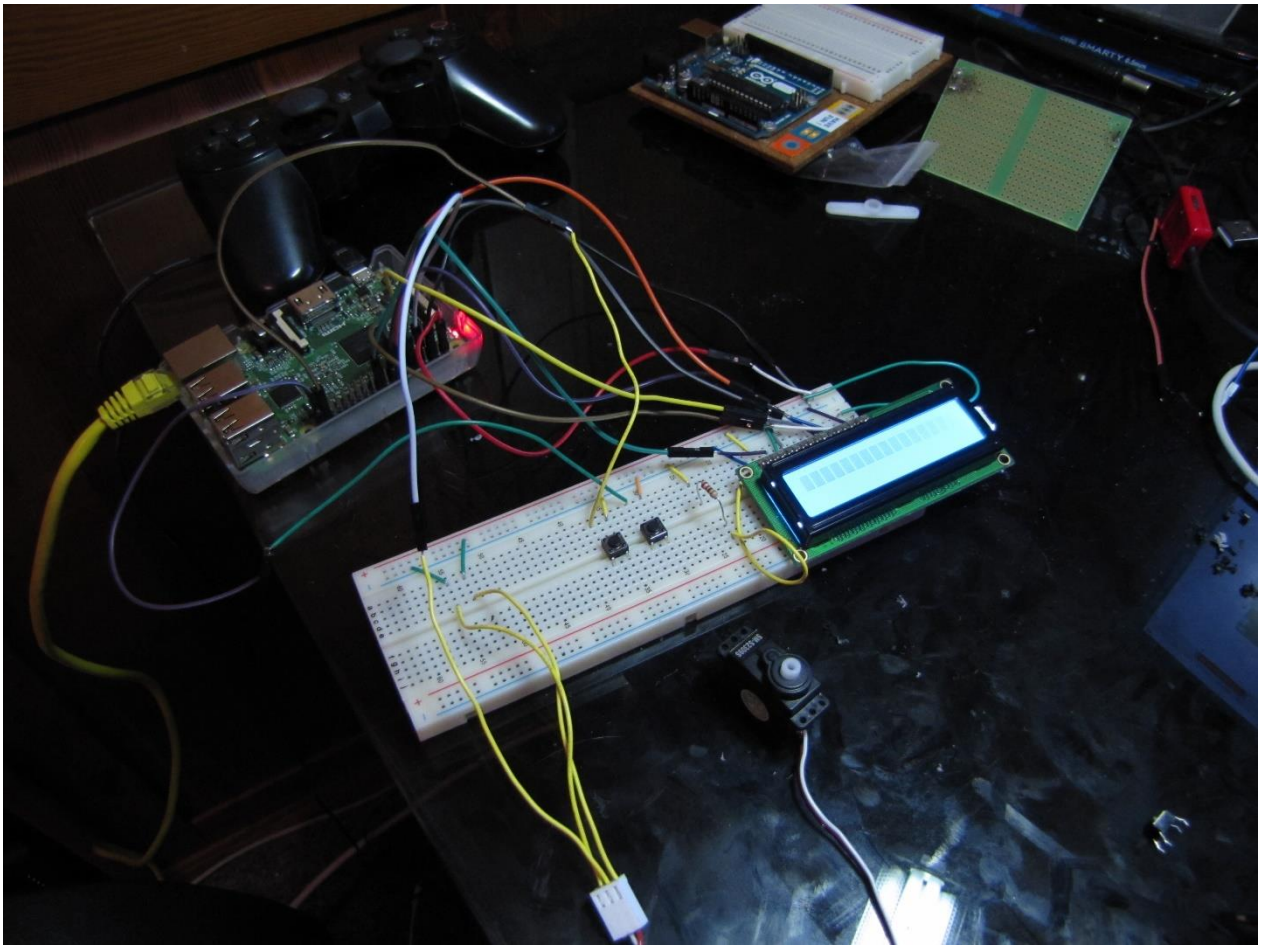
- Raspberry Pi 2 miniarvuti, mida eelkõige kasutati kuna antud seade oli juba autori olemas ja antud juhul polnud see kasutuses. Antud miniarvuti on kasutuses ka põhjusel, et erinevalt arduino lahendustest, on Raspberry Pi-ga võimalik lihtsalt kellaajast sõltuvaid funktsioone kasutada. Samuti annab Raspberry Pi võimaluse aparaati edasi arendada tulevikus, näiteks luua sellele internetil põhinevaid funktsionaalsusi (Raspberry Pi Foundation, 2015).
- SM-S2309S servomootor, mille pöördemoment on pingel 4.8V 1,1kg*cm, on kasutusel, et liigutada anumas olevat rullikut toidu väljastamiseks (SM-S2309S, kuupäev puudub).
- Kaks lülitit, mille eesmärgiks on toidu väljastamisaja muutmine.
- 16*2 LCD ekraan, mis kuvab kellaaega ja toitmise aega.

3.2. Seadme loomine

Käesoleva töö käigus loodi olemasolevast anumast aparaat, mis on võimeline automaatselt kindlal ajal toitu väljastama. Seadme loomine algas kõigepealt prototüübi loomisest, mida katsetati kõigepealt ilma seda anumaga ühendamist. Seejärel kohandati anumat, et sinna oleks võimalik loodud lahendus külge panna ja lõpuks pandi kõik kokku.

