TALLINNA TEHNIKAÜLIKOOL

Infotehnoloogia teaduskond

Triinu Tamm 175300IDAR

PUURILOOMADE MONITOORINGUSÜSTEEMI LOOMINE VIIRPAPAGOIDE NÄITEL

Diplomitöö

Juhendaja: Meelis Antoi

Magistrikraad

Tallinn 2020

Autorideklaratsioon

Kinnitan, et olen koostanud antud lõputöö iseseisvalt ning seda ei ole kellegi teise poolt varem kaitsmisele esitatud. Kõik töö koostamisel kasutatud teiste autorite tööd, olulised seisukohad, kirjandusallikatest ja mujalt pärinevad andmed on töös viidatud.

Autor: Triinu Tamm

13.01.2020

Annotatsioon

Diplomitöö eesmärgiks on luua rakendus, mis toetab puuriloomade omanike igapäevast elu. Kiire elutempo tõttu võib omanikul ununeda oma puuris elavate koduloomade eest hoolt kanda, kuna nad ei saa tingimata olla nii silmapaistvad või häälekad, et oma vajadustest omanikule teada anda. Seetõttu luuakse diplomitöö raames rakendus, mis jälgib ning teavitab puuris toimuvast ning annab lõppkasutajale märku, kui on vaja sekkuda ühel või teisel viisil.

Loodud lahendus on uudne oma idee poolest, mida tulevikus saab rakendada ka suurematele koduloomadele, kes puuris ei ela, näiteks kassidele ja koertele. Töö raames loodud rakendus keskendub konkreetselt viirpapagoidele ning nende puuripõhisele elukeskkonnale. Rakendus jälgib lindude toidu- ja joogiveetaset, samuti teavitab taimeri abil, kui puuriliiva või vett on vaja vahetada. Tänu sellele ei pea loomaomanik ise meeles pidama, kas ja millal ta näiteks lindude vett vahetas, vaid saab toetuda süsteemile, mis seda tema eest teeb.

Diplomitöö raames loodud rakendus ei ole lõpptoode, vaid pidevas arengus olev prototüüp.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 29 leheküljel, 4 peatükki, 11 joonist, 0 tabelit.

Abstract

Creating a Monitoring System for Pets in Cages by the Example of Budgerigars

The aim of the thesis is to create a monitoring system for pets in cages to help pets' owners in their everyday life. Life today is fast-paced and full of different obligations for people to remember and act upon. In busier days owners might, for example, forget to feed their pets in cages simply because pets might not be successfully able to signal their needs to their owners in time. Pets in cages are usually smaller and less noisy and therefore can go unnoticed since they are not always "in the way" or asking for attention. The purpose of the application created in the thesis is to support the owners so that both the pets' owners and pets themselves will not suffer from any shortcomings.

The system created is innovative since to the author's knowledge there has been no such application created before. In the future it can be developed for bigger pets who do not live in cages, for example cats or dogs. The thesis focuses on budgerigars and their cage-based life in particular. It monitors the levels of food and water in the containers and lets the owners know by email if there is a need to make any changes. The system also keeps a timer on when the sand in the cage or water in the container has last been changed. The bottom of the tray needs to be cleaned at least once a week and water should be changed at least every two or three days. That way the owner does not need to remember when the sand or water had been last changed, nor does he or she constantly need to check whether the food or water for budgerigars is running out. The owner can count on the system and therefore worry less about the current status of the birds.

The application created is not a finished product, but rather a prototype being developed.

The thesis is in Estonian and contains 29 pages of text, 4 chapters, 11 figures, 0 tables.

Lühendite ja mõistete sõnastik

CLK	*Clock*, info sünkroniseerimiseks
DAT	*Data*, info edastuseks
GND	*Ground*, maandus
I2C-liides	*Inter-Integrated Circuit,* kahejuhtmeliides
IP-aadress	*Internet protocol address*, Interneti aadress
RST	*Reset*
SD-kaart	*Secure Digital,* mälukaardi formaat
SMTP	*Simple Mail Transfer Protocol*, lihtne meiliedastusprotokoll
SSID	*Service Set IDentifier*, teenusepakkuja poolt antud nimi võrgule
USB	*Universal Serial Bus*, universaalne järjestiksiin
VCC	Toide
Wi-Fi	Juhtmevaba Internet

Sisukord

Jooniste loetelu

Sissejuhatus

Tänapäeva kiire eluviisi juures on raske meeles pidada kõike, mida on päeva jooksul vaja ära teha, olgu see siis arvete õigeaegne tasumine, postipakil järgi käimine või kas või vajalike toiduainete koju ostmine. Inimestel, kel on lisaks igapäevastele kohustustele ka koduloomad, lisandub eelpool nimetatud tegevustele veel lemmiku(te) söötmine-jootmine, nendega tegelemine ja loomaarsti juures käimine. Tiheda päevakava juures võib nii mõnigi nendest tegevustest ununeda, eriti kui tegemist on koduloomadega, kes nii aktiivselt endast märku ei anna, näiteks viirpapagoidega. Tihti võib juhtuda, et unustatakse vahetada lindude vett või lisada toitu, kuna üldiselt on sellised puuriloomad kogu aeg ühes kohas ning kiires elutempos võivad nad märkamatuks jääda. Kuna linnud ei söö, joo ega musta puuripõhja samas koguses iga päev, ei saa eeldada, et neid tuleb hooldada täpselt nendel päevadel nädalas ja nii nädalast nädalasse, kuust kuusse. Seetõttu ei ole kalendrisse märkme tegemine asjakohane, kuna võib juhtuda, et tol päeval ei ole vaja lindudele süüa anda, küll aga oleks seda vaja teha juba järgmine päev. Antud töö raames loodav lahendus ongi mõeldud selleks, et vältida sarnaste olukordade tekkimist, kus igapäevaselt on vaja meeles pidada, millal viimati lindudele süüa anti ning millal seda järgmisena uuesti teha tuleb.

Töö eesmärgiks on luua monitooringusüsteem, mis jälgib viirpapagoide toidu- ja veekogust ning annab märku, kui toit või vesi hakkab otsa saama. Samuti hakkab süsteem jälgima seda, millal lindude puur vajab puhastamist ning millises temperatuuris linnud parasjagu elavad. Selle tulemusel on viirpapagoide toidu- ja veenõud alati täidetud, puur puhas ning linnuomanikel elu selle võrra lihtsam. Pikemas perspektiivis lisandub ka kaameravõimekus, mille abil saab telefonis jälgida lindude hetkeolukorda ning nende tegemisi.

Töö on jagatud neljaks peatükiks, kus esimeses peatükis analüüsitakse lahenduse jaoks sobilikke võimalusi ja alternatiive, teises peatükis seletatakse lahti töökäik, s.o arendusplaadivahelised ühendused ning kirjutatud kood, kolmandas peatükis kirjeldatakse, mis keskkonnas toimus testimine ning millised olid testimise tulemused ja

tehtud muudatused ning viimases peatükis võetakse kokku autori poolt tehtud töö – püstitatud eesmärk ja loodud lahendus.

