TARTU ÜLIKOOL
Arvutiteaduse instituut
Informaatika õppekava

Marvin Maran

Arduino platvormiga ühilduvad andurid ja liidesed ning nuputelefoni ehitamine

Bakalaureusetöö (9 EAP)

Juhendajad: Alo Peets
Anne Villems
Taavi Duvin

Tartu 2017

Arduino platvormiga ühilduvad andurid ja liidesed ning nuputelefoni ehitamine

Lühikokkuvõte:

Käesoleva bakalaureusetöö eesmärgiks on tutvustada mikrokontrolleritega ühilduvaid andureid ja liidesed Arduino näitel ning kirjeldada, kuidas ehitada Arduinole nuputelefon GSM-mooduli abil. Töö esimeses peatükis tutvustatakse Arduino platvormi. Teises peatükis jagatakse andurid väljundsignaali järgi kaheks. Kolmandas peatükis tutvustatakse nelja liidest ja protokolli, mida andurites ning muus riistvaras kasutatakse. Neljandas peatükis on juhend Arduino nuputelefoni ehitamiseks.

Võtmesõnad:

Arduino, andurid, robootika

CERCS: P170 (Arvutiteadus, arvanalüüs, süsteemid, kontroll)

Arduino compatible sensors and interfaces and building a button phone

Abstract:

The aim of this bachelor thesis is to introduce sensors and interfaces that are compatible with microcontrollers on the example of Arduino and how to build a button phone on Arduino with GSM-module. The first chapter of the thesis introduces Arduino platform. In the second chapter sensors are grouped by output signal. The third chapter describes four interfaces and protocols that are used in sensors and hardware. The fourth chapter contains instructions how to build a button phone on Arduino.

Keywords:

Arduino, sensors, robotics

CERCS: P170 (Computer science, numerical analysis, systems, control)

Sisukord

Sissejuhatus 5

1. Arduino üldine kirjeldus 6

1.1. Arduino tutvustus 6

1.2. Arduino Uno 6

1.3. Arduino Uno pesad 7

1.4. Arduino arenduskeskkond 9

1.5. Arduino Uno arvutiga ühendamine ja töökorda seadmine 10

1.6. Arduino teegid 13

1.6.1. Teekide struktuur 14

1.6.2. Teekide kasutamine 14

2. Andurid 16

2.1. Analoogandurid 16

2.1.1. Analoogsignaal 16

2.1.2. Analoogandurite suhtlus arvuti ja Arduinoga 17

2.1.3. Analoogsignaalil töötav fototakisti 18

2.1.4. Programm fototakistilt loetud näidu kuvamiseks 19

2.2. Digitaalandurid 21

2.2.1. Digitaalsignaal 21

2.2.2. Digitaalandurite suhtlus arvuti ja Arduinoga 22

2.2.3. Digitaalsignaalil töötav alkoholiandur 23

2.2.4. Programm alkoholiandurilt näidu lugemiseks 23

3. Liidesed ja protokollid 26

3.1. Paralleelühendus 26

3.1.1. Paralleelühenduse põhimõte 26

3.1.2. Paralleelühendusel töötav numbrimaatriks 27

3.2. Jadaedastus 28

3.2.1. Jadaedastuse põhimõte ... 29
3.2.2. Asünkroonne jadaedastus ... 29
3.2.3. Riistvara ja selle ühendamine asünkroonse jadaedastuse jaoks ... 31
3.2.4. Jadaedastusel töötav GPRS Shield ... 33
3.3. Serial Peripheral Interface ... 36
3.3.1. Sünkroonne liides ... 36
3.3.2. Ühe või mitme alluvaga suhtlemine ... 37
3.3.3. SPI liidesega atmosfääriandur ... 38
3.4. The Inter-Integrated Circuit Protocol ... 41
3.4.1. I^2C põhimõte ... 41
3.4.2. Protokoll ... 41
3.4.3. I^2C liidesega OLED ekraan ... 42
4. Nuputelefoni ehitamine GSM-mooduli abil ... 45
4.1. Idee ning sarnased projektid ... 45
4.2. Nõuded nuputelefonile ... 45
4.3. Vajalikud komponendid ... 46
4.3.1. GPRS Shieldi kasutamine SIM900 teegiga ... 47
4.3.2. RGB valgusdiood ... 48
4.4. Riistvara ühendamine ... 49
4.5. Nuputelefoni programmeerimine ... 51
4.6. Tulemused ... 54
Kokkuvõte ... 56
Viidatud kirjandus ... 57
Lisad ... 61
I. Nuputelefoni kood ... 61
II. Litsents ... 67

Sissejuhatus

Tänapäeva maailmas on lihtne enda elu mugavamaks muutmiseks soetada valmis seadmeid ja tooteid, mille kasutamiseks on vaja vaid kasutusjuhend läbi lugeda ning seejärel seade sisse lülitada. Paljud sellised elektroonikaseadmed, nagu näiteks taimeritega valgustid ja automaatsed küttesüsteemid, on üpriski lihtsa ehituse ja tarkvaraga. Järjest rohkem luuakse platvorme ja tarkvarasid, millega on võimalik neid seadmeid endale valmistada ja seejuures kordades odavamalt võrreldes poest ostetutega. Seega on selliseid võimalusi mõistlik luua koolidesse, et tutvustada kooliõpilastele infotehnoloogilisi lahendusi neile jõukohaste vahendite ja projektidega.

Üheks selliseks platvormiks on Arduino, mille programmeerimiseks vajalik tarkvara on vabavaraliselt saadav ja arendusplaat tuleb lihtsalt endale juurde osta. Lisaks toodetakse järjest rohkem ja erinevaid Arduinoga kokkusobivaid andureid ning mooduleid. Arduino stardikomplekti, mikrokontrollerit ja mitmeid andureid on Eestis võimalik osta elektroonikapoodidest, näiteks Oomipood. Välismaistest internetipoodidest saab soetada endale kõiki toodetud komponente väga taskukohase hinnaga. Võimalused nendest midagi valmistada on tohutud, kuid sageli ei jõua kõik see informatsioon kooliõpilasteni. Tihti ollakse arvamusel, et Arduino programmeerimine ning Arduino arendusplaatidele uute seadmete ehitamine käib õpilastel üle jõu. Seega on vaja tuua koolilasteni võimalus tutvuda Arduinoga, et saada aimu selle lihtsusest ja lõpututest võimalustest kasutades erinevaid andureid.

Käesoleva bakalaureuse töö eesmärgiks on uurida Arduinoga kokku sobivaid andureid ja mooduleid. Selgitada erinevate liideste ja protokollide kasutamist ning teha valmis näidisprojekt, milleks on nuputelefoni ehitamine.

Antud bakalaureusetöö koosneb viiest peatükist. Esimeses peatükis tutvustatakse Arduino platvormi. Täpsemalt vaadeldakse levinuimat arendusplaati Arduino Uno, selle seadistamist ning programmeerimist. Teises peatükis grupeeritakse andurid väljundsignaali järgi kaheks: analooganduriteks ja digitaalanduriteks. Kolmandas peatükis tutvustatakse digitaalsignaalil töötavat nelja levinumat suhtlusprotokolli ning liidest, tuuakse näiteid nende kasutamisest. Neljandas peatükis demonstreeritakse eelnevates peatükkides kasutatud kasutamist ehitades Arduino baasil klassikalise nuputelefoni.

1. Arduino üldine kirjeldus

Selles peatükis kirjeldatakse Arduino platvormi ning tutvustatakse arendusplaati Arduino Uno ning selle programmeerimist. Arduino platvormi on vaja tunda selleks, et saaks erinevaid andureid, mooduleid ja seadmeid Arduino Uno arendusplaadiga töötama panna.

1.1. Arduino tutvustus

Arduino on avatud lähtekoodiga riist- ja tarkvaraga platvorm. Arduino sündis Itaalia väikelinnas Ivrea projekti raames, kus seda kasutati kui lihtsat ja kiiret tööriista prototüüpimisel. Suunatud oli see õpilastele, kellel polnud elektroonika ja programmeerimisega varasemat kogemust. Arduino kogus palju populaarsust ning hakkas kasvama. Praegu on see levinud üle maailma kui kõige populaarsem "asjade interneti" ja prototüüpimise platvorm. Arduino on lihtne ning sobiv lähtekoht inimestele, kes pole varem elektroonika või programmeerimisega kokku puutunud. Arduinot kasutavad õpetajad, õpilased, disainerid, arhitektid, muusikud, programmeerijad ja paljud teised [1].

Arduino arendusplaadid on suhteliselt odavad ning kättesaadavad nii hästivarustatud Eesti elektroonikapoodidest kui ka internetist tellides. Arenduskeskkond ehk Arduino IDE (*Integrated Development Environment*) töötab Windows'i, Linux'i ja Mac operatsioonisüsteemidel. Programmeerimine toimub *Wiring* raamistikul C++ keeles mõningate muudatuste ja lihtsustustega [1, 2].

Arendusplaatide valik on väga suur. Neid toodetakse erinevas suuruses ja erinevate sisend-väljund viikude arvu ning lisadega. Üks populaarseimatest plaatidest alustamaks tööd Arduinoga on Arduino Uno, millest pikemalt kirjutatakse järgmises punktis [2].

1.2. Arduino Uno

Arduino Uno plaat on väga hea alustamaks robootika ja prototüüpmisega Arduino platvormil. Uno on mikrokontroller-plaat põhinedes mikrokontrolleril ATmega328P. Uno oli esimene plaat, millel on USB-ühenduse võimalus ning plaadi see tähistas Arduino Software (IDE) 1.0

väljalaset. Sel on 14 digitaalset sisend-väljund pesa, 6 analoogpesa, USB-pesa ja toitepesa [3]. Tabelis 1 on toodud välja Uno plaadi olulisemad tehnilised andmed.