Esiteks loodi aparaadi prototüüp. Selle jaoks kasutas autor Raspberry Pi-d, kuhu külge ühendati 16*2 LCD ekraan, kaks nuppu ja servomootor. Algselt oli plaanis kasutada tavalist 9 voldist mikromootorit, kuid hiljem autor otsustas, et lihtsam oleks kasutada servomootorit, kuna seda on

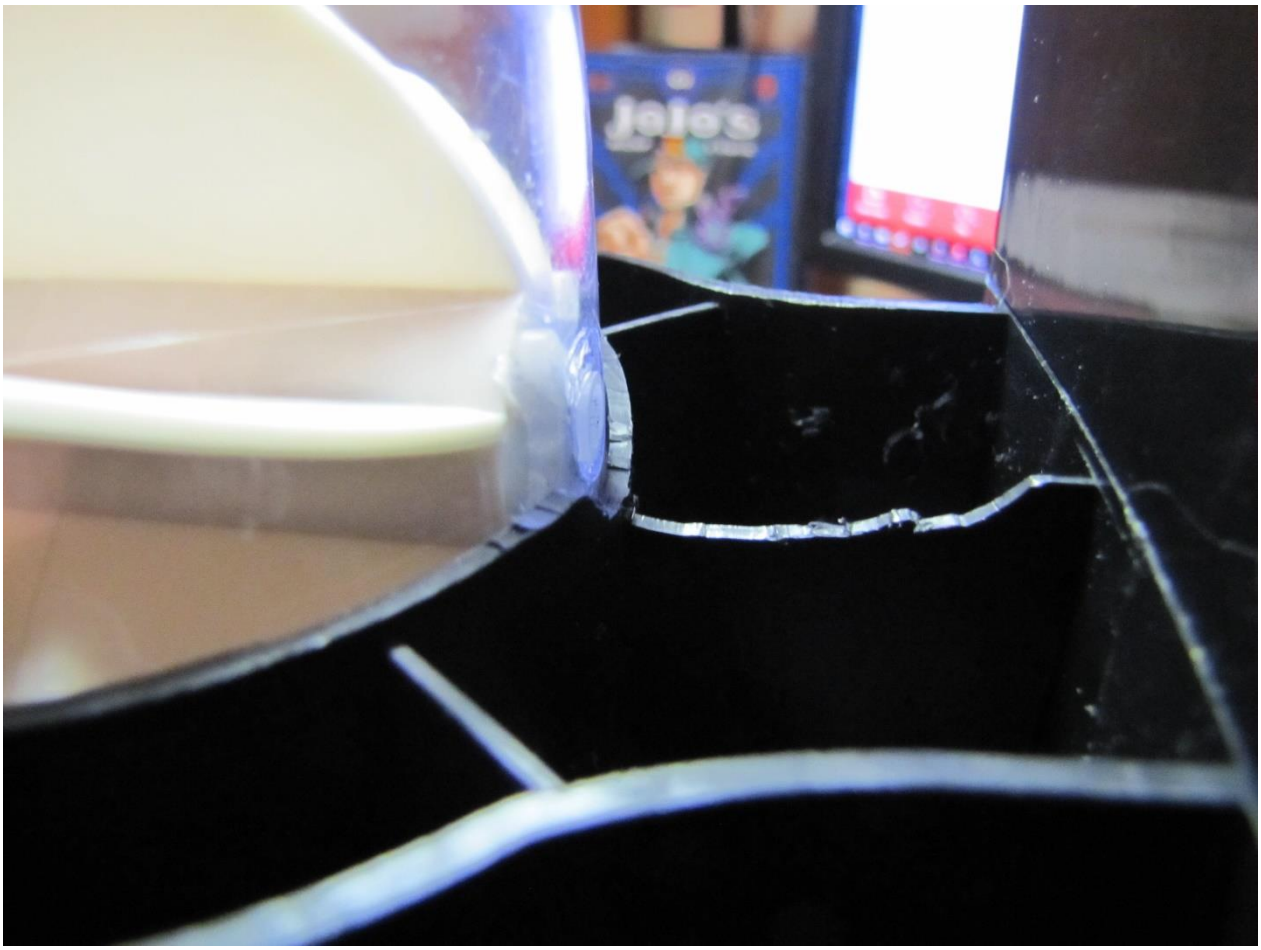
võimalik kasutada ka Raspberry Pi pealt tuleva toitega. Valmis prototüübi (Joonis 1) testimisega ilmnes vajadus kohandada anumat, kuna ilma anuma kohandamiseta polnud võimalik servomootorit kuidagi ühendada rulliku külge, mis liigutab anumas olevat toitu (Joonis 2).