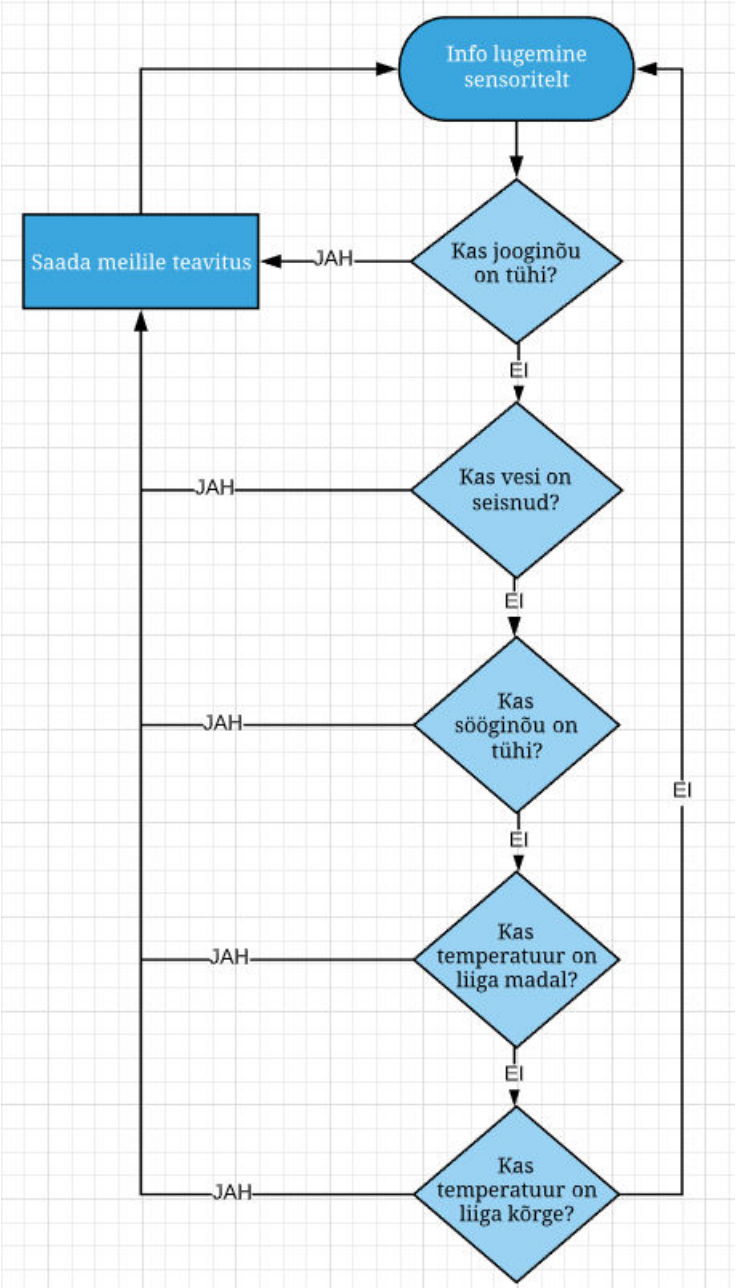
1 Analüüs

Peatükk kirjeldab ja seletab, mida läheb vaja loodava lahenduse realiseerimiseks. Kõige suurem osa on seotud arendusplatvormiga, kuna sellest saab kõik alguse. Samuti määrab arendusplatvormi valik ära lisafunktsionaalsuste võimalused, lahenduse edasise arengukäigu ning programmeerimiskeele. Valiku tegemisel arvestab autor lisaks eelnevale ka enda eelistustega keele ja platvormi osas.

Analüüsi peatükis autor ei süvene konkreetselt erinevatesse arendusplatvormidesse, vaid toob välja nende eelised ja puudujäägid, ning seejärel põhjendab otsust. Kuigi platvormi valikul peetakse silmas tulevikuaspekti, s.t lahenduse edasiarendamist reaalseks, laia ringlusesse minevaks tooteks, ei seata seda siiski esikohale.

Lahenduse tööpõhimõte on lihtne. Arendusplaat saab anduritelt info, mida töödeldes teeb see otsuse, kas kasutajat tuleb teavitada toimunud muudatustest või mitte. Temperatuuri- ja niiskusesensori eesmärk on informeerida kasutajat, kui temperatuur lindude puuri juures on tõusnud liiga kõrgeks (27 kraadi Celsiust või üle) või madalaks (15 kraadi Celsiust või alla). Niiskusetase on hetkel lisainformatsioon, millega loodav lahendus otseselt midagi peale ei hakka. Söögitase peab jõudma sellise tasemini, mis herkonlüliti aktiveerib, et kasutajat meili teel teavitada, samamoodi toimivad ka veetaset mõõtvad elektroodid.

Järgnevalt on välja toodud loodava lahenduse tööpõhimõte plokkskeemina (Joonis 1):



Joonis 1. Lahenduse protsessiplokkskeem.

1.1 Arendusplatvorm

Arendusplatvormi puhul jäi autorile silma kahe peamise – Arduino ja Raspberry Pi – erinevad mudelid. Peatükis ei hakata lahkama nimetatud arhitektuuride mudelite eripärasid, vaid võrreldakse arhitektuure kui selliseid ja nende sobivust lahenduse loomiseks.

Raspberry Pi langes kohe konkurentsist välja, kuna on autori arvates liiga võimekas loodava töö jaoks. Antud lahenduse jaoks ei ole oluline, et arendusplatvorm oskab töötada nagu arvuti, samuti ei ole oluline, et sellele saab külge panna hiire, klaviatuuri, monitori ning muud mugavused[1]. Kuna loodava töö eesmärk on puhtalt monitoorida ning teavitada kasutajat keskkonnas toimunud muudatustest, puudub vajadus selliste funktsionaalsuste järgi. Ka tulevikus ei ole planeeritud praegu arendatava töö ümber tegemist arvuti peale. Lahenduse seisukohalt seisneb Raspberry Pi eelis selles, et alates 2013. aastast on seal peal kaameramoodul, mis lihtsustaks tulevikus kaameravõimekuse arendamist[2]. Samas muudab Raspberry Pi kasutamise ebamugavaks tõik, et see vajab eraldi operatsioonisüsteemi installeerimist. Kuigi see ei ole autori jaoks keeruline, vajab see ka eraldi, vähemalt kaheksa gigabaidi suurust SD-kaarti, kuhu peale operatsioonisüsteem ning tulevane kood salvestada[3]. Seetõttu ei arva autor, et Raspberry Pi on kõige mugavam ning sobilikum variant diplomitöö teostamiseks.

Autori silmis on lahenduse loomiseks kõige parem variant Arduino. See on võrdlemisi lihtsasti käsitletav ning laia kasutajaskonnaga. Samuti on olemas palju erinevaid õpetusi nii Arduino enda koduleheküljel (https://www.arduino.cc/) kui ka mujal Internetis (hea näide on https://learn.sparkfun.com). See on kohe valmis kasutamiseks ning ei vaja eraldi operatsioonisüsteemi installeerimist, rääkimata lisaosade ostmisest[4]. Kuna autori enda soov on selgeks õppida C++-keel, on Arduino alustamiseks sobilik vahend. Ka on Arduino (või sellepõhist) arendusplaati odavam soetada kui näiteks Raspberry Pi oma.

Pidades silmas eelnevaid põhjendusi, valib autor lahenduse realiseerimiseks Arduino arendusplatvormi, täpsemalt Arduino-põhise NodeMCU Amica (ESP8266 12-E). Arendusplaat on odav, võrdlemisi laialt levinud ning omab Wi-Fi-moodulit, mis antud töö jaoks on väga suur eelis[5]. Just hind on prototüübi loomisel määrav, kuna võimaldab väheste kuludega valmis arendada töötava lahenduse, mille hiljem saab vajadusel ümber tõsta mõnele teisele arendusplaadile. Samuti on oluline, et valitud mudeli kohta on Internetis palju juhendeid, mis lihtsustavad probleemide lahendamist nende tekkimise korral.

1.2 Sensorid ja moodulid

Kogu töö peale on kokku neli erinevat sensorit/moodulit – temperatuuri- ja niiskusesensor lindude keskkonnaga kursis olemiseks, herkonlüliti söögi koguse mõõtmiseks, elektroodid joogiveetaseme mõõtmiseks veeanumas ning kellamoodul ajalise määratlusega teavitusteks. Puuri puhastamisest teavitamise funktsionaalsus tugineb samuti kellamoodulile, mille abil omanikku teavitatakse, kui puur on olnud puhastamata nädal aega. Sama mooduliga teavitab süsteem, kui joogivesi on vahetamata olnud kaks päeva, kuid veetaset mõõtev andur ei ole varasemalt teada andnud vee lisamise vajadusest.

1.2.1 Kellamoodul

Kellamooduli valimisel võrreldi kahte mudelit – DS1302 ja DS1307. Mõlemad variandid on väga sarnased, kuid DS1302 vajab kolme ühendust korrektseks töötamiseks ning DS1307 kahte ühendust (I2C-liides)[6]. DS1307 mudelil on suurem vahemälu, mis autori poolt loodud töös ei ole määrava tähtsusega[7]. Kuna töö tegemise ajal oli DS1302 autorile kättesaadav ja kiire variant, otsustas autor kahest variandist selle kasuks. Lisaks hetkelisele kättesaadavusele on DS1302 odav, lihtne kasutada ning sarnaselt DS1307 mudelile omab akut, mis konkreetses töös on vajalik töökindluse lisamiseks.

1.2.2 Temperatuuri- ja niiskuseandur

Autor valis kahe temperatuuri- ja niiskusesensori vahel – DHT11 ja DHT22. Mudelite vahel on väikesed täpsuse ja värskendamissageduse erinevused. DHT11 loeb niiskust 5% täpsusega ning temperatuure loeb ±2°C täpsusega, samal ajal, kui teine võrdluses olev mudel loeb niiskust 2-5% ja temperatuuri ±0.5°C täpsusega. Värskendamissagedus on vastavalt iga ühe sekundi (1Hz) ja iga kahe sekundi (0,5Hz) tagant[8]. Autori jaoks ei ole niiskusetaseme täpsus niivõrd oluline, kuna see ei ole oluline lahenduse loomisel, küll aga on oluline temperatuuritäpsus ning DHT11 puhul kahekraadine erinevus on juba piisav eksimisruum, et valida DHT22. Lisaks täpsusele on valitud mudel ka odav, suhteliselt populaarne ning kaubanduses kättesaadavam.