Mikrokontroller	ATmega328P
Tööpinge	5V
Sisendpinge	7-12V (miinimum 6V ja maksimum 20V)
Digitaalpesad	14 (nendest 6 on väljundtoite võimekusega)
Analoogpesad	6
Välkmälu	32KB
Taktsagedus	16 MHz
Mõõtmed	68.6 x 53.4 mm

Tabel 1. Arduino Uno R3 spetsifikatsioon [3].

Arduino Unot kasutatakse selles töös anduritega töötamiseks, kuna ta on võimalikult lihtne ja väike ning enamus anduritest on Uno arendusplaadi toega. Arduino Unole leidub internetis ka kõige rohkem näiteid ja õpetusi, kuidas töötada erinevate andurite ja suhtlusprotokollidega. Järgmises punktis selgitatakse Arduino Uno pesade funktsionaalsusi.

1.3. Arduino Uno pesad

Arduino Uno arendusplaadil on 14 digitaalsignaali ja 6 analoogsignaali lugemise võimekusega pesa. Lisaks on veel pesad sisend-väljund toitepinge jaoks. Siin punktis kirjeldataksegi plaadi pesade funktsionaalsusi. Pesade märgistused on võetud jooniselt 1.

a) **IOREF** (*Input-Output Reference*) pesa kaudu saab pakkuda mikrokontrollerile toitepinget. Plaat saab lugeda IOREF pesa kaudu tuleva pinge suurust ning seega valida sobiva toiteallika või aktiveerida pingetranslaatorid 5V või 3.3V väljunditel [3].
b) **RESET** võimaldab taaskäivitada mikrokontrollerit, kui selle pesa väärtus viia LOW-i[1] [3].

[1] LOW/HIGH – loogilise digitaalsignaali võimalikud väärtused, LOW=0 ja HIGH=1

c) **3.3V** tähendab 3.3 voldist väljundpingeallikat maksimaalse voolutarbega 50mA [3].

d) **5V** tähendab 5 voldist väljundpingeallikat, mis on reguleeritud plaadilt. Toidet saab kas alalisvoolu toitepesast (7-12V), USB-pesast (5V) või VIN (*Input Voltage*) pesast (7-12V) [3].

e) **GND** (*ground*) ehk maandus või miinusjuhe elektriühendustes.

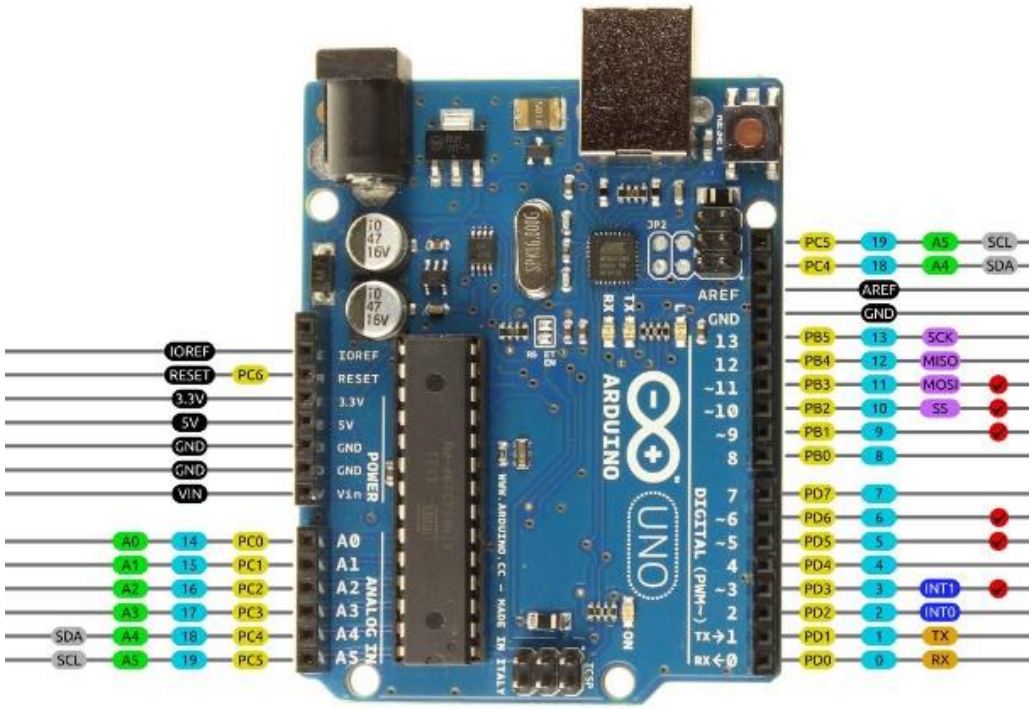
f) **VIN** pesa kaudu saab anda plaadile sisendpinget, kui see kasutab välist toiteallikat [3].

g) **A0 - A5** on analoogpesad, mis mõõdavad vaikimisi analoogsignaali vahemikus 0-5 V. Selle vahemiku ülemist piiri on võimalik muuta kasutades **AREF** pesa ja *analogReference()* funktsiooni [4]. Samuti on kõigil neil olemas ka digitaalsignaali võimekus.

h) **0 - 13** on digitaalsignaali sisend-väljund võimekusega pesad [3]:
    a. **0** ja **1** on RX ja TX ehk ressiiver ja transmitter. Nende pesade kaudu töötab andmete jadaedastus.
    b. **2** ja **3** on välise katkestuste kasutamise jaoks.
    c. **3, 5, 6, 9, 10** ja **11** võimaldavad riistvaralise pulsilaiusmodulatsiooni võimekust *analogWrite()* funktsiooniga.
    d. **10 (SS[2]), 11 (MOSI[3]), 12 (MISO[4]), 13 (SCK[5])** on pesad, mis võimaldavad SPI (*Serial Peripheral Interface*) suhtlust kasutades SPI teeke.
    e. **13** – Viigu külge on ühendatud statsionaarne valgusdiood ehk märgutuli, mida saab kasutada koodi testimiseks ja analüüsik. Kui pesa väärtus on HIGH, siis LED põleb, kui LOW, siis on kustus.

i) **SCL** ja **SDA** võimaldavad I^2C suhtlust. Samuti võimaldavad seda **A4** ja **A5**, mis tegelikult ongi SCL ja SDA-ga samad pesad, sest need on omavahel ühenduses[3].

[2] SS (Slave-Select) – SS kaudu öeldakse alluvale seadmele, et aeg on infot vastu võtta/saata.
[3] MOSI (Master-Out/Slave-In) - MOSI kaudu saadetakse andmeid ülemalt alluvale.
[4] MISO (Master-In/Slave-Out) - MISO kaudu saadab alluv ülemale andmeid vastu.
[5] SCK (Clock From Master) - SCK kaudu tuleb ülemalt taktsagedus ka alluvale.



Joonis 1. Arduino Uno pesade märgistused ja aadressid [4].

Arendusplaadi riistvaralised näitajad ning pesade võimekused on üldisel tasemel kirjeldatud. Järgmises punktis hakatakse tutvustama Arduino arenduskeskkonda.

1.4. Arduino arenduskeskkond

Nüüdseks on meil teada, mis on Arduino ning oleme ka tutvunud ühe Arduino perekonna esindaja Arduino Unoga. Selleks, et nüüd arendusplaati mõistlikult kasutada, on vaja seda programmeerida. Arduino programmeerimiseks on kaks põhilist võimalust - veebiversioon ja allalaetav tarkvara.

Veebiversioon on *Arduino Web Editor*. Seda saab kasutada otse brauserist aadressil https://create.arduino.cc/editor . Vajalik on teha endale kasutajakonto vajutades *Sign Up* lingile ning täites registreerimisvorm ja seejärel konto aktiveerida. Samuti tuleb installeerida Arduino Create pistikprogramm veebilehitsejasse. Veebiversioonis salvestatakse kõik kirjutatud programmid pilveserverisse ning on seetõttu kättesaadavad igast internetiühendusega arvutist. Samuti on veebiversioonis alati uusim tarkvara ning teegid [5].

Arduin IDE on allalaaditav arenduskeskond, mis on tasuta tarkvara. Selle saab hankida Arduino ametlikult kodulehelt www.arduino.cc valides menüüribalt *Software*. Avanenud aknast valida enda operatsioonisüsteemile vastav paigaldussfail. See arvutisse alla laadida ning installeerida Arduino IDE koos kõikide täiendavate komponentidega, mis tagab, et hiljem programmeerides ei oleks vaja hakata uuesti puuduolevaid osi installeerima. Programmi esmasel käivitamisel avaneb sarnane vaatepilt nagu joonisel 2.