Joonis 1 Prototüüp

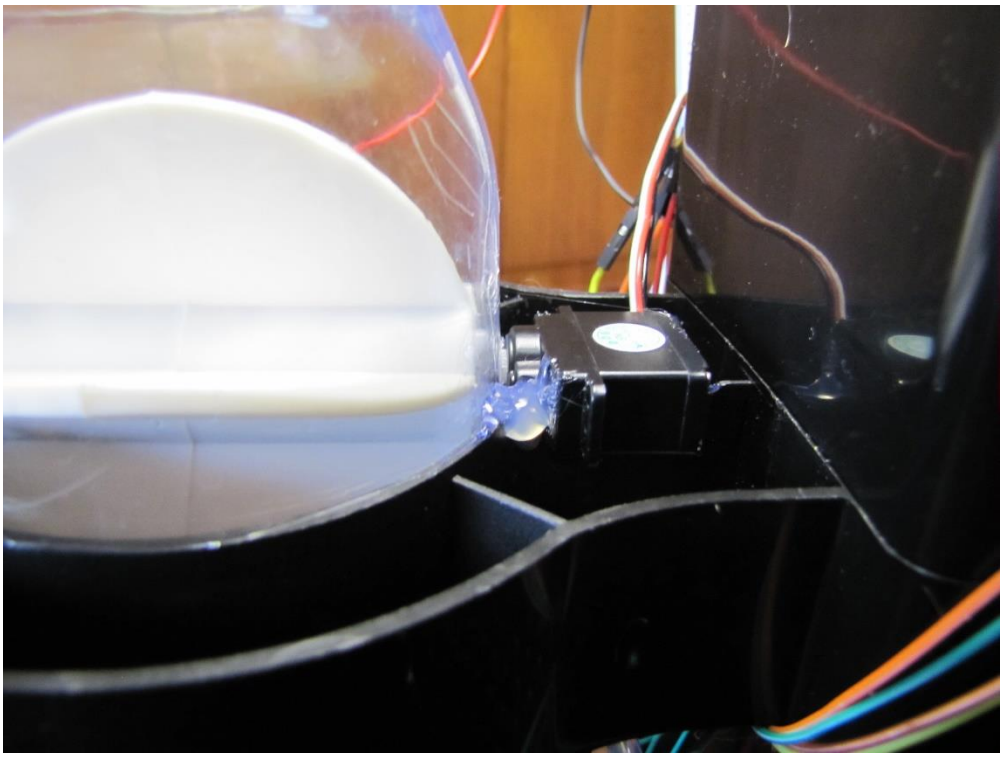
Joonis 2 Algne anum

Selleks, et seadet oleks võimalik anuma külge monteerida, oli vaja teha muudatusi anumale. Esiteks tuli leida viis kuidas oleks võimalik liigutada servomootoriga anumas olevat rullikut. Probleemi lahendamiseks proovis autor erinevad võimalusi. Esimesena proovis autor lihtsalt servomootorit kinnitada rulliku küljes oleva kraani külge. Kuigi antud variant töötas, leidis autor, et antud lahendus pole piisavalt elegantne ja samuti oleks seda võimalik kogemata ära lõhkuda. Järgmisena tahtis autor anumalt kraani ära võtta ja panna mootorit tagaküljele, et seda poleks eest üldse näha. Selle saavutamiseks tuli võtta rulliku küljest kraan ära ja seejärel keerata anuma sees olev rull tagurpidi, et rulliku sees olevat auku oleks võimalik koos servomootoriga kasutada. Rulliku vahele pidi autor ka lisama hammasratta laadse detaili, mille külge oleks võimalik kinnitada servomootor. Selleks, et mootorit saaks kinnitada tagaküljele, pidi autor korpuse sisse lõikama ava, et mootor mahuks täpselt sisse (Joonis 3).

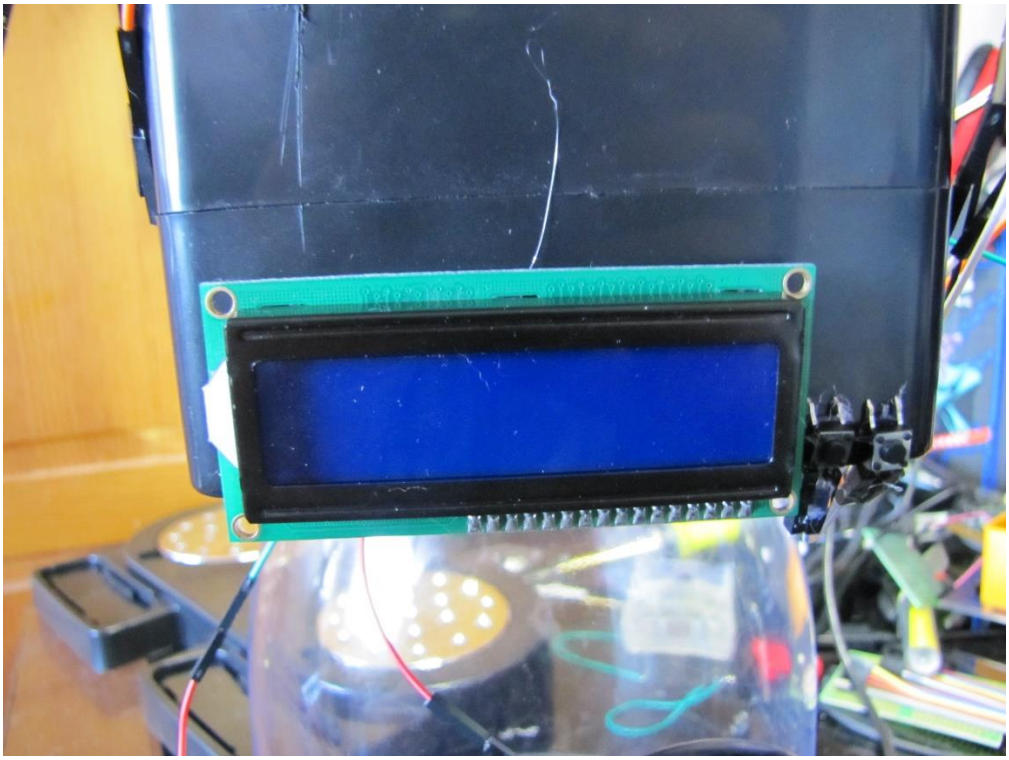


Joonis 3 Korpuse lõikus

Peale korpuse lõikamist kinnitati mootor korpuse külge. Selleks kasutas autor kuumaliimi püstolit, et mootor kõvasti korpuse külge liimida. Liimimise juures oli oluline, et see püsiks, kuna mootori pidi kinnitama alaspidi (Joonis 4). Korpuse tagumisele küljele kinnitati ka kaks nuppu ja LCD ekraan. Algselt oli autoril plaan kinnitada nii ekraan kui ka nupud seadme eesmisele küljele, kuid selle saavutamine osutus keerulisemaks, kui autor oleks oodanud. Samuti tekkis probleem juhtmete pikkusega. Kui ekraan ja nupud oleks ette kinnitatud, siis poleks juhtmed kuidagi jõudnud Raspberry Pi-ni. Seega otsustas autor kinnitada ekraani ja nupud anuma tagumisele küljele. Kuna need on tagaküljel, jääb esikülg palju viisakama välimusega, mis on autori arvates piisavalt hea kompromiss mugavuse ja hea välimuse vahel. Kuigi eesolevate nuppudega oleks mugavam seadistada, näeks aparaat visuaalselt halvem välja (Joonis 5).