1.2.3 Herkonlüliti

Kuna lindude söögiks olevad seemned on suhteliselt kerged, on raske leida kaalu mõõtmiseks sobivat vahendit, mis ühtlasi sobiks ka söögianuma sisse ning ei oleks puurist väga väljaulatuv ega häiriks linde. Seemnetest järgi jäänud kestad on kergemad kui seemned ise, mistõttu kaalu järgi toidukoguse mõõtmine on sobiv lahendus. Seetõttu valis autor toidu koguse mõõtmiseks herkonlüliti ehk keelrelee. Leides herkonile sobiva magneti ning söögianumasse piisavalt tundliku vetruva materjali, peaks see suutma reageerida toidukoguse erinevustele ning andma edasi adekvaatset infot.

1.2.4 Veetaseme mõõtmine

Veetaseme mõõtmiseks on kolm peamist valikut. Üheks variandiks on kasutada mahtuvuslikku andurit. See on täpne, mistõttu saaks tulevikus loodavale lahendusele lisada juurde funktsionaalsuse konkreetse veetaseme lugemiseks ning kasutajale meilile saatmiseks. Arvestades aga joogianuma kuju, on keeruline diplomitöö raames mahtuvusliku anduri kasutamist realiseerida.

Teiseks valikuks on ultrahelianduri kasutamine. See on samuti täpne, kuid, nagu ka mahtuvusliku anduri puhul, on see mõõtmetelt suur ning arvestades joogitopsi kuju ja suurusega, ei sobi lahenduse realiseerimiseks.

Kolmandaks variandiks on kasutada elektroode, mis lähevad joogitopsi sisse ning annavad märku vaid sellest, kas vett on või ei ole. Neid on lihtne kasutada ning füüsilistelt mõõtmetelt väga väikesed, mistõttu sobivad hästi lahenduse loomiseks. Elektroodide kasutamisel on oluline nende materjal. Kuna nende töökeskkond on vees, on tähtis, et need ei eraldaks ohtlikke aineid, mida linnud omale sisse jooksid.

Lahenduse loomiseks valis autor metallist elektroodide kasutamise vee mõõtmiseks, kuna tegemist on kõige lihtsamini realiseeritava ning kasutatava valikuga, mis täidab töö jaoks vajalikku eesmärki.

1.3 Programmeerimiskeel ja arenduskeskkond

Töö autor valis kolme programmeerimiskeele vahel – Micropython, C++ ja C. MicroPython on Pythoni järeltulija mikrokontrolleritele. Mõlemad on teineteisele väga

sarnased ning suhteliselt lihtsasti hoomatavad[9]. C++- ja C-keel on keerulisemad. C++-keel on välja kujunenud C-keelest ning on natuke rohkemate võimalustega kui C-keel[10].

Autor eelistas kasutada oma töös C++-keelt, mis on samaväärselt laialt levinud ning võimaldab vajadusel loodava lahenduse ümbertõstmist teisele platvormile.

Kood kirjutatakse ja laetakse arendusplaadile Arduino IDE 1.8.10 versiooniga.

1.4 Teavituste saatmine

Meilile teavituste saatmine toimub läbi autori poolt valitud SMTP-serveri (*Simple Mail Transfer Protocol)*. Serveri valiku otsuse tegemine sõltub suuresti sellest, kas teenus on tasuline või ei. Samuti on oluline, et valitud serveri kohta oleks varasemalt informatsiooni ning vajadusel juhendeid. Autor eelistab kasutada sellist, mis on laiemalt levinud.

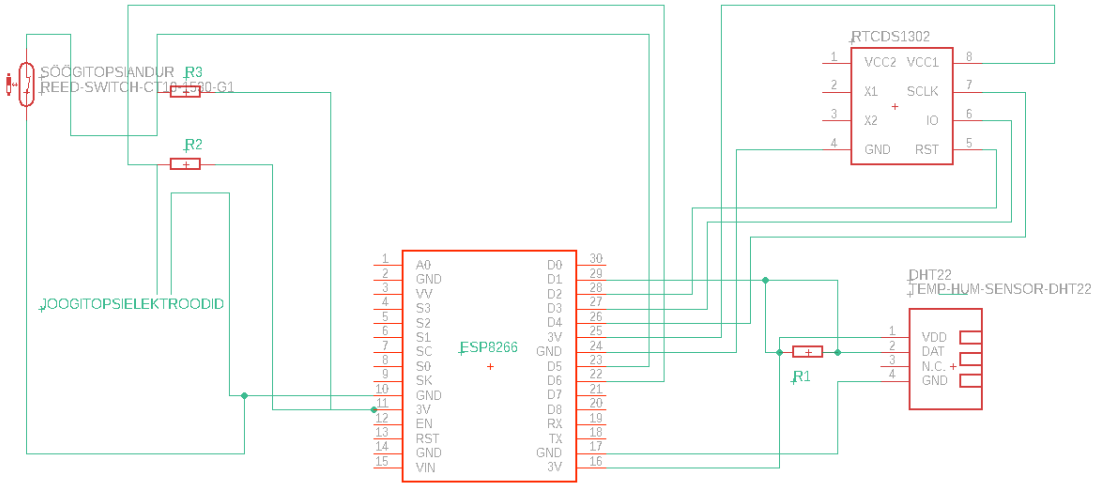
Lahenduse loomiseks otsustas autor kasutada SMTP2GO teenust. Otsimaks sobivat SMTP-serverit, paistis kõige enam silma just väljavalitu, kuna sellel on olemas ka tasuta versioon, samuti on seal mõistlikud piirangud, mis hetkel sobivad autori loodud töö jaoks kõige paremini – kuus saab saata kuni 1000 e-maili ning tunnis 25 e-maili. Juhul, kui antud piirangut peakski ületama, hoitakse limiidist üle läinud e-maile puhvris nii kaua, kuni uus tund/kuu saabub. Alati on võimalus minna üle tasulisele teenusele, kui kasutajaskond suureneb ning mahud kasvavad. Samuti on hea, et tasuta versioon ei aegu, s.t puudub n-ö prooviperiood, peale mida tuleb kas minna üle tasulisele teenusele või leida uus rahavaba variant [11]. SMTP2GO on laialt kasutusel ka erinevates ettevõtetes, kus nende teenust ja töökindlust erinevatel saitidel kiidetakse, mis samuti suurendab autori usaldust tehtud valiku suhtes [12][13].

Kuna vaikesättena tahab SMTP2GO teenus kasutaja domeenikonto lisamist ning autoril domeen puudub ja ei pea vajalikuks ka prototüübi jaoks selle loomist, võttis autor ühendust SMTP2GO meeskonnaga, kes kiiresti andsid võimaluse kasutada nende teenust ka ilma domeenikontota. Mõtte selleks sai autor teiselt kasutajalt, kes samuti seisis probleemi ees, kus isikliku projekti tarbeks oli vaja SMTP-serveri teenust, kuid domeen puudus[14].

2 Töö käik

Antud peatükk kirjeldab ettevalmistusi Arduino IDE's ning arendusplaadi ja sensorite/moodulite omavaheliseks suhtlemiseks vajalike ühenduste loomist. Samuti on peatükis lahti seletatud rakenduse kood, selle toimimine ning koodi kirjutamisega kaasnenud probleemid.

Järgnevalt on juurde lisatud Eagle-programmiga loodud elektriskeem, mis kirjeldab lahenduse funktsionaalsuste jaoks vajalikke ühendusi (Joonis 2). Peatüki edasises osas kirjutatakse igast komponendist eraldi, juurde on lisatud töös kasutatud sensori/mooduli pilt joonisena.



Joonis 2. Lahenduse elektriskeem.

2.1 Arduino IDE

Esimese sammuna laetakse tarkvara Arduino kodulehelt alla[15]. Arendusplaat on ühendatud arvutiga USB-A->USB-micro (*Universal Serial Bus)* kaabliga. Koodi kirjutamise alustamiseks tuleb ESP8266 mooduli tugi lisada Arduino programmeerimise keskkonda ning siis laadida alla teek selle jaoks. Peale seda on valitud Arduino IDE 1.8.10 tarkvaras õige arendusplaat ning port, millesse see

ühendatud on[16]. Autori töös on Tools -> Board valikuks NodeMCU 1.0 (ESP 12-E Module) ning ühenduspordiks COM3.

2.2 RTC DS1302

Kellamooduli CLK-, DAT- ja RST-jalg (*Clock, Data, Reset)* on ühendatud vastavalt D4 (GPIO02), D3 (GPIO00) ja D2 (GPIO04) NodeMCU jalgadega. VCC-jalg on ühendatud 3V3 jalaga arendusplaadil ja GND-jalg (*Ground)* samale nimetusele vastavale jalale (Joonis 3).