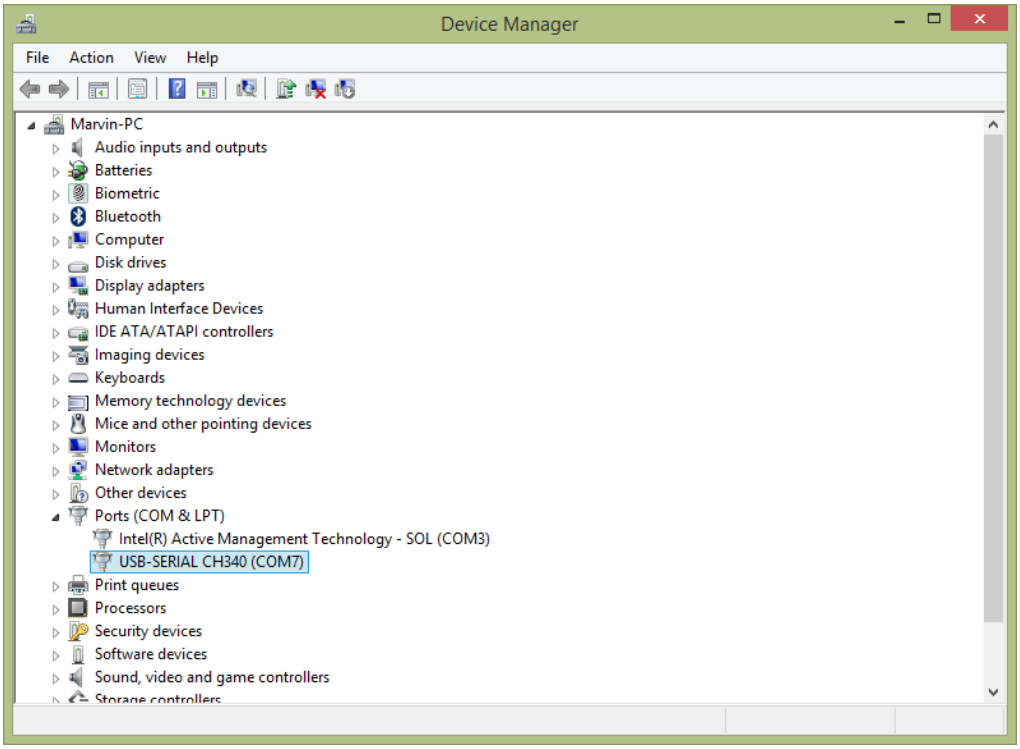
Joonis 2. Arduino IDE avatuna.

Käesolevas töös kasutatakse allalaetavat programmeerimiskeskkonda Arduino IDE, kus toimub edaspidine koodikirjutamine. Järgnevalt tuleb ühendada arendusplaat arvutiga ja teha vajalikud seadistused, et Arduino arenduskeskonnas kirjutatud programmi saaks arendusplaadis kasutada.

1.5. Arduino Uno arvutiga ühendamine ja töökorda seadmine

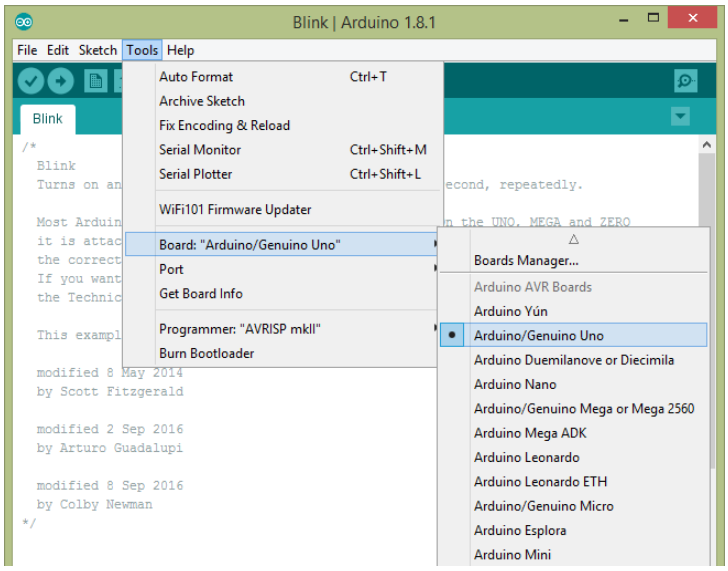
Arduino Uno plaadi arvutiga ühendamiseks on vaja USB-ühendust, seega tuleb Micro-USB otsaga juhe ühendada plaadiga ning teine ots panna arvutis vabasse USB-porti. Osad

arendusplaadid kasutavad ka Mini ja B tüüpi USB-ühendusi. Õige ühenduse korral hakkavad plaadil põlema punane sisseehitatud valgusdiood märgistusega ON. Seejärel tuleb lasta arvutil installeerida vajalik draiver ning otsida *Device Manager*'ist ehk seadmehaldurist, millise pordi külge plaat on ühendatud (joonis 3). Seadmehalduri leiab vajutades paremklõpsu tööribal oleva Windowsi logo peal või kirjutades stardimenüü otsingusse "Device Manager". Õiget porti tuleb otsida avanenud seadmehaldurist *Ports (COM & LPT)* jaotise alt (joonis 3). Originaal Arduino plaadid on kindlasti eristatavad nime järgi. Kui nime järgi pole võimalik kindlaks teha, milline port on ühenduses plaadiga, siis tuleks USB-kaabel ühendada lahti ja uuesti ühendada ning vaadata, milline nimi tekkis portide nimekirja juurde. Pordi number on nimes kujul COMx, kus x tähistab pordi numbrit.



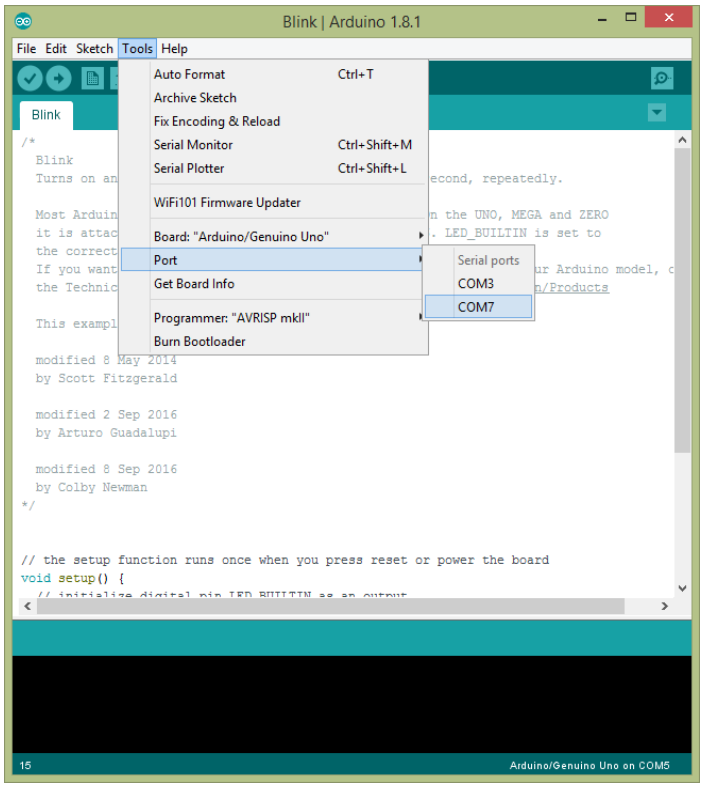
Joonis 3. Õige pordi leidmine programmist *Device Manager*.

Järgnevalt seadistame Arduino IDE vastavalt kasutatavale arendusplaadile, kuna vastasel juhul ei hakka plaat kirjutatud programmiga tööle. Valime menüüst *Tools* ja sealt *Board* rippmenüüst õige plaadi nimetuse. Kuna kasutame Arduino Unot, siis valime *Arduino/Genuino UNO* (joonis 4).



Joonis 4. Õige plaadi valimine Arduino IDE-st.

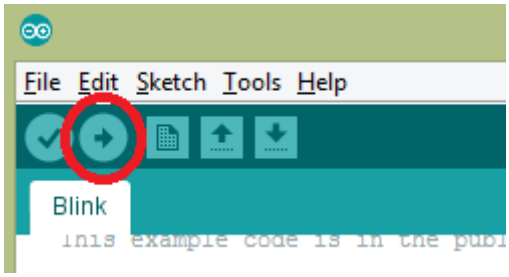
Seejärel seadistame õigeks pordi, kuhu arendusplaat on arvuti külge ühendatud, valides *Tools* menüüst *Port*. Õige pordi leidsime eelnevalt seadmehaldurist ning paneme linnukese selle pordi ette (joonis 5). Tuleb valida õige port, kuna muidu ei suuda Arduino IDE tuvastada arendusplaati ning ei tea, kuhu pärast programm üles laadida.



Joonis 5. Õige pordi märkimine Arduino IDE-s.

Kui kõik eelnev on tehtud probleemideta, käivitame mõne näiteprogrammi, veendumaks, et kõik töötab. Arduino IDE-ga on kaasas mitmeid näiteprogramme. Nendeni jõuab valides menüüst *File -> Examples*. Valime sealt *01.Basics* alt *Blink* programmi, mis paneb plaadil oleva valgusdioodi tule vilkuma. Avanenud aknas näeme *Blink* programmi koodi. Kood on kirjutatud keeles C/C++ ning Arduino programmi töötamiseks on vaja ainult kahe funktsiooni defineerimist: *setup()* ja *loop()* [6]. Funktsioon *setup()* töötab ainult programmi algul ning selles tehakse ühekordsed tegevused nagu sisend-väljundviikude ja siinide initsialiseerimine. Funktsiooni *loop()* tuleb kirjutada programmi sisu ning seda kutsutakse välja korduvalt kuni plaadi väljalülitamiseni.

Kuna hetkel on programm juba ette kirjutatud, siis tuleb see ainult plaadi mälusse üles laadida ja käivitada. Selleks tuleb vajutada *Upload* nuppu (joonis 6). Kui üles laadimine on lõpetatud, siis hakkab valgusdiood vilkuma ühe sekundilise intervalliga.



Joonis 6. Programmi üleslaadimine mikrokontrollerisse.

Sageli ei piisa ainult enda kirjutatud koodist, et programm koos riistvaraga korrektselt töötaks. Keerukamate anduritega suhtlemiseks on vaja kasutada erinevaid teeke, mida tutvustatakse järgmises punktis.