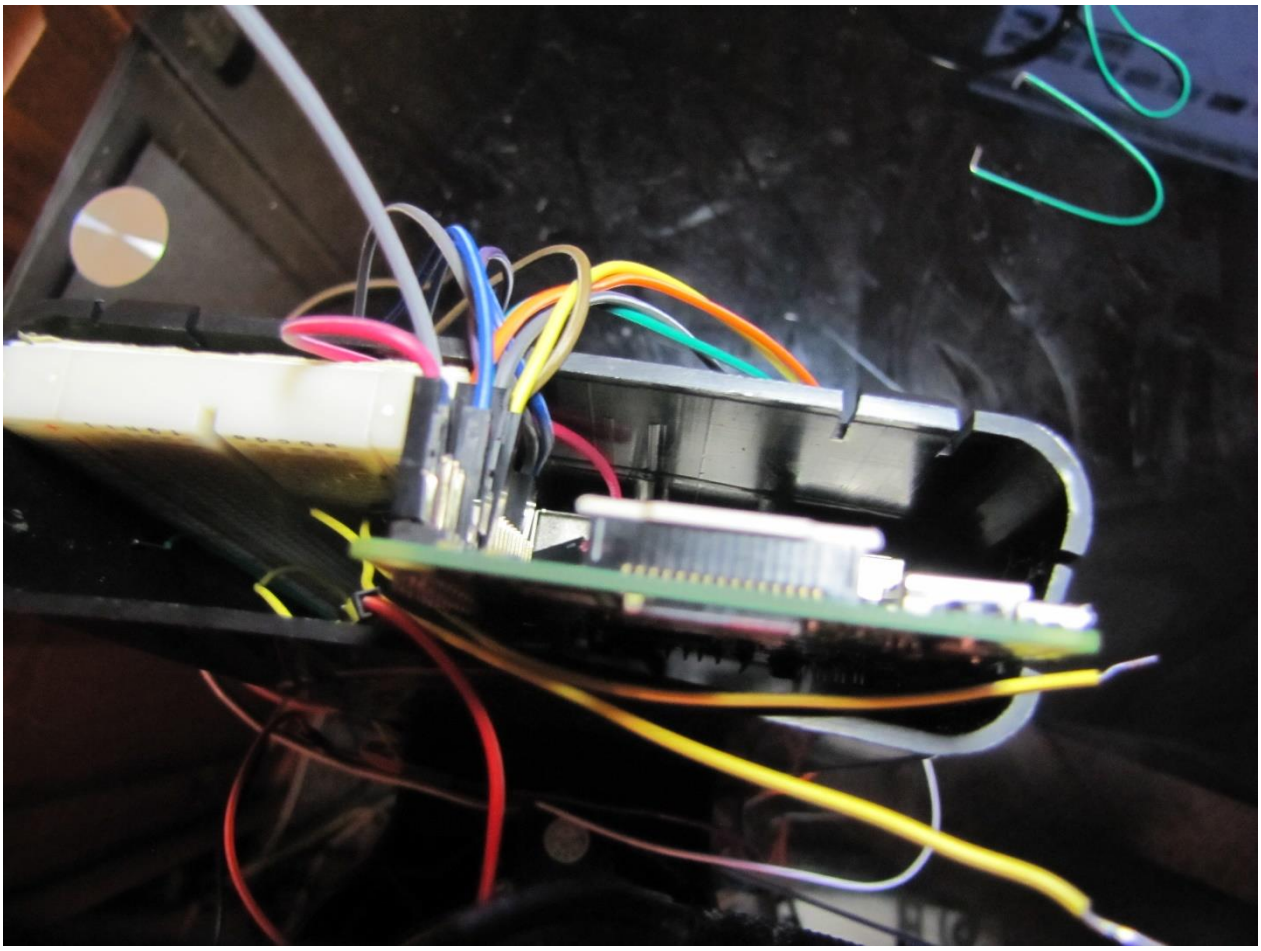


Joonis 4 Servomootor



Joonis 5 LCD ekraan ja nupud

Kui servomootor, LCD ekraan ja nupud paigas, oli vaja leida viis, kuidas Raspberry Pi seadme külge paigutada. Algselt oli autoril plaan miniarvuti lihsalt kinnitada korpuse tagumisele küljele, kuid autor leidis, et antud meetodiga oleks miniarvuti liiga avatuks jäänud, mis omakorda oleks tähendanud, et seda oleks olnud võimalik kogemata kahjustada. Probleemi lahendamiseks otsustas autor ära kasutada korpuse sees olevat õõnsust. Katsetamise käigus selgus, et õõnsuses sisse mahu väga täpselt Raspberry Pi koos maketeerimislauaga. Kui miniarvuti oli sisse mahutatud, siis oli aeg kõik uuesti kokku panna ja sellega oli seadme loomise osa valmis (Joonis 6).



Joonis 6 Raspberry Pi õõnsuse sees

3.3. Programmi loomise portsess

Kuna seadme tarkvara töötab Raspberry Pi pealt, siis otsustas autor kirjutada programmi Python-i keeles. Kogu töö automatiseerimiseks kirjutas autor ühe Python-i programmi, mis juhib LCD ekraani, nuppe kui ka mootorit.