Joonis 3. RTC DS1302.

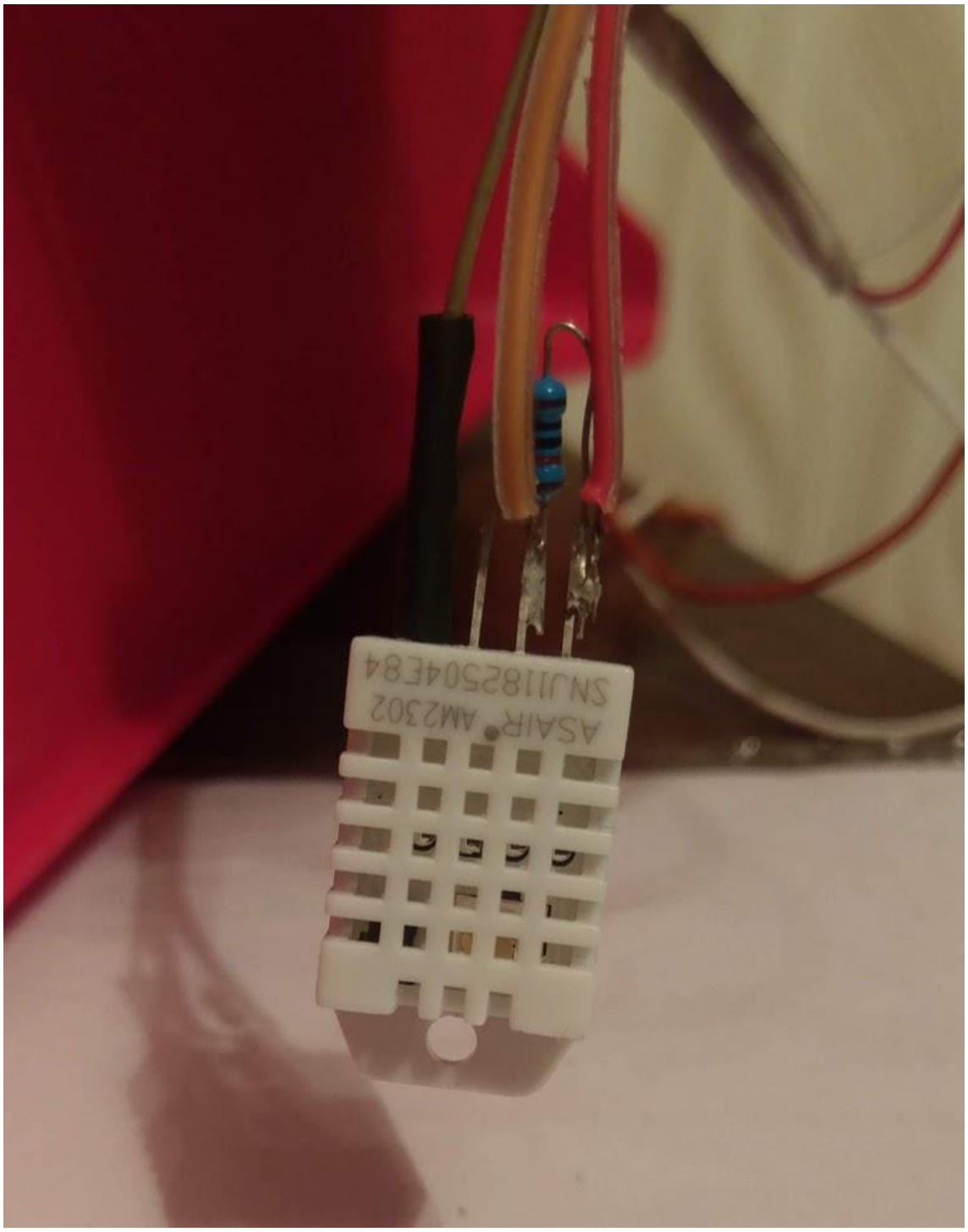
Edasi tuleb Arduino IDE's lisada teek nimega DS1302.h, mis tagab toe mooduli kasutamiseks. Järgmisena on deklareeritud kell muutujanimega kell, sulgudes jalgade GPIO-väärtused RST-DAT-CLK järjestuses. Esimesel korral tuleb kellale ette öelda, mis ajaparameetrid hetkel on (ja millest see edasi aega lugema hakkab). Need kolm rida tuleb järgnevateks käivitusteks välja kommenteerida, muidu hakkab kell igal käivitusel uuesti lugema just sellest etteantud ajast. See oli ka üks probleemidest kellamooduli õigesti tööle saamisel[17].

Üheks esimeseks murekohaks oli see, et koodi üleslaadimisel arendusplaadile ilmus väljundiks erinevate tähemärkide rida aja ja kuupäeva kuvamise asemel. Selle lahenduseks tuli andmevahetuskiirus muuta ka monitooringus endas 9600 peale,

ühildumaks käsu „Serial.begin(9600);“ poolt etteantud kiirusega (vaikeväärtus oli 115200)[18]. Peale nimetatud muudatuse sisseviimist hakkas kell õiget aega kuvama ning rohkem suvalisi tähemärke ei esinenud.

2.3 DHT22

DHT22 puhul tuleb ühendada kokku kolm jalga, millest ainult üks edastab infot. VCC- ja GND-jalg ühenduvad 3V3- ja GND-jalgadega arendusplaadil, DAT-jala ühendas autor D1 (GPIO05) jalaga. DAT- ja VCC-jalga ühendab omavahel 5.1 kilo-oomine takisti (Joonis 4).



Joonis 4. DHT22.

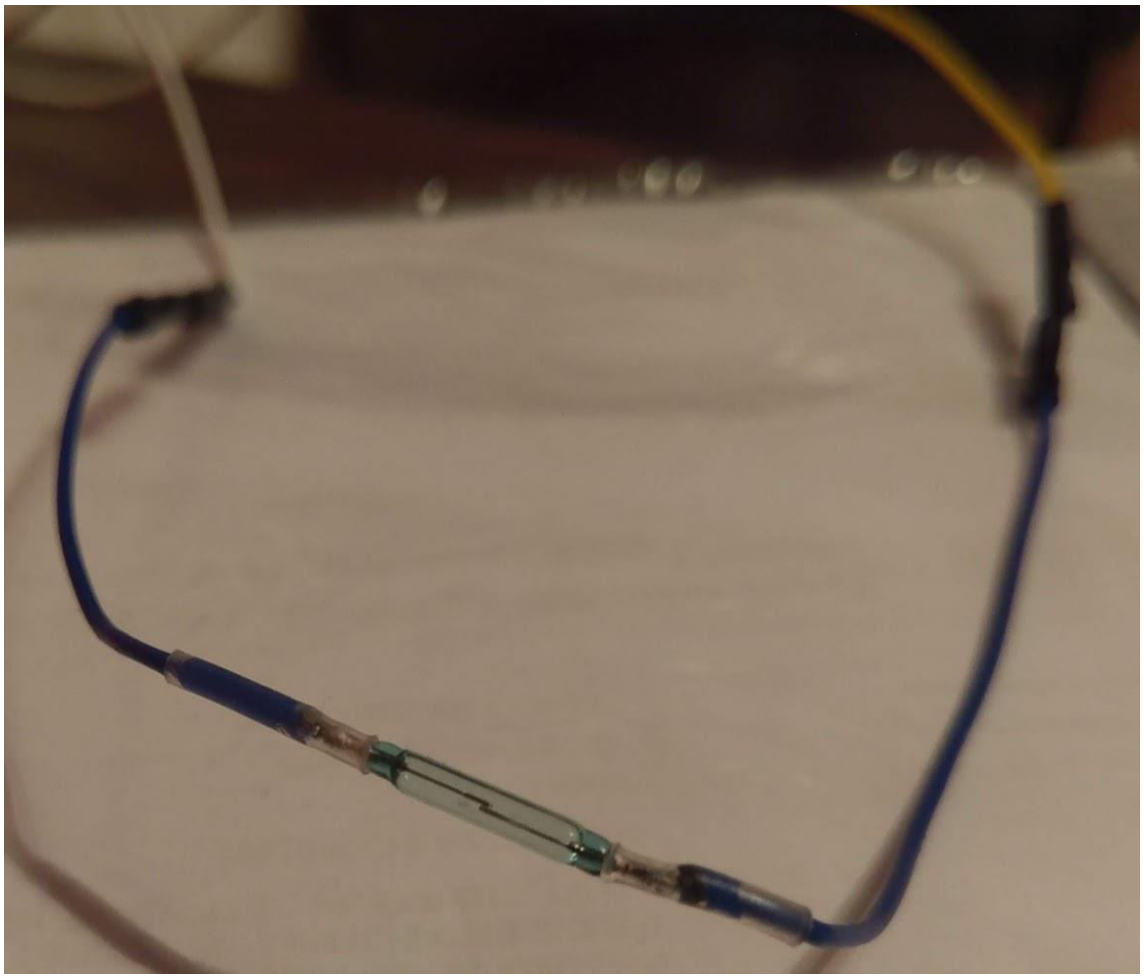
Andurilt andmete saamiseks on koodi vaja lisada DHT.h teek. Sellele järgnevalt on defineeritud GPIO-väärtuses jalg ning anduri mudel, millest andmeid edastatakse. Käsuga „DHT niiskusTemp(DHTPIN, DHTTYPE);“ on defineeritud anduri olemasolu. Lisaks arendusplaadile tuleb ka andur käivitada. Käskudega „niiskusTemp.readHumidity()“ ja „niiskusTemp.readTemperature();“ saab anduri käest

vastavalt loetud niiskuse- ja temperatuuritaseme, mida monitooringusse ka kuvatakse. IF-klausliga kontrollitakse, et loetud andmed ei oleks tühjad[19].