1.6. Arduino teegid

Arduino keskkonnas töötamist saab lihtsustada teekide (inglise keeles *library*) kasutamisega. Teekidega saab lisada funktsionaalsust töötamisel erineva riistvaraga. Näiteks omavad paljud andurid teeke, mis lihtsustavad nende kasutamist oluliselt. Arduino ametlikud standardteegid on juba Arduino IDE-s olemas, kui arenduskeskkonna installeerimise ajal vajalik linnuke teekide installeerimise kohta sai tehtud. Arduino kasutajad kirjutavad ja modifitseerivad ka ise erinevaid teeke, mis laetakse internetti, et teised kasutajad võiksid neid samuti kasutada

[7]. Teekidel on kindel struktuur, et nende kasutamine teha kasutajatele arusaadavaks ja lihtsaks. Teekide struktuuri tutvustatakse järgmises punktis.

1.6.1. Teekide struktuur

Teegid koosnevad vähemalt kahest failist: päisefailist (inglise keeles *header file*) ja koodifailist (inglise keeles *code file/source file*). Päisefail lõpeb laiendiga .h ning seal sisalduvad selle teegiklassi definitsioonid: klassikonstruktorid, muutujanimed, funktsioonid. Koodifail lõpeb laiendiga .cpp ning sinna kirjutatakse kõikide päisefailis defineeritud funktsioonide reaalsed implementatsioonid. Failinimed nimetatakse klassinime järgi. Näiteks, kui on defineeritud klass AndurX, siis failid oleksid AndurX.h ja AndurX.cpp. Enamus teeke ei koosne ainult nendest kahest failist, sest keerulisemate teekide puhul läheksid need kaks faili liiga suureks ning nendest oleks raske midagi üles leida. Samuti võtaksid arendusplaadi mälus asjatult palju ruumi. Suurte teekide puhul on üks ülemklass ning mitu alamklassi, igal klassil on eraldi .h ja .cpp fail. Jagamine on tehtud loogiliselt ning erinevate funktsionaalsuste jaoks. Ühe teegi failid asetsevad teegiga samanimelises kaustas, kuhu need on eelnevalt imporditud [8]. Teekide kasutamist praktikas kirjeldatakse järgmises punktis.

1.6.2. Teekide kasutamine

Arduino IDE-s kaasas olevaid teeke saab kasutada valides menüüst *Sketch -> Include Library*. Avanenud nimekirjast valida vajalik teek. Koodi päisesse lisati nüüd read valitud teegi kasutamise kohta vormis *#include <teek.h>*, mis tähendab, et selles programmis saab kasutada teegis defineeritud funktsioone.

Alati ei saa hakkama juba olemasolevate teekidega. Keerukamate ja uuemate andurite ja lisamoodulitega töötamise jaoks tuleb teegid kas ise kirjutada või internetist otsida. Internetist leiab palju õpetusi ja näiteid teekide kasutamise kohta. Üldiselt on erinevate õpetuste juures vajalike teekide hankimise jaoks olemas allalaadimislingid. Kui teek on arvutisse alla laaditud, siis tuleb see Arduino IDE-sse importida. Menüüst valida *Sketch -> Include Library* ja avanenud nimekirjast vajutada *Add .ZIP Library*. Avanenud aknast navigeerida kausta, kus alla laaditud teek paikneb ning see lisada. Nüüd on see teek lisandunud teekide nimekirja ja saame seda kasutada nii nagu eelmises lõigus kirjeldatud.

Teegi .h failist saame uurida, missugused funktsioonid seal leiduvad ning kuidas neid kasutada. Enamasti on teekidega kaasas ka näiteprogrammid või õpetused.

Järgnevalt, et Arduino Unoga edasi töötada ja suuremaid riist- ja tarkvaralisi projekte teha vaja, on vajalik kasutada erinevaid lisaseadmeid. Üheks võimaluseks on kasutada andureid, mida saab jagada laias laastus kaheks: analooganduriteks ja digitaalanduriteks. Analoog- ja digitaalanduritest kirjutatakse järgmises peatükis.

2. Andurid

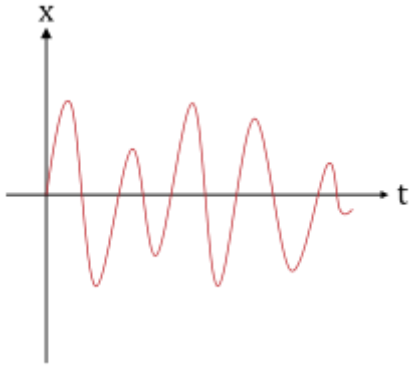
Andur on elektrooniline seade, mis mõõdab väliskeskkonnas toimuvaid muutusi ning saadab mõõdetud informatsiooni mõnda teise elektroonikaseadmesse. Käesolevas peatükis jagatakse andurid kahte gruppi: analoogandurid ja digitaalandurid. Selgitatakse, mis on analoog- ja digitaalsignaal ning kuidas analoog- ja digitaalandureid kasutada Arduinoga. Samuti tuuakse mõlemast grupist näide ühe anduri kohta.

2.1. Analoogandurid

Analoogandurid saadavad informatsiooni analoogsignaalina. Enamasti on analoogandurid küllaltki lihtsa ehitusega ja anduriga suhtlemiseks peab lugema ainult ühte analoogsignaali [9]. Esmalt teeme selgeks, mis on analoogsignaal ning seejärel kuidas analoogandureid kasutada.

2.1.1. Analoogsignaal

Analoogsignaal on pidev signaal, kuna see on igal ajahetkel mõõdetav. Terve meid ümbritsev maailm on analoogsignaale täis, kuna enamik looduslikke ja tehislikke protsesse on toimelt pidevad [10]. Protsesse, mida saab edastada analoogsignaaliga on näiteks elektrivool, helilained, temperatuur ja rõhk. Analoogsignaali iseloomustab lõpmatus, kuna sel on lõpmata arv olekuid ning on mõõdetav igal ajahetkel [10]. Joonisel 7 on näidatud analoogsignaali graafik. Joonisel telg t ehk aeg ja telg x ehk amplituud võivad olla lõpmatu suured väärtused olenevalt sellest, mida mõõdetakse. Samuti võib amplituudi telg olla fikseeritud maksimum- ja miinimumväärtusega, mis siiski annab lõpmatu arvu väärtuseid selles vahemikus, kuna kasutades reaalarve, saame jaotada vahemiku lõpmata väikesteks osadeks [11]. Analoogsignaali iseloomustab sujuvate käänupunktidega graafik [11].



Joonis 7. Analoogsignaali graafik [10].

Analoogsignaali lõpmata arv olekuid ja mõõdetavus igal ajahetkel moodustavad lõpmatu resolutsiooni, seega on tema suurimaks eeliseks väga täpse info. Lõpmatusest tuleneb ka signaali suur tihedus [10]. Suur tihedus ja lõpmatu arv väärtusi toovad kaasa ka puudusi. Üheks selliseks puuduseks on müra tekkimine väljundisse. Antud kontekstis tähendab müra korrapäratute või korrapäraste anomaaliate tekkimist. Müra tekib signaali ülitundlikkusest ehk salvestatakse iga väiksemgi signaalimuutus, mis tegelikult ei mängi üldse suurt rolli.

Nüüd, kui on aimu, mis on analoogsignaal, saame asuda analooganduritega töötamise juurde.

2.1.2. Analoogandurite suhtlus arvuti ja Arduinoga

Järgnev punkt on kirjutatud internetiallika [9] põhjal.
Analoogandur mõõdab väliskeskkonnast mingi informatsiooni ning saadab selle informatsiooni analoogsignaalina anduriga ühendatud seadmesse. Käesolevas töös on andurid ühendatud Arduino Unoga. Seega peab Arduino Uno olema võimeline lugema analoogsignaali ning selle endale arusaadavaks tegema. Et analoogsignaal oleks protsessorile arusaadav, tuleb analoogsignaal konverteerida digitaalsignaaliks. Selle jaoks on plaadil olemas analoog-digitaalmuundur ehk ADC (*analog-to-digital converter*).

Arduino Uno kasutab 10-bitist analoog-digitaalmuundurit. ADC bitilisus tähendab seda, kui täpselt muundur analoogsignaali digitaalsignaaliks muudab. 10-bitti tähendab seda, et analoogsignaali mõõtevahemik jagatakse 10-bitisesse vahemikku. 10-bitises vahemikus on 2^{10} = 1024 erinevat väärtust. Analoogsignaali mõõdetakse Arduino Unol vahemikus 0 - 5 volti, kuna Uno toitepinge on 5 volti. Vahemik 0 - 5 volti jagatakse 1024 erinevaks väärtuseks, seega 5V / 1024 = 4.88mV on ADC täpsus. Et saada suuremat täpsust, tuleb kasutada kõrgema bitisusega analoog-digitaalmuundurit.

Analoogsignaali lugemiseks on vaja Arduino Uno vastavalt programmeerida ning analooganduriga ühendada. Analoogsignaali suudavad lugeda Arduino Uno plaadil ühendusviigud aadressiga A0 - A5, mis tähendab, et analoogandur tuleb ühega neist viikudest ühendada. Arduino arenduskeskkonnas tuleb arendusplaat programmeerida ühendatud viigult signaali lugema. Esmalt tuleb defineerida, milliselt viigult analoogsignaali lugeda. Seda tehakse kujul *int viigu_aadress = aadress.* Kus *viigu_aadress* on muutujanimi ja *aadress* on

arendusplaadi viigu aadress, mille külge andur on ühendatud. Seejärel tuleb *setup()* funktsioonis seadistada viik sisendiks funktsiooni *pinMode()* abil, millele tuleb anda kaks argumenti kujul *pinmode(viigu_aadress, INPUT).* Selles funktsioonis on *viigu_aadress* muutuja, kuhu on eelnevalt salvestatud viigu aadress, millelt sisendit saadakse. *INPUT* näitab, et seadistame viigu sisendiks. Väljundi korral antakse selleks argumendiks *OUTPUT*. Nüüd on viik seadistatud ning saame sellelt viigult sisendit lugeda funktsiooni *analogRead()* abil, mis tuleb kirjutada *loop()* funktsiooni kujul *analogRead(<viigu_aadress>).*

Analoogsignaali kasutatavatest anduritest on heaks näiteks tuua fototakisti, mille kirjeldust ja Arduinoga ühendamist ning analoogsignaali lugemist selgitatakse järgmises punktis.