Kõige pealt tuleb programmi alguses importida mõningad teegid (Koodinäide 1).

```
import RPi.GPIO as GPIO
import datetime
import time
import Adafruit_CharLCD as LCD
```

Koodinäide 1: Programmis kasutatavad teegid

Esimese reaga importitakse teek RPi.GPIO ja seejärel määratakse, et antud teeki oleks võimalik välja kutsuda GPIO nime all. Teek RPi.GPIO annab Python-le kontrolli Raspberry Pi GPIO üle (B. Croston, 2016). Järgmise reaga importitakse teek datetime, millega on võimalik kätte saada kellaaeg (Python Software Foundation, 2017). Teeki time kasutame, et teha arvutusi kellajaga (Python Software Foundation, 2017). Teegiga Adafruit_CharLCD on võimalik saata teksti seadmes kasutusel olevale 16*2 LCD ekraanile (Adafruit, 2016).

Järgmisena peab ära määrama milliste GPIO nõelade külge LCD ekraan ühendatud on. Selle tarvis tuleb iga ühenduse kohta luua üks muutuja mis määrab ära nõela numbri. Seejärel luuakse üks ühtne lcd muutuja, mis võtab kõik määratud nõela numbrid kokku (Koodinäide 2).

```
lcd_rs = 25
lcd_en = 24
lcd_d4 = 23
lcd_d5 = 17
lcd_d6 = 18
lcd_d7 = 22
lcd_backlight = 2
```

```
lcd_columns = 16
lcd_rows = 2


lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6,
lcd_d7, lcd_columns, lcd_rows, lcd_backlight)
```

Koodinäide 2: LCD ekraani seadistamine

Esimese reaga määrame 25-ndale nõelale *register select*-i. Sellega on võimalik LCD-le saata käsklusi, et muuta kursori positsiooni või näiteks kustutada ekraanilt tekst. Samuti on sellel võimalus saata andmeid ekraanile. Järgnev rida määrab loa, et oleks võimalik registrisse andmed kirjutada. Järgneva nelja reaga määrame, et kasutame LCD ekraani 4-bitises olekus. LCD_columns määrab LCD ekraanil veergude arvu ja lcd_rows määrab ekraanil ridade arvu. Lõpetuseks võetakse kõik määratud nõelte positsioonid kokku üheks muutujaks, et hiljem LCD-ga tegeledes oleks lihtsam kasutada LCD ekraani (M. Sklar, 2015).

Järgnevalt määrame GPIO nõeltele kaks kasutuses olevat nuppu ja servomootori, ning seadistame servomootori (Koodinäide 3).

```
GPIO.setup(21, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(20, GPIO.IN, pull_up_down = GPIO.PUD_UP)


GPIO.setup(4, GPIO.OUT)
P = GPIO.PWM(4, 50)
p.start(7.0)
```

Koodinäide 3: Nuppude ja servomootori seadistamine

Esimese kahe reaga määrame ära mõlemad nupud. Selle tarvis määrame esiteks nõela numbri, siis määrame, et nupp on sisend ehk annab andmeid sisse ja viimasena määrame, et nupu vajutuse peale tagastatakse väärtus `False` (viide). Servomootori seadistamiseks tuleb kõigepealt määrata GPIO nõela number ja määrata, et antu nõel on väljund, kuna mootor ei saada mingisuguseid andmeid tagasi. Järgmise reaga loome muutujua `p` ja määrame, et nõel 4 on PWM ja määrame PWM-i sageduseks 50 hertsi. Viimane rida määrab servomootori algpositsiooni, milleks siin on neutraalne positsioon (G. MacDonald, 2013).

Järgnevalt tuleb luua muutujad kellaegadel, mil servomootor peab tegema oma tööd (Koodinäide 4).

```
feedTimeMem = datetime.datetime(2001, 1, 1, 18, 30, 0)
feedTimeMem2 = datetime.datetime(2001, 1, 1, 18, 30, 1)
feedTime = feedTimeMem.time()
feedTime2 = feedTimeMem2.time()
```

Koodinäide 4: Toitmisaja muutujad

Põhjuseks miks nii *feedTimeMem* kui ka *feedTime* kaks tükki on seisneb selles, et mootor peab töötama nende kahe aja vahemikus. Esimest muutujat kasutame kuna Python-s on võimalik kellaaegu muuta ainult siis, kui kellaeg on koos kuupäevaga, seega kasutame esimest muutujat põhimõtteliselt mäluna. Järgmisena võtame sellest ära kuupäeva ja siis jääb alles ainult kellaeg, selletõttu võib ka muutujas *feedTimeMem* olla täiesti suvaline kuupäev, kuna see ei sega programmi töötamist.

Järgmisena loome tsükkli, kus toimub programmi põhi töö (Koodinäide 5).

```
while True:
    up_button = GPIO.input(21)
    down_button = GPIO.input(20)
    rn = datetime.datetime.now().strftime("%H:%M:%S")
    feedt = feedTimeMem.strftime("%H:%M:%S")
    lcd.message("    "+feedt+"\n    "+rn)
    time.sleep(1.0)
    lcd.clear()
```

Koodinäide 5: Tsükkli algus

Tsükkli alguses määrame kohe ära kaks nuppu, mida hiljem kasutame. Järgnevalt loome muutuja, mis saab kätte hetke kellaaja. Muutujat *feedt* kasutame, et kuvada LCD ekraanil aega, mil seade väljastab anumast toitu. Järgnev rida kirjutab LCD ekraani esimesele reale aja, mil väljastatakse

toit ja teisele reale hetke kellaaja. Lõpetuseks ootab programm alati ühe millisekundi, et ekraani uuendada ja see järel teeb alati ekraani tühjaks.