2.4 Herkonlüliti

Herkonlüliti on paigutatud söögitopsi põhja alla, väljapoole. Topsi sees on poroloonitükk, mis toimib vetruva materjali eest. Selle peal on 3D-prinditud alus, mis läheb porolooni peale ja hakkab hoidma endas lindude terasid (Joonis 5). Aluse all on magnet, mis aktiveerib herkoni (Joonis 6). Kui lindudel on anumas piisavalt seemneid, vajub alus raskusega allapoole ning lüliti püsib magneti mõjul koos. Kui seemned hakkavad otsa saama, muutub alus kergemaks, mistõttu poroloon tõuseb ülespoole ning magnet läheb kaugemale lülitist. Selle tulemusel viimase vooluring katkestatakse ning saadetakse teavitus meilile, et toit on otsas.

Joonis 5. Lahendus 3D-prinditud aluse ja porolooniga.



Joonis 6. Herkonlüliti.

Lüliti ise on otseselt ühendatud kahe jalaga – maanduse ja vooluga. Lisaks on informatsiooni edastav osa (kokku joodetud voolu andva juhtmega) ühendatud D5 (GPIO 14) jalaga.

Arduino IDE tarkvaras tuleb deklareerida herkoni muutuja vastava GPIO-jalaga. Selle järgi tuleb koodis ära määrata see, mida konkreetne muutuja teeb – kas ta annab sisendi või väljundi. Antud juhul annab herkoniga ühendatud DAT-jalg sisendi loodavale programmile. Järgmisena on määratud, mida teeb lüliti, kui selle lähedal on magnet. Kuna takisti (5.1 kilo-oomine) on pandud voolu ette, tähendab vooluringis seisund 1, et toit on otsas ning 0, et toit on olemas (Joonis 7)[20].

```
//GPIO jalad
int herkon = 14;

void setup() {

  Serial.begin(9600);

  //jalgade funktsionaalsuse määramine
  pinMode(herkon, INPUT);
}

void loop() {

  //Herkoni lugemine
  if (digitalRead(herkon) == HIGH) {
    Serial.println("Toit otsas!");
  }
  else if (digitalRead(herkon) == LOW) {
    Serial.println("Toit olemas!");
  }
  delay(1000);
}
```

Joonis 7. Keelrelee kasutamise kood.

2.5 Veetaseme lugemine

Joogitopsi põhjast läbi on pandud kaks elektroodi – üks lühem ning teine pikem. Kui vesi on pikemast üle, on vesi olemas, kui vesi on kahe vahel, annab rakendus teada, et vesi hakkab otsa saama (Joonis 8).

Joonis 8. Elektroodid joogitopsis.

Elektroodid on ühendatud voolu ja maandusega, sellele lisandub DAT-ühendus, mis arendusplaadil läheb D5 (GPIO 12) jala külge.

Sarnaselt herkonile tuleb programmis deklareerida muutuja, mis annab sellele GPIO-väärtuse. Järgmisena on määratud jala eesmärk, mis samuti on sisendiks lahendusele. Lugemaks elektroodide poolt saadud informatsiooni, on koodis kirja pandud, mida seisund 1 ja 0 tähendavad – vastavalt vesi on otsas ja vesi on olemas (Joonis 9).

```
//GPIO jalad
int elektrood = 12;

void setup() {

  Serial.begin(9600);

  //jalgade funktsionaalsuse määramine
  pinMode(elektrood, INPUT);
}

void loop() {

  //elektroodi lugemine
  if (digitalRead(elektrood) == HIGH) {
    Serial.println("Jook otsas!");
   }
   else if (digitalRead(elektrood) == LOW) {
    Serial.println("Vesi olemas");
   }
  delay(1000);
}
```

Joonis 9. Elektroodide kasutamise kood.

2.6 Meilide saatmine

Teavituste saatmiseks ESP8266'lt autori meilile on kõigepealt vajalik arendusplaadi ühendamine Internetiga. Selleks peab Arduino IDE's olema lisatud teek ESP8266WiFi.h, mis võimaldab kasutada WiFiga ühendamiseks vajalikke käske. Füüsiliselt pole tööga midagi lisaks tehtud, kõik on seadistatud tarkvara sees.

Peale teegi lisamist deklareeritakse ühendatava ruuteri SSID (*Service Set IDentifier)* ja salasõna, lisaks veel SMTP-server, mille kaudu hakkab ESP8266 meile saatma. Käsuga „WiFiClient esp;" deklareeritakse kliendi olemasolu. WiFi-ühenduse alustamiseks tuleb koodireaga „WiFi.begin();" ette anda ruuteri nimi ning parool. Peale seda kontrollitakse ühenduse olemasolu ning kinnituseks kuvatakse ka saadud IP-aadress (*Internet Protocol Address).*

SMTP-serveriga luuakse ühendus käsu „esp.connect(server, 2525);“ abil, kus 2525 on SMTP2GO port[21]. Kohe kontrollitakse üle ka see, et ühendus serveriga on saavutatud. Järgmisena saadetakse serverile käsk, mis alustab suhtlust arendusplaadi ja serveri vahel (esp.println(„EHLO www.headphones.com“);)[22]. Sellele järgneb lõppkasutaja autentimine, kus saadetakse base-64 formaadis kodeeritud kasutajanimi ja parool. Edasi määratakse ära, kes saadab meili ning kes selle kätte saama peaks, selleks on käsud „MAIL FROM: „ ja „RCPT TO: „. Neile lisandub „DATA“-käsk, kus, lisaks sellele, kes on saatja ja saaja, antakse ette ka meili teema ning sisu. Kõik see toimub esp.println() käsuga. Kui kiri on formeeritud, lõpetakse suhtlus serveriga ära. Järgnevalt on lisatud kood WiFi-ühenduse loomiseks ja meili saatmiseks puudulike DHT22-sensori näitude näitel (Joonis 10)[23].

```
#include <ESP8266WiFi.h>
WiFiClient esp;

const char* wifiSSID = "ZyXEL NBG-418N v2";
const char* wifiPSW = "xxx";
const char server[] = "mail.smtp2go.com";

void setup() {
  Serial.begin(9600);

  WiFi.begin(wifiSSID, wifiPSW);
  //kuni ühendus ei ole saavutatud, prindi punkt
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  //kinnituseks antakse IP-aadress
  Serial.print("Ühendatud wifivõrku nimega ");
  Serial.print(wifiSSID);
  Serial.print(". IP address ");
  Serial.println(WiFi.localIP());
}

void loop() {
  if (esp.connect(server, 2525) == 1) {
    Serial.println(F("Ühendus SMTP2GO serveriga loodud!"));
  }
  else {
    Serial.println(F("Ühendamine ebaõnnestus!"));
    return 0;
  }
  Serial.println(F("Sending EHLO..."));
  esp.println("EHLO www.headphones.com");
  Serial.println(F("Sending authentication request..."));
  esp.println("AUTH LOGIN");

  Serial.println(F("Sending user name..."));
  esp.println("xxx"); //kasutajanimi base-64 kodeeringus

  Serial.println(F("Sending password..."));
  esp.println("xxx"); //parool base-64 kodeeringus

  Serial.println(F("Sending from srxwrc@gmail.com"));
  esp.println(F("MAIL From: srxwrc@gmail.com"));

  Serial.println(F("Sending to srxwrc@gmail.com"));
  esp.println(F("RCPT To: srxwrc@gmail.com"));

  Serial.println(F("Sending the data part..."));
  esp.println(F("DATA"));
  esp.println(F("To: srxwrc@gmail.com"));
  esp.println(F("From: srxwrc@gmail.com"));
  esp.println(F("Subject: Näitude lugemine ebaõnnestus!"));
  esp.println(F("DHT22 sensorilt temperatuuri- ja niiskusenäitude lugemine ebaõnnestus!"));
  esp.println(F("."));

  Serial.println(F("Sending quitting request..."));
  esp.println(F("QUIT"));
  esp.stop();
  Serial.println(F("Disconnected from the server."));
}
}
```

Joonis 10. WiFi- ja SMTP-serveri ühenduse loomise kood.