2.1.3. Analoogsignaalil töötav fototakisti

Käesolevas töös kasutame fototakistit ehk valguse intensiivsuse andurit [12]. Fototakisteid kasutatakse valguse intensiivsuse mõõtmiseks, aga ka lihtsalt valguse olemasolu või mitte olemasolu tuvastamiseks. Fototakisti takistus väheneb valguse toimel, seega mida pimedam on, seda suurem on takistus. Olenevalt fototakistist võib täielikus pimeduses mõõdetav takistus olla 500 000 – 1 000 000 oomi [12, 13]. Ereda päikesevalguse käes võib takistus olla vaid 50 oomi. Konkreetse fototakisti takistuse vahemikke on võimalik mõõta multimeetriga. Fototakisti koosneb kahest jalast, mis on kinnitatud keraamilise alus külge (joonisel 8). Keraamilisel alusel on valgustundliku aine kiht, näiteks kaadiumseleniidist või pliisulfiidist.

Joonis 8. Fototakisti [12].

Fototakisti mõõdab analoogsignaali vahemikus 0-5 volti. Elektripinge tekitamiseks on vaja fototakisti ühele jalale rakendada 5 voldist pinget ning teisele jalale 0 voldist pinget ehk

maandust. 5 voldise sisendpinge võtab fototakisti endale sisendiks, mis keraamilisel alusel olevates valgustundlikes materjalides toimuvate protsesside mõjul kahaneb ning jõuab teise jalani, mille pinget mõõtes saab anduri näidu. Füüsilist ühendust arendusplaadiga ja praktilist kasutus näidatakse järgmises punktis.

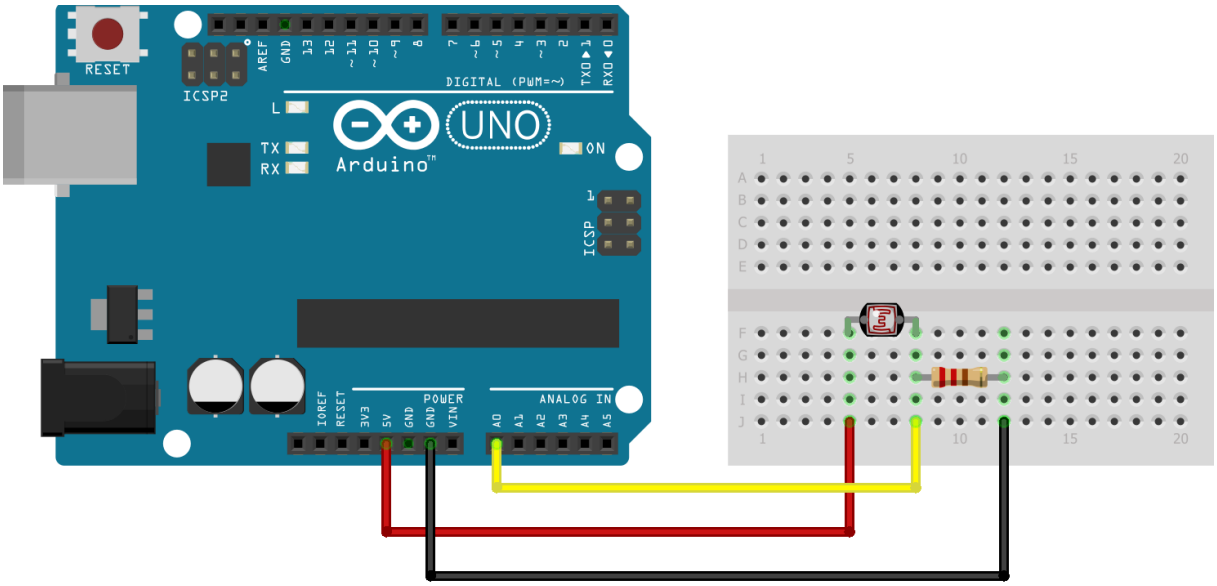
2.1.4. Programm fototakistilt loetud näidu kuvamiseks

Eesmärgiks on fototakistilt loetud väärtuse kuvamine ja väljatrükk jadapordi monitorisse.

Ülesande täitmiseks vajalikud komponendid on:

- Arduino Uno arendusplaat;
- fototakisti;
- takisti;
- mõned ühendusjuhtmed.

Fototakisti ühendamiseks arendusplaadiga on vaja teha kolm ühendust. Üks juhe on analoogsignaali edastamiseks. Selleks ühendada fototakisti üks jalg arendusplaadi ühe analoogsisendit võimaldava viiguga, näiteks märgistusega **A0**. Teine jalg ühendada arendusplaadi viiguga **5V**. Samuti vajab fototakisti maandust, et ta ennast ei rikuks ega tekiks ülepinget. Maanduse juhe ühendada fototakisti selle jala külge, mis ühendati analoogviiguga. Fototakisti jala ja juhtme vahele tuleb lisada takisti, et liigne voolutugevus andurit ei kahjustaks. Eelnevalt kirjeldatud ühendusskeem on näidatud joonisel 9.



Joonis 9. Fototakisti ühendamine Arduino Unoga.

Nüüd kui füüsilised ühendused on korras, saame ühendada arendusplaadi arvutiga USB-ühenduse kaudu. Järgmiseks kirjutame valmis programmi. Avada Arduino IDE ning *File* menüüst valikuga *New* avada tühi programm. Kõigepealt tuleb defineerida analoogviik, millelt loetakse andurilt tulev näit ning defineerida ka muutuja, millesse näit salvestada. *setup()* funktsioonis seadistada analoogviik sisendiks ning määrata jadapordi boodikiiruseks 9600. *loop()* funktsioonis lugeda analoogviigult väärtus ja salvestada see muutujasse. Seejärel trükkida muutuja väärtus jadapordi monitorisse. Joonisel 10 on töötav näidiskood.

```
int fototakisti = A0; //viik, mille külge fototakisti on ühendatud

void setup() {
  pinMode(fototakisti, INPUT); //seadistame fototakisti sisendiks
  Serial.begin(9600); //jadapardi monitor jaoks andmeedastuskiiruse
määramine
}

void loop() {
  int sensorValue = analogRead(fototakisti); //fototakistilt loetakse
näit muutujasse
  Serial.println(sensorValue); //näidu väljatrükk jadapordi monitorisse
  delay(200); //väike viivitus enne järgmise näidu lugemist
}
```

Joonis 10. Näidiskood fototakistilt näidu lugemiseks ja kuvamiseks jadapordi monitoris.

Kui programm on valmis, siis laadida see arendusplaati ja võtta lahti jadapordi monitor. Selle saab menüüst *Tools* valides *Serial Monitor*. Jadapordi monitorisse trükitakse andurilt loetud väärtused. Jälgida tuleks, et jadapordi monitor oleks samuti seadistatud andmevahetuskiiruse

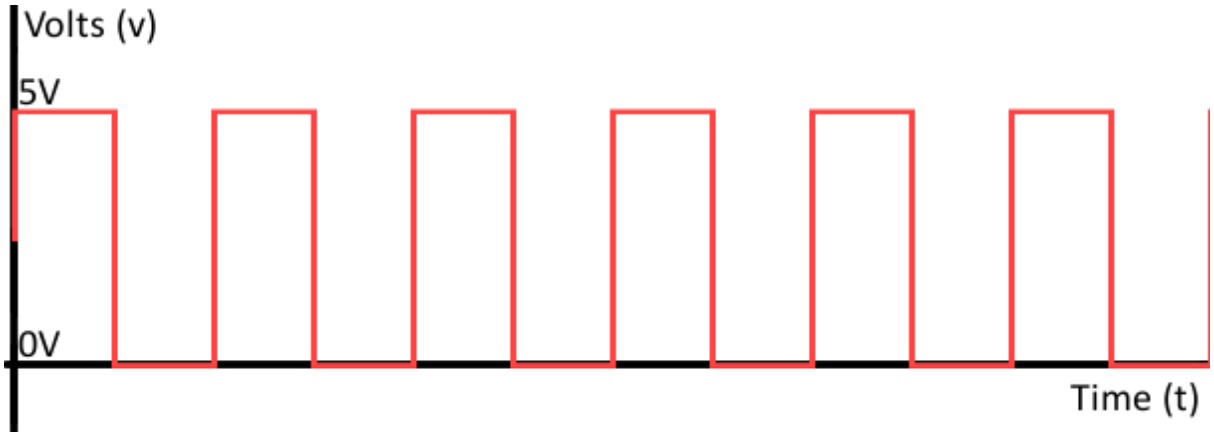
9600 bps peale. Kontrollida, kas avatud jadapordi monitori aknas all paremas nurgas on valitud *9600 baud.* Anduri töötamist kontrollida näiteks andurile puhumisega ning samal ajal jälgida, kas jadapordi monitorisse trükitavad väärtused suurenevad.