Seadme töötamiseks tuleb luua tsükkli sisse ka `if` lause, mille korral servomootor hakkab oma tööd tegema (Koodinäide 6).

```
If     datetime.datetime.now().time()     >     feedTime     and
datetime.datetime.now().time() < feedTime2:
    p.ChangeDutyCycle(7.0)
    time.sleep(1)
    p.ChangeDutyCycle(11.5)
    time.sleep(1)
    p.ChangeDutyCycle(2.0)
    time.sleep(1)
    p.ChangeDutyCycle(11.5)
    time.sleep(1)
    p.ChangeDutyCycle(7.0)
    time.sleep(1)
```

Koodinäide 6: Toitmise funktsioon

Selleks, et `if` lause töötaks peab olema kaks olukorda täidetud- esiteks peab olema hetke aeg suurem kui toitmis aja algusaeg ja hetke aeg peab olema väiksem kui toitmis aja lõppaeg. Kui need kaks olukorda on täidetud, liigub mootor kõigepealt algpositsioonile, kus see peaks ka juba olema, seejärel liigub ühele poole maksimumpositsiooni ja teisele poole maksimum positsiooni, nii kaks korda, et rullik anumas saaks võimalikult palju liikuda.

Järgnevalt on vaja luua ka funktsioonid nuppudele. Selle tarvis kasutame taaskord `if` lauset (Koodinäide 7).

```
if up_button == False:
    feedTimeMem = feedTimeMem + datetime.timedelta(0, 600)
    feedTimeMem2 = feedTimeMem + datetime.timedelta(0, 600)
    feedTime = feedTimeMem.time()
```

```
    feedTime2 = feedTimeMem2.time()
    lcd.message('Set to: '+feedTime.strftime("%H:%M:%S"))
```

Koodinäide 7: Nupu vajutus funktsioon

Kui `up_button` saab signaali `False` ehk kui nupp on alla vajutatud, lisatakse `feedTimeMem`-le 10 minutit juurde ja täpselt sama tehakse ka muutujale `feedTimeMem2`. Seejärel muudetakse ära muutujad `feedTime` ja `feedTime2` ning siis kuvatakse ekraanil, et muudeti toitmisaega. Sarnane funktsionaalsus tuleb luua ka teisele nupule, aga sellisel juhul tuleb ajale juurde liitmise asemel ajalt maha lahutada.

Selleks, et programm tööle läheks kui Raspberry Pi ühendatakse vooluga, pidi autor veel ka lisama konfiguratsiooni faili viite antud programmi failile. Selleks tuli navigeerida Raspberry Pi peal kausta `~/.config/lxsession/LXDE-pi`, seejärel avad fail `autostart` ja sinna faili lisada uuele reale rida `@python ~/Documents/PetFeeder.py`.

3.4 Seadme testimine

Seadme testimine toimus kahes etapis. Esiteks testiti seadet prototüübina, sest kui seadet oleks testitud alles siis, kui kõik oli juba valmis ehitatud, oleks olnud keerulisem probleeme lahendada. Algse prototüübi testimise juures ilmnes mõningaid probleeme. Esiteks ei olnud algselt kasutuses olev servomootor piisavalt võimas, et liigutada anumas olevat rullikut, seega pidi autor vahetama servomootori välja. Prototüübi testimisel üldiselt peale mõne programeerimisvea probleeme ei ilmnenud, küll aga kui tuli aeg, mil oli vaja prototüübist saada reaalne seade, tekkis probleeme sobiva anuma leidmisega. Algselt oli autor arvamusel, et anumat oleks võimalik kodus olemas olevatest asjad valmis ehitada, kuid kahjuks aja puuduse tõttu pidi autor leppima sellega, et tellida tuleb valmis olev anum, millel oleks võimalik prototüübina valminud seada juurde lisada. Valmis seadme testimise käigu selgus, et piisavalt toitu anumast välja saada, peaks programmikoodi nii palju muutma, et servomootor teeks vähemalt kaks tsüklit veel, vastasel juhul ei tulnud anumast piisavalt toitu välja, et toita autori kassi. Seadme testimisel oli algselt seatud toitmise ajaks 18:30 ja peale igat toitmist muudeti aega kümne minuti võrra kaugemale, testimaks kas aparaat järgneval päeval ka väljastab toitu eelnevast päevast 10 minutit hiljem. Antud funktsionaalsus töötas täpselt nii nagu autor oli ette näinud, seega kui välja arvata probleem toidu väljastamisvähesusega, siis

töötas seada vastavalt ette nähtud visioonile. Testimise käigus ilmnes ka probleem, et autori kassile tundusid huvi pakkuvat juhtmed, mis suundusid seadme taha ja sealt omakorda korpuse sisse, seega autor arvab, et parem oleks, kui juhtmeid saaks võimalikult kiiresti korpuse sisse juhtida. Kahjuks puudusid autoril võimalused korpuse sisse augu lõikamiseks, et luua elegantsem lahendus juhtmete peitmiseks korpuse sisse.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli automatiseerida lemmiklooma toitmise protsessi. Eesmärgi saavutamiseks valmistas autor Raspberry Pi-st seadme, mille külge oli ühendatud ekraan, nupud ja mootor, mis suudaks anumas olevat rullikut iseseisvalt liigutada. Selleks, et seada ka töötaks, pidi autor kirjutama tarkvara, mille tarvis kasutati Python-i programmeerimiskeelt. Lisaks seadme ehitamisele läbiti seadmele ka lühike testimisperiood, mille käigus uuriti kas seade töötab vastavalt nõuetele.

Töö on jaotatud kolme ossa. Esimese osas on juttu üldisest lemmiklooma toitmise automatiseerimisest, sellest miks see vajalik on ja millised seadmed on vaja, et seda ise teha. Teises osas on põgusalt juttu teiste autorite poolt loodud lahendustest. Kolmas osa on autori poolt loodud seadme kirjeldamine, nii seadme ehitamise osas kui ka tarkvara loomise osas.