Järgneval joonisel on kujutatud lõputöö raames valminud lahendust puuri küljes (Joonis 11). Arendusplaat on pandud puuri põhja alla koos RTC DS1302'ga. Lahenduse täispikk kood on nähtav Lisa 1 all[Lisa 1].

Joonis 11. Valmis töö puuri küljes olevana.

3 Lahenduse testimine

Diplomitöö raames loodud lahendus on testitud autori viirpapagoide puuri küljes. Esimest korda testis autor tehtud tööd viis päeva ning testimisel oli joogitopsiku küljes olevad elektroodid ning DHT22 andur, lisaks veel läbi kellamooduli toimivad puuri puhastamise ja joogivee vahetamise teavitus.

Autor otsustas söögitopsiku täituvuse mõõtmist mitte testida, kuna enne testimist selgus, et 3D-prinditud alus on liiga madal lindude söögi mahutamiseks – nii vähese söögiga ei ole võimalik herkonlülitit reageerima panna. Samuti selgus, et vedru, mis algselt oli topsi sisse paigutatud, ei olnud siiski sobilik, nagu ka poroloon, mis vedru asendama pidi. Seetõttu otsustas autor, et söögitopsi toimimist ei saa adekvaatselt testida ning tulemusi korrektselt hinnata.

Lisaks eelnevale selgus, et loodud söögitopsilahendus, kui see eelpool mainitud probleemi ei oleks omanud, oleks siiski tekitanud muresid seemnete juurde lisamisel. Kuna herkonlüliti pidi olema vahetult topsi küljes, oleks topsi puuri küljest eemaldamine olnud keeruline. Seemneid ei saa juurde lisada eelnevast korrast ülejäänud koori ära viskamata jättes ning kui topsi ei saa korralikult ära võtta ilma keelreleed topsi küljest eemaldamata, muudab see autori poolt loodud lahenduse lõppkasutajale ebamugavaks.

Samuti selgus sarnane tõik joogitopsiga, kus topsi põhja sisse on pandud elektroodid. Joogivee vahetamine on oluliselt keerulisem selle tõttu, kuna joogianumat peab aeg-ajalt ka puhastama, et põhjas seisnud vesi topsi seinu mustaks ei jätaks. Kui aga anuma põhi on põhimõtteliselt puuri küljes kinni, ei saa põhja puhastada jooksva vee all, samuti on joogivee vahetamine raskendatud, kuna topsi ei saa ühe tükina puuri küljest ära võtta ning kraanikausi juurde viia, vaid peab eraldama ning siis ettevaatlikult kaks osa omavahel ühendama nii, et vesi maha ei jookseks.

Esimese testimise tulemused olid võrdlemisi ebaedukad. Kuigi enne reaalsesse testkeskkonda minekut kõik funktsionaalsused töötasid, s.o meili saatmine, joogivee

olemasolu hindamine ning taimer, ei olnud viiepäevase testimise tulemusel kordagi autori meilile teavitust tulnud kummagi veega seotud probleemi kohta.

Selline anomaalia oli põhjustatud käsu „delay“ liiga suurest väärtusest[24]. Suure väärtuse eesmärk oli tsükli kordamine iga 12 tunni järel, et kontrolle ei toimuks liiga tihti. Sellest tulenevalt muutis töö kirjutaja „delay“ väärtuse 12-tunnilt ühe tunni peale ning teisel testil toimusid pidevad kontrollid korrektselt. Autor leiab, et tehtud muudatus lisas lahendusele usaldusväärsust, kuna kontrollitav ja saadetav info on aktuaalsem. Selline aktuaalsus on eriti oluline just temperatuurimuudatuste korral, kus ühetunnine tsükkel annab kasutajale võimaluse olukorraga kursis olla ning vajadusel lindude juurde ise kohale minna ja veenduda korrasolekus.

Peale tehtud parandust kõik töös ettenähtu, v.a söögianum, töötas korrektselt.

4 Kokkuvõte

Diplomitöö eesmärk oli luua rakendus, mis jälgib viirpapagoide puuris toimuvaid muudatusi ning annab lõppkasutajale emaili teel märku, kui on vaja temapoolset sekkumist.

Diplomitöö lõpuks lõi autor arendusplaadi ja erinevate moodulite vahel vajalikud ühendused ning kirjutas koodi sensoritelt info saamiseks. Söögitopsis olevate seemnete olemasolu mõõtmine jäi poolikuks, kuna autor ei leidnud sobilikke vahendeid, mis lindude sööki täpselt kaaluks ja adekvaatset infot edastaks. Joogitopsi veetaseme mõõtmine enne mõlemat testi ning pärast teist testi töötas, samuti töötas meili- ja taimerifunktsionaalsus.

Esmakordsel lahenduse testimisel selgus, et lahendus, mis enne puuri külge paigaldamist töötas, õiges keskkonnas ei töötanud. See oli tingitud „delay“ käsule liiga suure väärtuse lisamisest.

Loodud töö hõlbustab lõppkasutaja igapäevast elu ning hoiab viirpapagoid õigeaegselt hoolitsetud. Seetõttu ei pea loomapidaja enam muretsema oma lemmikloomade heaolu eest, vaid saab usaldada ja tugineda loodud süsteemile, mis ise peab järge ning teavitab puudujääkidest. Majanduslik efekt puudub, kuna hetkel rakendus ei tõsta ega langeta rahalist kulu, vaid hoiab lõppkasutaja aega ja üleliigset muretsemist kokku.

Üks esimesi samme edasi on leida töötav lahendus söögitopsi jaoks, millega saaks lindude olemasolevat söögikogust kasutajasõbralikult ja adekvaatselt hinnata. Sellele lisaks luua kaameravõimekus lindude ajakohaseks jälgimiseks. Samuti leiab autor, et puuri puhastamisest teavitamine võiks olla paremini lahendatud, see ei peaks tuginema ainult taimerile.

Kasutatud kirjandus

[1] „Raspberry Pi," [Võrgumaterjal]. Available: https://www.raspberrypi.org/help/what-%20is-a-raspberry-pi/. [Kasutatud 30.10.2019].

[2] „After Dawn," [Võrgumaterjal]. Available: https://www.afterdawn.com/news/article.cfm/2012/05/20/raspberry_pi_mini-pc_getting_camera_module. [Kasutatud 30.10.2019].

[3] „Raspberry Pi," [Võrgumaterjal]. Available: https://projects.raspberrypi.org/en/projects/raspberry-pi-setting-up/2. [Kasutatud 30.10.2019].

[4] „Sparkfun," [Võrgumaterjal]. Available: https://learn.sparkfun.com/tutorials/what-is-an-arduino/all. [Kasutatud 30.10.2019].

[5] „Random Nerd Tutorials," [Võrgumaterjal]. Available: https://randomnerdtutorials.com/getting-started-with-esp8266-wifi-transceiver-review/. [Kasutatud 30.10.2019].

[6] „Last Minute Engineers," [Võrgumaterjal]. Available: https://lastminuteengineers.com/ds1307-rtc-arduino-tutorial/. [Kasutatud 01.11.2019].

[7] „Parallax," [Võrgumaterjal]. Available: https://forums.parallax.com/discussion/83298/difference-between-ds1307-and-ds1302. [Kasutatud 01.11.2019].

[8] „Adafruit," [Võrgumaterjal]. Available: https://learn.adafruit.com/dht/overview. [Kasutatud 01.11.2019].

[9] „Random Nerd Tutorials," [Võrgumaterjal]. Available: https://randomnerdtutorials.com/getting-started-micropython-esp32-esp8266/. [Kasutatud 01.11.2019].

[10] "Upwork," [Võrgumaterjal]. Available: https://www.upwork.com/hiring/development/c-vs-c-plus-plus/. [Kasutatud 01.11.2019].

[11] „SMTP2GO," [Võrgumaterjal]. Available: https://www.smtp2go.com/blog/free-smtp-server-offered-smtp2go/. [Kasutatud 03.11.2019].

[12] „Trustpilot," [Võrgumaterjal]. Available: https://www.trustpilot.com/review/www.smtp2go.com. [Kasutatud 03.11.2019].

[13] „Capterra,“ [Võrgumaterjal]. Available: https://www.capterra.com/p/166646/SMTP2GO/. [Kasutatud 03.11.2019].

[14] „Element14,“ [Võrgumaterjal]. Available: https://www.element14.com/community/people/neilk/blog/2019/02/22/send-an-email-from-esp8266. [Kasutatud 16.12.2019].