2.2. Digitaalandurid

Digitaalandur on andur, mis suhtlemiseks kasutab digitaalsignaali. Digitaalsignaale on mitmesuguseid - lihtsaid ja keerulisemaid. Lihtsad digitaalandurid kasutavad lihtsaid digitaalsignaale, millest kirjutatakse selles punktis. Esmalt selgitatakse, mis on digitaalsignaal ning seejärel, kuidas digitaalandureid kasutatakse ning tuuakse näide ühe digitaalanduri kasutamisest.

2.2.1. Digitaalsignaal

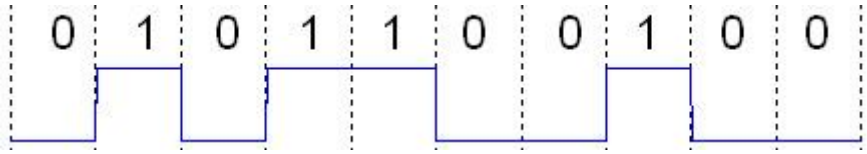
Digitaalsignaal on diskreetsignaal, mis tähendab, et sellele omistatakse väärtusi ainult kindlatel ajahetkedel [14]. Seega digitaalsignaal pole nii tihe ja pidev, kui oli seda analoogsignaal. Digitaalsignaalil on lõplik arv võimalikke väärtusi, mis jäävad kahe kindlalt fikseeritud piirväärtuse vahele [11]. Näitena võib siinkohal tuua andurid, mis mõõdavad elektripinget. Üldjuhul mõõdetakse siis väärtusi 0 V ja 5 V vahel (joonis 11), mis konverteeritakse ADC abil arvutile mõistetavateks väärtusteks.



Joonis 11. Digitaalsignaal, vahemikus 0 - 5 V [11].

Selline signaal, millel on ainult kaks võimalikku väärtust, nimetatakse loogiliseks signaaliks [14]. See on eelkõige kasutuses arvutiteaduses, kuna kirjeldab hästi kahendsüsteemi bitivooge.

Seda võib nimetada ka kahendsignaaliks (joonis 12). Lihtsamalt öeldes on see nullide ja ühtede jada. Tihti kasutatakse ingliskeelseid väljendeid HIGH ja LOW, kus esimene tähistab väärtust 1 ning teine väärtust 0.



Joonis 12. Loogiline kahendsignaal [14].

Digitaalsignaali graafikut iseloomustab astmelisus (joonis 11 ja 12), mis tuleneb sellest, et mõõtmisi teostatakse kindlatel ajahetkedel. Sellist kindlatel ajahetkedel mõõtmist nimetatakse inglise keeles *clocking*. Eesti keeles saaks seda nimetada taktsignaaliks, mis tähendab, et mõõtmine teostatakse kindla intervalli tagant. Erinevatel seadmetel ja tehnoloogiatel võib intervalli kestus erineda.

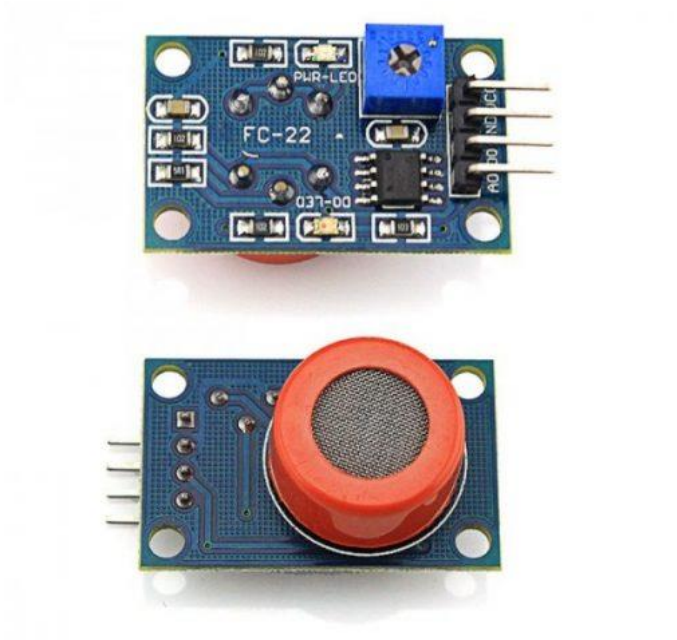
Järgmises punktis kirjeldatakse, kuidas lihtsaid digitaalandureid kasutada koos Arduinoga.

2.2.2. Digitaalandurite suhtlus arvuti ja Arduinoga

Lihtsad digitaalandurid edastavad informatsiooni 1 biti kaupa. Edastatud biti väärtusest sõltubki, kas andur tuvastas midagi või mitte.
Digitaalsignaali lugemiseks tuleb andur ühendada arendusplaadi sellise viigu külge, mis suudab lugeda digitaalsignaali. Sarnaselt analoogsignaali lugemisele, tuleb selle viigu aadress, kuhu andur ühendati, defineerida muutujasse kujul *int viigu_aadress = aadress*, kus *viigu_aadress* on muutujanimi ja *aadress* on selle arendusplaadi viigu aadress, kuhu külge andur ühendatud on. Seejärel tuleb *setup()* funktsioonis seadistada viik sisendiks funktsiooni *pinMode()* abil, millele tuleb anda kaks argumenti kujul *pinmode(viigu_aadress, INPUT)*, kus *viigu_aadress* on eelnevalt defineeritud muutuja ja *INPUT* näitab, et viik seadistatakse sisendiks. Nüüd on viik seadistatud lugema digitaalsignaali, mida loetaks funktsiooni *digitalRead()* abil, mis tuleb kirjutada *loop()* funktsiooni kujul *digitalRead(viigu_aadress)*. Funktsioon *digitalRead()* loeb andurilt näidu, mille suurus on 1 bitt ehk kontrollib, kas signaal on HIGH või LOW [9].

2.2.3. Digitaalsignaalil töötav alkoholiandur

MQ-3 alkoholiandur mõõdab alkoholikonsentratisooni väljahingatavast õhust. Kasutame alkoholiandurit moodulplaadil FC-22 (joonis 13). Moodulplaadil on neli viiku. A0, mis edastab analoogisignaalina alkoholikonsentratsiooni väljahingatavast õhust. D0, mis edastab lihtsa digitaalsignaali, selle kohta, kas mõõdetud alkoholikonsentratsioon on ületanud kindla piirväärtuse või mitte. GND ja VCC on toitepinge saamiseks, vajalik on 5 V. Moodulplaadil on veel vajalikud takistid alkoholianduri jaoks, toiteindikaator, D0 viigu väärtuse indikaator ning piirväärtust reguleeriv muuttakisti [16].



Joonis 13. Alkoholiandur MQ-3 moodulplaadiga FC-22 [15].

Järgnevas punktis on kirjeldatud alkoholianduri praktilist kasutamist.

2.2.4. Programm alkoholiandurilt näidu lugemiseks

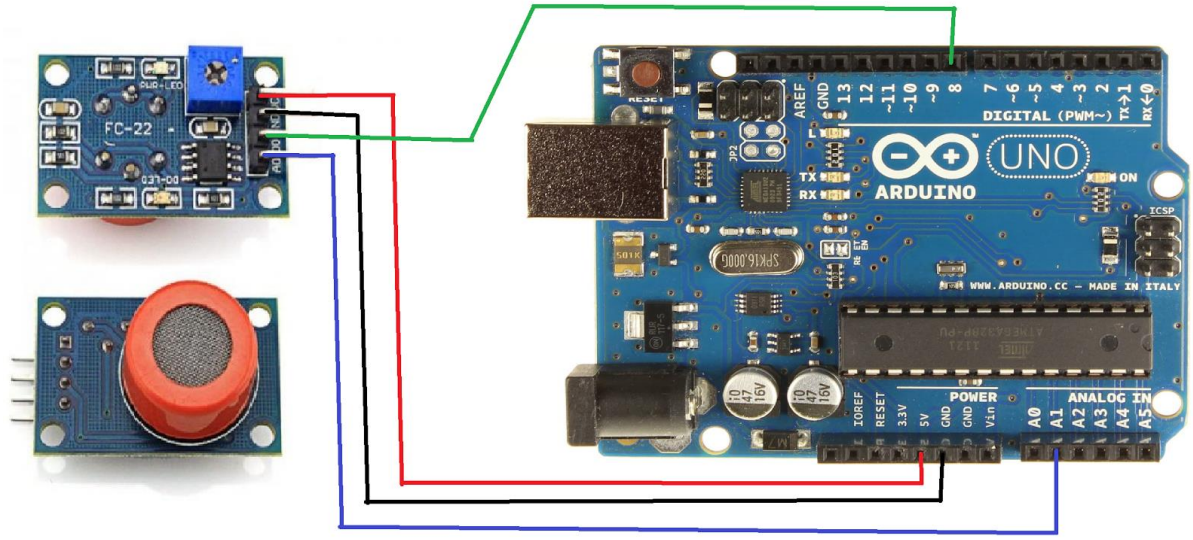
Eesmärk on luua programm, mis loeb MQ-3 alkoholiandurilt näidu ja trükib selle jadapordi monitorisse.

Ülesande täitmiseks vajalikud vahendid on:

- Arduino Uno arendusplaat;
- alkoholiandur MQ-3 moodulplaadiga FC-22;

- mõned ühendusjuhtmed.

Alkoholianduri moodulplaadi ühendamiseks arendusplaadiga on vajalik teha neli ühendust. Kokku tuleb viia arendusplaadi **5V** viik anduri **VCC** viiguga. Samuti ühendada **GND** viigud omavahel. Anduri **A0** viik ühendada arendusplaadi ühe analoogviiguga, näiteks **A1** ning **D0** viik ühe digitaalviiguga, näiteks aadressiga **8**. Eelnevalt kirjeldatud ühendusviis on näidatud joonisel 14.