Seadme loomisel kasutati valmis olevat anumat, mille küljes oli kraan,mida keerates on võimalik anumast toitu väljastada. Anumat muudeti, et oleks võimalik anuma külge ühendada esiteks mootor ja teiseks kinnitada ekraan ning nupud. Samuti tuli leida viis kuidas miniarvuti anumasse peita.

Tarkvara loomise osas kirjeldati, miks on midagi programmis kirjutatud nii nagu on. Programmiread on üksikasjalikult lahti seletatud, et ka inimesed, kes on vähem programeerinud suudaks jälgida miks on antud lahendust kasutatud.

Seadme testimise osas prooviti seadet kasutada algselt prototüübi faasis, et võimalikult vara üles leida suuremad vead. Valmis seadet testiti paari päeva jooksul, muutes toitmise aegu, et näha kas seada töötaks õigesti paari päeva jooksul.

Töö tulemusena sai autori arvates töö eesmärk saavutatud, kuid autor on arvamusel, et ta oleks võinud tööd võtta tõsisemalt ja teha midagi paremat kui lõpptulemusena välja tuli. Seadme loomine andis autorile palju uusi kogemusi, kuna eelnevalt antud tööle polnud autoril väga palju kokkupuudet automatiseerimisega olnud. Samuti andis töö kirjutamine autorile teada, et autor peab endiselt õppima aega paremini planeerima.

Kasutatud kirjandus

Adafruit. (2016). *Adafruit_Python_CharLCD.* Loetud 25. aprill 2017 aadressil

https://github.com/adafruit/Adafruit_Python_CharLCD

Bryan, D. (2013). *Raspberry Pi Power Cat Feeder.* Loetud 23. aprill 2017 aadressil

http://drstrangelove.net/2013/12/raspberry-pi-power-cat-feeder-updates/

Croston, B. (2016). *Raspberry Pi GPIO.* Loetud 28. aprill 2017 aadressil

https://pypi.python.org/pypi/RPi.GPIO

krishsdoit. *Internet Enabled Raspberry Pi Pet Feeder.* Loetud 23. aprill 2017 aadressil

http://www.instructables.com/id/Internet-Enabled-Raspberry-Pi-Pet-Feeder/

MacDonald, G. (2013). *Servo Control with the Raspberry Pi.* Loetud 25. aprill 2017 aadressil

https://www.youtube.com/watch?v=ddlDgUymbxc

Petnet Inc. *Petnet.* Loetud 26. aprill 2017 aadressil

http://www.petnet.io/

Python Software Foundation. (2017). *Python datetime.* Loetud 27. aprill 2017 aadressil

https://docs.python.org/2/library/datetime.html

Python Software Foundation. (2017). *Python time.* Loetud 27. aprill 2017 aadressil

https://docs.python.org/2/library/time.html

Raspberry Pi Foundation. (2015). *Raspberry Pi 2 Model B.* Loetud 28 aprill 2017 aadressil

https://www.raspberrypi.org/products/raspberry-pi-2-model-b/

Raspberry Pi Foundation. *An introduction to GPIO and physical computing on the Raspberry Pi.* Loetud 28. aprill 2017 aadressil

https://www.raspberrypi.org/documentation/usage/gpio-plus-and-raspi2/

Sklar, M. (2015). *Drive a 16x2 LCD with the Raspberry Pi.* Loetud 25. aprill 2017 aadressil https://cdn-learn.adafruit.com/downloads/pdf/drive-a-16x2-lcd-directly-with-a-raspberry-pi.pdf

Cetronic. *SM-S2309S.* Loetud 26. aprill 2017 aadressil http://descargas.cetronic.es/microservo.pdf

Summary

Title: Automating Pet Feeding with Raspberry Pi

The goal of this thesis was to automate the pet feeding process. To reach the goal, author of the thesis built a device with Raspberry Pi which was connected with a LCD screen, couple of buttons and a servomotor which could move the flapper inside the dispenser. To make the device work, author had to write software for it which was written in Python. Additionally the divces was tested in short period of time in which it was investigated to see if it meets the requierments.

The thesis is split into three parts. The first part is mostly about the general automating of pet feeding and why it is useful to automate the process. The second part talks about automated pet feeding devices that other authors have made. The third part describes the process in which the author built his own automated pet feeding device.

In the making of device, author used a cereal dispenser, which has a flapper that can be rotated to dispense food from it. The dispenser was modified so it would be possible to fit a servomotor on to the flapper and also to fit a screen and buttons on the outside of the dispenser. It was also necessary to find a place where to place the Raspberry Pi.

In the process of writing software, every line of code was described in detail, so it would be easy to follow and so people who aren't that good at programming could understand why chosen functions are used.

In the testing process author first tested devices prototype to find any problems as early as possible and later it was tested again after the devices was fully built. The fully built device was tested for a few days to see if it works after few days of working.

As the result of the thesis, the author finds that the goal of the thesis was achieved but he thinks that he could have taken the writing of the thesis more seriously and make something better than what he ended up with. Making of the device gave the author a lot of new experiences because before he hadn't done anything automating. The thesis also showed the author that he still needs to learn to manage his time a lot better.