[15] „Arduino,“ [Võrgumaterjal]. Available: https://www.arduino.cc/en/main/software. [Kasutatud 08.11.2019].

[16] „Isetegija,“ [Võrgumaterjal]. Available: http://isetegija.ee/esp8266-wifi-mooduli-programmeerimine/. [Kasutatud 08.11.2019].

[17] „educ8s.tv,“ [Võrgumaterjal]. Available: https://educ8s.tv/arduino-realtime-ds1302/. [Kasutatud 22.11.2019].

[18] „Github,“ [Võrgumaterjal]. Available: https://github.com/esp8266/Arduino/issues/4005. [Kasutatud 22.11.2019].

[19] „Losant,“ [Võrgumaterjal]. Available: https://www.losant.com/blog/getting-started-with-the-esp8266-and-dht22-sensor. [Kasutatud 23.11.2019].

[20] „Random Nerd Tutorials,“ [Võrgumaterjal]. Available: https://randomnerdtutorials.com/monitor-your-door-using-magnetic-reed-switch-and-arduino/. [Kasutatud 15.12.2019].

[21] „SMTP2GO,“ [Võrgumaterjal]. Available: https://www.smtp2go.com/setup/. [Kasutatud 23.11.2019].

[22] „Samlogic Software,“ [Võrgumaterjal]. Available: https://www.samlogic.net/articles/smtp-commands-reference.htm. [Kasutatud 24.11.2019].

[23] „Electronics Hub,“ [Võrgumaterjal]. Available: https://www.electronicshub.org/send-an-email-using-esp8266/. [Kasutatud 16.12.2019].

[24] „StackExchange,“ [Võrgumaterjal]. Available: https://arduino.stackexchange.com/questions/43556/how-to-set-the-delay-on-for-7-hours. [Kasutatud 03.01.2020].