Joonis 14. Alkoholiandur MQ-3 moodulplaadi FC-22 ühendamine Arduino Uno arendusplaadiga.

Järgmiseks kirjutame valmis programmi. Avada Arduino IDE ning *File* menüüst valikuga *New* avada tühi programm. Kõigepealt tuleb defineerida analoog- ja digitaalviik ning defineerida ka muutujad, millesse nendelt viikudelt loetud näidud salvestada. *setup()* funktsioonis seadistada analoogviik ja digitaalviik sisendiks ning määrata jadapordi boodikiiruseks 9600. *loop()* funktsioonis lugeda analoog- ja digitaalviigult väärtused ja salvestada need eelpool defineeritud muutujatesse. Seejärel trükkida muutujate väärtused jadapordi monitorisse. Joonisel 15 on töötav näidiskood.

```
//MQ-3 alkoholianduri näidu lugemine

int AOpin = A1; //viik, analoogsignaali lugemiseks
int DOpin = 8;  //viik, digitaalsignaali lugemiseks

int limit; //muutuja, kuhu salvestatakse digitaalsignaali väärtus (kas 1
või 0)
int value; //muutuja, kuhu salvestatakse analoogsignaali väärtus

void setup() {
  Serial.begin(9600); // jadapordi monitori jaoks boodikiiruse määramine
  pinMode(DOpin, INPUT); //seadistame viigud sisendiks
  pinMode(AOpin, INPUT);
  }

void loop(){
  value = analogRead(AOpin); //loeme andurilt näidu
  limit = digitalRead(DOpin); //loeme digitaalsignaali väärtuse
  Serial.print("Alkoholi vaartus: ");
  Serial.print(value); // prindime välja alkoholiväärtuse
  Serial.print("  Piirvaartus: ");
  Serial.println(limit); // prindime välja, kas piirväärtus on ületatud
või mitte
  delay(100);            // kui on 1 siis mitte, kui 0 siis on ületanud
 }
```

Joonis 15. Näidiskood MQ-3 alkoholiandurilt näidu lugemiseks.

Kui füüsilised ühendused on tehtud ja programm on valmis, tuleb ühendada arendusplaat arvutiga USB-ühenduse kaudu, laadida programm arendusplaadi mälusse ja võtta lahti jadapordi monitor. Selle saab menüüst *Tools* valides *Serial Monitor*. Jadapordi monitorisse trükitakse andurilt loetud väärtused. Jälgida tuleks, et jadapordi monitor oleks samuti seadistatud boodikiiruse 9600 peale, selleks tuleb kontrollida, kas avatud jadapordi monitori aknas all paremas nurgas on valitud *9600 baud.* Alguses tuleb lasta anduril umbes 20 sekundit soojeneda, et ta mõistlikke tulemusi annaks. Anduri töötamist saab kontrollida, kui asetada anduri lähedusse kange alkohol või lasta alkoholi tarbinud täiskasvanul puhuda andurile. Samal ajal tuleb jälgida, kas jadapordi monitorisse trükitavad väärtused suurenevad. Samuti jälgida piirväärtuse ületamist, kui see on ületatud, siis on digitaalsignaali edastavalt viigult loetud väärtus 0 ja anduril süttib indikaator, vastasel juhul on indikaator kustus ja väärtus on 1.

Keerulisemad digitaalandurid suhtlevad keerukamate digitaalsignaalidega. Nende lugemine pole enam nii lihtne, kuna keerukamaid signaale edastatakse kindlate protokollide ja liideste kaudu. Järgnevas peatükis tutvustatakse enimkasutatavaid suhtlusprotokolle ja neid toetavaid liideseid.

3. Liidesed ja protokollid

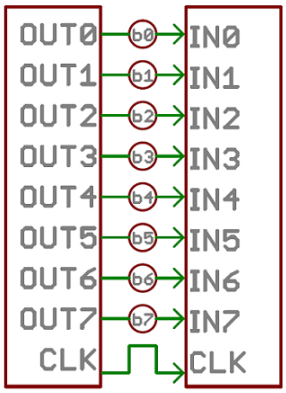
Eelmises punktis tutvustati analoog- ja digitaalandureid. Digitaalandureid kasutatakse palju rohkem, kuna need edastavad andmeid kohe arvutile arusaadaval kujul ehk digitaalsignaalina. Digitaalsignaale on mitmesuguseid ja need on pandud kirja protokollidena. Erinevate digitaalsignaalide edastamiseks on kindlad liidesed. Selles peatükis kirjutatakse neljast tähtsamast protokollist ja liidesest: paralleelühendus (inglise keeles *Parallel Communication*, jadaedastus (inglise keeles *Serial Communication*), *Serial Peripheral Interface* (SPI) ja *The Inter-Integrated Circuit Protocol* (I^2C).

3.1. Paralleelühendus

Käesolevas punktis selgitatakse, mis on paralleelühendus, kirjeldatakse paralleelühenduse liidest ning tuuakse näide andurist, mis paralleelsiini põhimõttel töötab.

3.1.1. Paralleelühenduse põhimõte

Paralleelühendus on lihtsa põhimõttega andmesuhtlusviis. Andmeedastus toimub läbi mitme juhtme. Joonisel 16 on toodud näide kaheksa juhtmega paralleelühendusest. Neid juhtmete arv oleneb kasutatavast riistvarast ning vajadusest. Tänu paljudele juhtmetele selles siinis on võimaldatud kiire andmete samaaegne edastus. Puuduseks on see, et nõuab palju pesasid arendusplaadilt ning seega on väiksemate pesade arvuga plaatidel paralleelühendust koos teiste anduritega keeruline kasutada [17].

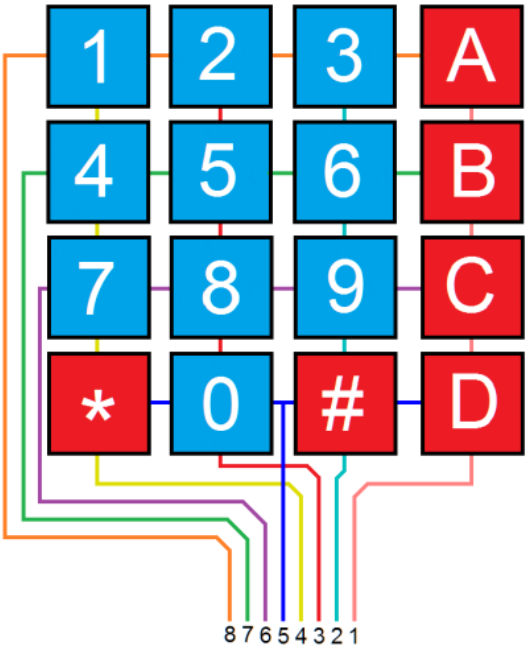


Joonis 16. Kaheksa juhtmega paralleelühendus [17].

Üheks paralleelühendust kasutatavaks anduriks on numbrimaatriks. See kasutab ka kaheksajuhtmelist ühendust. Numbrimaatriksist ja selle kasutamisest kirjutatakse järgmises punktis.

3.1.2. Paralleelühendusel töötav numbrimaatriks

4x4 numbrimaatriks on lihtne vahend registreerimaks nupuvajutust. Numbrimaatriksi ühendamiseks on vaja kaheksat juhet: neli ridade jaoks ja neli veergude jaoks. Ridade ja veergude ühenduste loogika ja juhtmete paigutus on esitatud joonisel 17. Iga nupp on ühenduses ühe rea ja ühe veeruga. Et registreerida nupuvajutust, muudab mikrokontroller, mille külge numbrimaatriks on ühendatud, kõigi nelja veeru olekuks LOW ning hakkab ridasid HIGH-ks muutma. Samal ajal kontrollides veergude väärtusi. Kui näiteks esimese rea väärtus tõsteti HIGH-ks ning neljanda veeru väärtus ka muutus HIGH-ks, siis see tähendab, et vajutati nuppu A [18] .



Joonis 17. 4x4 numbrimaatriksi ühenduste loogika [18].

Numbrimaatriksi kasutamiseks on vaja Keypad teeki. Teegi saab hankida otse Arduino IDE-st valides menüüst *Sketch -> Include Library -> Manage Libraries*. Avanenud aknast tuleks otsingulahtrisse kirjutada „keypad“ ning installeerida sellenimeline teek.