LISAD

Lisa 1

Antud lisas on välja toodud kogu programmi kood, mis loodi antud töö käigus

```
import RPi.GPIO as GPIO
import datetime
import time
import Adafruit_CharLCD as LCD

lcd_rs = 25
lcd_en = 24
lcd_d4 = 23
lcd_d5 = 17
lcd_d6 = 18
lcd_d7 = 22
lcd_backlight = 2

lcd_columns = 16
lcd_rows = 2

lcd = LCD.Adafruit_CharLCD(lcd_rs, lcd_en, lcd_d4, lcd_d5, lcd_d6,
lcd_d7, lcd_columns, lcd_rows, lcd_backlight)

GPIO.setup(21, GPIO.IN, pull_up_down = GPIO.PUD_UP)
GPIO.setup(20, GPIO.IN, pull_up_down = GPIO.PUD_UP)

GPIO.setup(4, GPIO.OUT)
P = GPIO.PWM(4, 50)
p.start(7.0)

feedTimeMem = datetime.datetime(2001, 1, 1, 18, 30, 0)
```

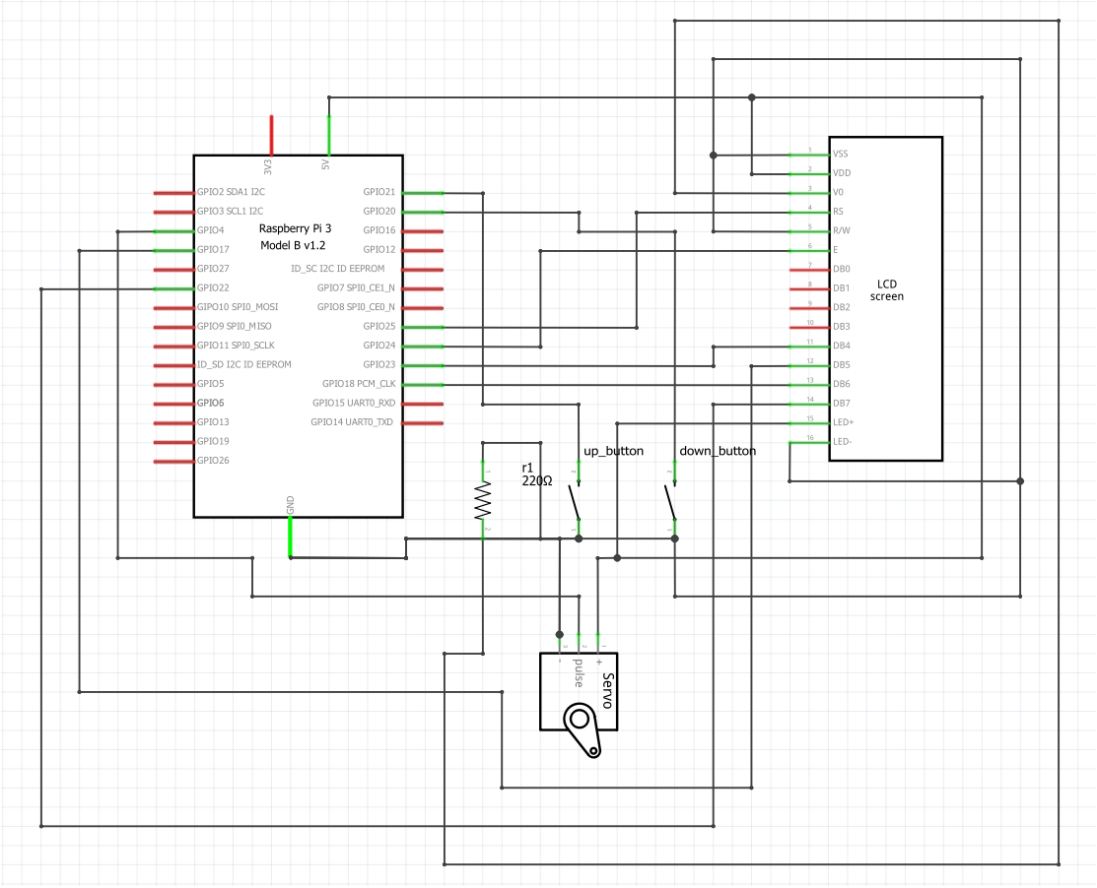
```
feedTimeMem2 = datetime.datetime(2001, 1, 1, 18, 30, 1)
feedTime = feedTimeMem.time()
feedTime2 = feedTimeMem2.time()


while True:
     up_button = GPIO.input(21)
     down_button = GPIO.input(20)
     rn = datetime.datetime.now().strftime("%H:%M:%S")
     feedt = feedTimeMem.strftime("%H:%M:%S")
     lcd.message("    "+feedt+"\n    "+rn)
     time.sleep(1.0)
     lcd.clear()
     If    datetime.datetime.now().time()    >    feedTime    and
datetime.datetime.now().time() < feedTime2:
          p.ChangeDutyCycle(7.0)
          time.sleep(1)
          p.ChangeDutyCycle(11.5)
          time.sleep(1)
          p.ChangeDutyCycle(2.0)
          time.sleep(1)
          p.ChangeDutyCycle(11.5)
          time.sleep(1)
          p.ChangeDutyCycle(7.0)
          time.sleep(1)
     if up_button == False:
          feedTimeMem = feedTimeMem + datetime.timedelta(0, 600)
          feedTimeMem2 = feedTimeMem + datetime.timedelta(0, 600)
          feedTime = feedTimeMem.time()
          feedTime2 = feedTimeMem2.time()
```

```
            lcd.message('Set to: '+feedTime.strftime(„%H:%M:%S"))
    if up_button == False:
            feedTimeMem = feedTimeMem - datetime.timedelta(0, 600)
            feedTimeMem2 = feedTimeMem - datetime.timedelta(0, 600)
            feedTime = feedTimeMem.time()
            feedTime2 = feedTimeMem2.time()
            lcd.message('Set to: '+feedTime.strftime(„%H:%M:%S"))
```

Lisa 2

Käesolevas lisas on välja toodud seadme elektriskeem (Joonis 7), mis on loodud tarkvara Fritzing abil, mille veebileht on leitav järgnevalt aadressilt: http://fritzing.org/home/



Joonis 7 Elektriskeem