Lisa 1 – Lahenduse lõppkood

```
#include <ESP8266WiFi.h>
#include <DHT.h>
#include <DS1302.h>

#define DHTPIN 5
#define DHTTYPE DHT22

DHT niiskusTemp (DHTPIN, DHTTYPE);
DS1302 kell (04, 00, 02);
WiFiClient esp;

Time t1; // mille vastu võrreldakse
Time t2; //vee riknevuse jaoks
Time t3; //puuri puhastamine

//wifi parameetrite määramine
const char* wifiSSID = "Elion-5E10A6";
const char* wifiPSW = "xxx";
//GPIO jalad
int herkon = 14;
int elektrood = 12;

void setup() {
  Serial.begin(9600);
  niiskusTemp.begin();
  //rtc.setDOW(FRIDAY);
  //rtc.setTime(21, 18, 0);
  //rtc.setDate(15, 11, 2019);
  t2 = kell.getTime(); //veega arveldamiseks
  t3 = kell.getTime(); //puuriga arveldamiseks
  //jalgade funktsionaalsuse määramine
  pinMode(herkon, INPUT);
  pinMode (elektrood, INPUT);
  WiFi.begin(wifiSSID, wifiPSW);
  //kuni ühendus ei ole saavutatud, prindi punkt
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  //kinnituseks antakse IP-aadress
  Serial.print("Ühendatud wifivõrku nimega ");
  Serial.print(wifiSSID);
  Serial.print(". IP address ");
```

```
  Serial.println(WiFi.localIP());
}
void loop() {
  t1 = kell.getTime(); //reaalne aeg
  tempPuudu();
  temperatuurLiiga();
  veeKontroll();
  puuriKontroll();

  //Söögitaseme lugemine
  if (digitalRead(herkon) == HIGH) {
    Serial.println("Toit hakkab otsa saama, saadan meili!");
    //meilToitOtsas();
  }
  else if (digitalRead(herkon) == LOW) {
    Serial.println("Toit olemas!");
  }
  //Veetaseme lugemine
  if (digitalRead(elektrood) == HIGH) {
    Serial.println("Jook otsas, saadan meili!");
    byte ret = meilVesiOtsas();
    t2 = kell.getTime(); // veelugemise taimer hakkab otsast peale käima
  }
  else if (digitalRead(elektrood) == LOW) {
    Serial.println("Vesi olemas");
  }
  delay(3600000); //1h
}
byte tempPuudu() {
  float niiskus = niiskusTemp.readHumidity();
  float temperatuur = niiskusTemp.readTemperature();

  while (isnan(temperatuur) || isnan(niiskus)) {
    Serial.println("DHT22 sensorilt andmete lugemine ebaõnnestus!");
    Serial.println("Saadan meili!");
    if (esp.connect(server, 2525) == 1) {
      Serial.println(F("Ühendus SMTP2GO serveriga loodud!"));
    }
    else {
      Serial.println(F("Ühendamine ebaõnnestus!"));
      return 0;
    }
    Serial.println(F("Sending EHLO..."));
    esp.println("EHLO www.headphones.com");
    Serial.println(F("Sending authentication request..."));
    esp.println("AUTH LOGIN");
    Serial.println(F("Sending user name..."));
    esp.println("xxx"); //kasutajanimi base-64 kodeeringus
    Serial.println(F("Sending password..."));
    esp.println("xxx"); //parool base-64 kodeeringus
    Serial.println(F("Sending from srxwrc@gmail.com"));
```

```
    esp.println(F("MAIL From: srxwrc@gmail.com"));
    Serial.println(F("Sending to srxwrc@gmail.com"));
    esp.println(F("RCPT To: srxwrc@gmail.com"));
    Serial.println(F("Sending the data part..."));
    esp.println(F("DATA"));
    esp.println(F("To: srxwrc@gmail.com"));
    esp.println(F("From: srxwrc@gmail.com"));
    esp.println(F("Subject: Näitude lugemine ebaõnnestus!"));
    esp.println(F("DHT22 sensorilt temperatuuri- ja niiskusenäitude lugemine
ebaõnnestus!"));
    esp.println(F("."));
    Serial.println(F("Sending quitting request..."));
    esp.println(F("QUIT"));
    esp.stop();
    Serial.println(F("Disconnected from the server."));
  }
}
//kontrollib, kas temperatuur on üle 27 v alla 15
byte temperatuurLiiga() {
  float niiskus = niiskusTemp.readHumidity();
  float temperatuur = niiskusTemp.readTemperature();
  if ((temperatuur >= 27.00) || (temperatuur <= 15.00)) {
    Serial.println("HÄIRE! TEMPERATUUR ON ÜLE 27 KRAADI/ ALLA 15 KRAADI!");
    Serial.println("Saadan meili!");
    if (esp.connect(server, 2525) == 1) {
      Serial.println(F("Ühendus SMTP2GO serveriga loodud!"));
    }
    else {
      Serial.println(F("Ühendamine ebaõnnestus!"));
      return 0;
    }
    Serial.println(F("Sending EHLO..."));
    esp.println("EHLO www.headphones.com");
    Serial.println(F("Sending authentication request..."));
    esp.println("AUTH LOGIN");
    Serial.println(F("Sending user name..."));
    esp.println("xxx");
    Serial.println(F("Sending password..."));
    esp.println("xxx");
    Serial.println(F("Sending from srxwrc@gmail.com"));
    esp.println(F("MAIL From: srxwrc@gmail.com"));
    Serial.println(F("Sending to srxwrc@gmail.com"));
    esp.println(F("RCPT To: srxwrc@gmail.com"));
    Serial.println(F("Sending the data part..."));
    esp.println(F("DATA"));
    esp.println(F("To: srxwrc@gmail.com"));
    esp.println(F("From: srxwrc@gmail.com"));
    esp.println(F("Subject: Temperatuur liiga kõrge/madal!"));
    esp.println(F("Lindude juures on temperatuur liiga kõrge/madal!"));
    esp.print(F("Hetketemperatuur toas: "));
    esp.println(temperatuur);
```

```
    esp.print(F("Hetkeniiskusetase toas: "));
    esp.println(niiskus);
    esp.println(F("."));
    Serial.println(F("Sending quitting request..."));
    esp.println(F("QUIT"));
    esp.stop();
    Serial.println(F("Disconnected from the server."));
    Serial.println("Disconnected from the server.");
  }
}
//vee reostuse kontroll taimeri näol, iga kahe päeva tagant
byte veeKontroll() {
  if (t2.dow == 0 && t1.dow == 2) {
    Serial.println("Pühapäev lisati vett, täna on teisipäev, vaja vett
vahetada, saadan meili!");
    byte ret = meilVesiVahetada();
  }
  else if (t2.dow == 1 && t1.dow == 3) {
    Serial.println("Esmaspäev lisati vett, täna on kolmapäev, vaja vett
vahetada, saadan meili!");
    byte ret = meilVesiVahetada();
  }
  else if (t2.dow == 2 && t1.dow == 4) {
    Serial.println("Teisipäev lisati vett, täna on neljapäev, vaja vahetada
vett, saadan meili!");
    byte ret = meilVesiVahetada();
  }
  else if (t2.dow == 3 && t1.dow == 5) {
    Serial.println("Kolmapäev vahetati vett, täna on reede, vaja vahetada
vett, saadan meili!");
    byte ret = meilVesiVahetada();
  }
  else if (t2.dow == 4 && t1.dow == 6) {
    Serial.println("Neljapäev vahetati vett, täna on laupäev, vaja vahetada
vett, saadan meili!");
    byte ret = meilVesiVahetada();
  }
  else if (t2.dow == 5 && t1.dow == 0) {
    Serial.println("Reedel vahetati vett, täna on pühapäev, vaja vahetada
vett, saadan meili!");
    byte ret = meilVesiVahetada();
  }
  else if (t2.dow == 6 && t1.dow == 1) {
    Serial.println("Laupäev vahetati vett, täna on esmaspäev, vaja vahetada
vett, saadan meili!");
    byte ret = meilVesiVahetada();
  }
  else
    Serial.println("Vesi pole seisnud");
}
//puuri põhja vahetamise funktsioon, iga kuue päeva tagant
byte puuriKontroll() {
```

```
  if (t3.dow == 0 && t1.dow == 6) {
    Serial.println("Puuri puhastati pühap, täna on laup, vaja uuesti
puhastada, saadan meili!");
    t3 = kell.getTime();
    byte ret = meilPuurPuhastada();
  }
  else if (t3.dow == 1 && t1.dow == 0) {
    Serial.println("Puuri puhastati esmasp, täna on pühap, vaja uuesti
puhastada, saadan meili!");
    byte ret = meilPuurPuhastada();
    t3 = kell.getTime();
  }
  else if (t3.dow == 2 && t1.dow == 1) {
    Serial.println("Puuri puhastati teisip, täna on esmasp, vaja uuesti
puhastada, saadan meili!");
    byte ret = meilPuurPuhastada();
    t3 = kell.getTime();
  }
  else if (t3.dow == 3 && t1.dow == 2) {
    Serial.println("Puuri puhastati kolmap, täna on teisip, vaja uuesti
puhastada, saadan meili!");
    byte ret = meilPuurPuhastada();
    t3 = kell.getTime();
  }
  else if (t3.dow == 4 && t1.dow == 3) {
    Serial.println("Puuri puhastati neljap, täna on kolmap, vaja uuesti
puhastada, saadan meili!");
    byte ret = meilPuurPuhastada();
    t3 = kell.getTime();
  }
  else if (t3.dow == 5 && t1.dow == 4) {
    Serial.println("Puuri puhastati reede, täna on neljap, vaja uuesti
puhastada, saadan meili!");
    byte ret = meilPuurPuhastada();
    t3 = kell.getTime();
  }
  else if (t3.dow == 6 && t1.dow == 5) {
    Serial.println("Puuri puhastati laup, täna on reede, vaja uuesti
puhastada, saadan meili!");
    byte ret = meilPuurPuhastada();
    t3 = kell.getTime();
  }
  else
    Serial.println("Puur on puhas");
}
byte meilToitOtsas() {
  if (esp.connect(server, 2525) == 1) {
    Serial.println(F("Ühendus SMTP2GO serveriga loodud!"));
  }
  else {
    Serial.println(F("Ühendamine ebaõnnestus!"));
    return 0;
```

```
  }
  Serial.println(F("Sending EHLO..."));
  esp.println("EHLO www.headphones.com");
  Serial.println(F("Sending authentication request..."));
  esp.println("AUTH LOGIN");
  Serial.println(F("Sending user name..."));
  esp.println("xxx");
  Serial.println(F("Sending password..."));
  esp.println("xxx");
  Serial.println(F("Sending from srxwrc@gmail.com"));
  esp.println(F("MAIL From: srxwrc@gmail.com"));
  Serial.println(F("Sending to srxwrc@gmail.com"));
  esp.println(F("RCPT To: srxwrc@gmail.com"));
  Serial.println(F("Sending the data part..."));
  esp.println(F("DATA"));
  esp.println(F("To: srxwrc@gmail.com"));
  esp.println(F("From: srxwrc@gmail.com"));
  esp.println(F("Subject: Toidukoguse teavitus"));
  esp.println(F("Lindude söögianum hakkab tühjaks saama!"));
  esp.println(F("."));

  Serial.println(F("Sending quitting request..."));
  esp.println(F("QUIT"));
  esp.stop();
  Serial.println(F("Disconnected from the server."));
}
byte meilVesiOtsas() {
  if (esp.connect(server, 2525) == 1) {
    Serial.println(F("Ühendus SMTP2GO serveriga loodud!"));
  }
  else {
    Serial.println(F("Ühendamine ebaõnnestus!"));
    return 0;
  }
  Serial.println(F("Sending EHLO..."));
  esp.println("EHLO www.headphones.com");
  Serial.println(F("Sending authentication request..."));
  esp.println("AUTH LOGIN");
  Serial.println(F("Sending user name..."));
  esp.println("xxx");
  Serial.println(F("Sending password..."));
  esp.println("xxx");
  Serial.println(F("Sending from srxwrc@gmail.com"));
  esp.println(F("MAIL From: srxwrc@gmail.com"));
  Serial.println(F("Sending to srxwrc@gmail.com"));
  esp.println(F("RCPT To: srxwrc@gmail.com"));
  Serial.println(F("Sending the data part..."));
  esp.println(F("DATA"));
  esp.println(F("To: srxwrc@gmail.com"));
  esp.println(F("From: srxwrc@gmail.com"));
  esp.println(F("Subject: Lindude joogivesi hakkab otsa saama"));
```

```
  esp.println(F("Lindude joogianum on tühjenenud. Lisa vett!"));
  esp.println(F("."));
  Serial.println(F("Sending quitting request..."));
  esp.println(F("QUIT"));
  esp.stop();
  Serial.println(F("Disconnected from the server."));
}
byte meilPuurPuhastada() {
  if (esp.connect(server, 2525) == 1) {
    Serial.println(F("Ühendus SMTP2GO serveriga loodud!"));
  }
  else {
    Serial.println(F("Ühendamine ebaõnnestus!"));
    return 0;
  }
  Serial.println(F("Sending EHLO..."));
  esp.println("EHLO www.headphones.com");
  Serial.println(F("Sending authentication request..."));
  esp.println("AUTH LOGIN");
  Serial.println(F("Sending user name..."));
  esp.println("xxx");
  Serial.println(F("Sending password..."));
  esp.println("xxx");
  Serial.println(F("Sending from srxwrc@gmail.com"));
  esp.println(F("MAIL From: srxwrc@gmail.com"));
  Serial.println(F("Sending to srxwrc@gmail.com"));
  esp.println(F("RCPT To: srxwrc@gmail.com"));
  Serial.println(F("Sending the data part..."));
  esp.println(F("DATA"));
  esp.println(F("To: srxwrc@gmail.com"));
  esp.println(F("From: srxwrc@gmail.com"));
  esp.println(F("Subject: Puuri puhastamise teavitus"));
  esp.println(F("Viimasest puuri puhastamisest on 6 päeva möödas!"));
  esp.println(F("."));
  Serial.println(F("Sending quitting request..."));
  esp.println(F("QUIT"));
  esp.stop();
  Serial.println(F("Disconnected from the server."));
}
byte meilVesiVahetada() {
  if (esp.connect(server, 2525) == 1) {
    Serial.println(F("Ühendus SMTP2GO serveriga loodud!"));
  }
  else {
    Serial.println(F("Ühendamine ebaõnnestus!"));
    return 0;
  }
  Serial.println(F("Sending EHLO..."));
  esp.println("EHLO www.headphones.com");
  Serial.println(F("Sending authentication request..."));
  esp.println("AUTH LOGIN");
```

```
  Serial.println(F("Sending user name..."));
  esp.println("xxx");
  Serial.println(F("Sending password..."));
  esp.println("xxx");
  Serial.println(F("Sending from srxwrc@gmail.com"));
  esp.println(F("MAIL From: srxwrc@gmail.com"));
  Serial.println(F("Sending to srxwrc@gmail.com"));
  esp.println(F("RCPT To: srxwrc@gmail.com"));
  Serial.println(F("Sending the data part..."));
  esp.println(F("DATA"));
  esp.println(F("To: srxwrc@gmail.com"));
  esp.println(F("From: srxwrc@gmail.com"));
  esp.println(F("Subject: Vesi vahetamata kaks päeva"));
  esp.println(F("Lindude joogivesi on vahetamata olnud kaks päeva!"));
  esp.println(F("."));
  Serial.println(F("Sending quitting request..."));
  esp.println(F("QUIT"));
  esp.stop();
  Serial.println(F("Disconnected from the server."));
}
```