Teegi kasutamiseks tuleb, see programmi importida reaga *#include <Keypad.h>*. Maatriksi objekt luuakse meetodiga *Keypad(makeKeymap(sümbolite list), read[], veerud[], ridade_arv, veergude_arv).* Ridade ja veergude arv tuleks defineerida vastavalt maatriksi suurusele. Hetkel tuleb nii ridade kui veergude arvuks panna 4. Read[] ja veerud[] listidesse tuleb panna arendusplaadi pesade aadressid, kuhu maatriks on külge ühendatud. Joonise x järgi võiks read[] = {8,7,6,5} ja veerud[] = {4,3,2,1} sellised olla, kui arendusplaadile ühendada samade aadressidega digitaalpesade külge. Sümbolite listis tuleb kirjeldada vajutatavate nuppude

väärtused. Numbrimaatriksil on nuppude peale juba trükitud tähed ja numbrid, seega on mõistlik ka sümbolite listi panna samad tähed ja numbrid, samade nuppude alla. Nupuvajutuse ja vajutatava tähe või numbri saab kätte *getKey()* meetodiga, mida tuleks kasutada *loop()* funktsioonis [31]. Joonisel 18 on eelneva üks võimalik praktiline teostus.

```
#include <Key.h>
#include <Keypad.h>

const byte READ = 4; //ridade arv
const byte VEERUD = 4; //veergude arv
char symbolid[READ][VEERUD] = { //sümbolite list ridade kaupa
  {'1','2','3', 'A'},
  {'4','5','6', 'B'},
  {'7','8','9', 'C'},
  {'*','0','#','D'}
};
byte ridadePins[READ] = {8, 9, 10, 11}; //ühendada ridade juhtmed
digitaalpesadesse 8-11
byte veergudePins[VEERUD] = {7, 6, 5, 4}; //ühendada veergude juhtmed
digitaalpesadesse 7-4

Keypad keypad = Keypad(makeKeymap(symbolid), ridadePins, veergudePins,
READ, VEERUD); //keypad objekti loomine

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

void loop() {
  char symbol = keypad.getKey(); //nupuvajutuse registreerimine ja
sümboli salvestumine muutujasse
  if (symbol != NO_KEY){  //kui toimub nupuvajutus siis trükitakse
sümbol välja
    Serial.println(symbol);
  }
}
```

Joonis 18. Näitekood numbrimaatriksi seadistamiseks ja nupuvajutuse registreerimiseks. Kasutatud allikat [19].

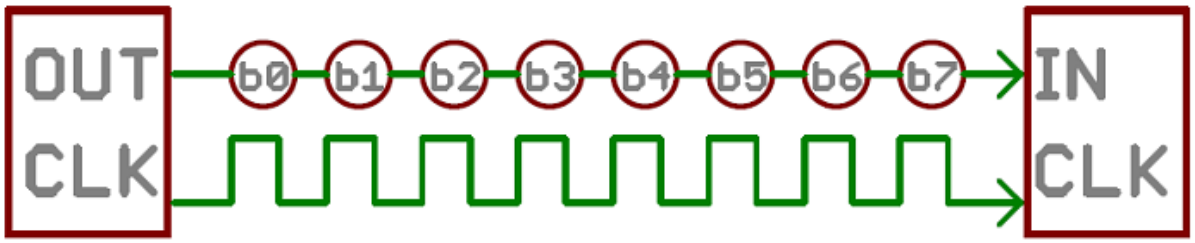
Maatriksiga on võimalik registreerida ka mitme nupu ühel ajal vajutamist, mille tuvastamiseks on eraldi meetodid. Kuna selles projektis ei kasutata mitme nupu üheaegselt vajutamise funktsiooni, siis neid meetodeid ei tutvustata.

3.2. Jadaedastus

Siin punktis selgitatakse, mis on jadaedastus, kirjeldatakse selle liidest ning riistvaraga ühendamist ja tuuakse näide jadaedastuselt töötavast andurist.

3.2.1. Jadaedastuse põhimõte

Jadaedastust kasutavad liidesed edastavad andmeid ühe biti kaupa ning seda üldjuhul ainult ühe juhtme kaudu [17]. Seda kõike kontrollib taktgeneraator, mis kindla taktsagedusega juhib seda bitivoogu. Iga taktimpulsi ajal edastatakse 1 bitt andmeid. Jadaedastuse üheks väga suureks eeliseks on see, et ta vajab väga vähe sisend-väljund pesasid. Joonisel 19 olevas näites on kasutuses ainult kaks juhet, mis on erinev paralleeledastusest, mis vajab tavaliselt vähemalt kaheksat või isegi rohkemat juhet.



Joonis 19. Näide jadaedastusest, mis on kontrollitud kindla taktsagedusega [17].

Jadaedastus minimaliseerib kasutatavate juhtmete arvu kahele. Kuidas jadaedastus täpsemalt toimub kirjeldatakse järgmises punktis.

3.2.2. Asünkroonne jadaedastus

Järgnev punkt on kirjutatud internetiallika [17] põhjal.
Jadaedastuse saab jagada kahte gruppi: sünkroonne või asünkroonne. Sünkroonne jadaedastus edastab oma andmeid alati koos taktsignaaliga ehk kõik seaded, mis kasutavad ühte siini jagavad sama taktsagedust. See teeb andmeedastuse väga üheselt mõistetavaks ja kiireks, kuid vajab seetõttu ühte lisajuhet seadmete vahel, et seda taktsagedust sünkroonis hoida.

Asünkroonne jadaedastus aga ei kasuta välise taktsignaali abi. Nii saame minimaliseerida kasutatavate juhtmete arvu ehk pole vaja lisajuhet taktsageduse sünkroniseerimiseks. Seega on andmete edastamine asünkroonne ning on vaja teha muid samme, et andmete saatmine ja vastuvõtmine kulgeks edukalt ning usaldusväärselt.

Et tagada vigadeta ja usaldusväärne andmeedastus on asünkroonse jadaedastuse protokollis kasutusel kindlad mehhanismid. Informatsiooni saadetakse kindlaksmääratud struktuuriga andmepakettide (joonis 20) kaudu.

Andmepakett koosneb järgnevatest osadest:

a) andmebitid
    - Iga blokk andmeid saadetakse pakettidena. Andmepaketis olevad vajaliku infoga varustatud andmebitte nimetatakse ka andmepakiks. Nende suurus võib varieeruda. See võib olla vahemikus 5 - 9 bitti. Kuigi standardne on ikkagi 8 bitine bait, aga kasutatakse ka 7 bitist lahendust, mis on sobiv ASCII sümbolite edastuseks.
b) sünkroniseerimisbitid
    - Sünkroniseerimisbitid on kaks või kolm erilist bitti, mis edastatakse koos iga andmebittide paketiga. Need on alustav bitt ja lõpetav bitt/bitid. Alustav bitt on andmepaketi algus ja lõpetav bitt lõpp. Lõpetavaid bitte võib olla ka kaks, kuid üldiselt kasutatakse ikka ühte. Alustav bitt muudab tegevuseta andmeliini väärtuse HIGH'lt LOW'ile, mis tähendab, et aktiivne andmeliin muutub HIGH'ks. Lõpetav bitt muudab tegevuseta andmeliini uuesti väärtusele HIGH.
c) paarsusbitid
    - Paarsusbitte kasutatakse väga madalatasemeliseks veakontrolliks. Nagu nimigi ütleb, siis loogika põhineb sellel, kas luuakse paarsusbitt või mitte. Kõik bitid andmepaketis summeeritakse kokku ning kui summa on paarisarv, siis otsustatakse, et andmepaketi edastus oli edukas ja paarsusbiti väärtuseks omistatakse HIGH, vastasel juhul LOW.
    - Paarsusbittide kasutamine pole tegelikult väga laialdases kasutuses ja on pigem valikuline. See aitab kindlasti kaasa vigade leidmisele, aga aeglustab natuke andmeedastuse kiirust, kuna kasutatakse ühte lisabitti ning seadmete tarkvarasse on vaja lisada koodiread, mis näitaksid, kuidas käitutakse vea tuvastamise korral.
d) boodikiirus/sümbolikiirus (inglise keeles *Baud Rate*)
    - Sümbolikiirus näitab kui kiiresti andmed saadetakse üle jadasiini. Seda väljendatakse tavaliselt ühikuga bitti sekundis (bps). Sümbolikiiruse võib teoorias konfigureerida ükskõik mis väärtusele, mida piirab ainult protsessori

kiirus, kui ainult mõlemad seadmed opereerivad ka samal sümbolikiirusel. Mida suurem on väärtus, seda kiiremini andmeid edastatakse. Tavaliselt ei kasutata suuremat väärtust kui 115200 bps, kuna see mikrokontrollerite jaoks juba piisavalt kiire ja suurem kiirus võib tuua kaasa vigasid. Standardsed sümbolikiirused on 1200, 2400, 4800, 19200, 38400, 57600 ja 115200.

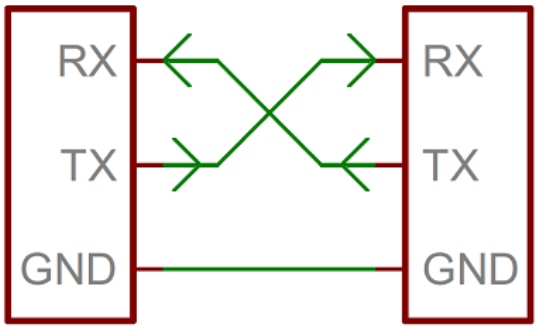
Joonis 20. Andmepaketi struktuur [17].

Kuna eelnevalt kirjeldatud mehhanisme on mitu ning neid saab konfigureerida päris laialt, siis pole ühte kindlat ja õiget seadistust. Küll aga tuleb olla kindel, et mõlemad seadmed, mis kasutavad seda jadaühendust, on seadistatud üheselt, muidu võib esineda anomaaliaid.

3.2.3. Riistvara ja selle ühendamine asünkroonse jadaedastuse jaoks

Järgnev punkt on kirjutatud internetiallika [17] põhjal.

Jadasiin koosneb kahest ühendusest - üks, mille kaudu saadetakse andmeid ühest seadmest teise ja teine, mille kaudu saadakse andmeid teisest seadmest esimesse. Näiteks, mikrokontroller saadab andurile info, mida ja kuidas mõõtma peab, andur saab selle kätte, teeb saadud info põhjal mõõtmised ning saadab saadud info mikrokontrollerile. Et sellist jadaedastust seadmete vahel teostada, peab seadmetel olema jadaedastust võimaldavad viigud. Nendeks on ressiiver ehk vastuvõttev viik märgistusega **RX** ja transmitter ehk saatev viik märgistusega **TX.** Sellised märgistused on tehtud seadmetele, tänu millele on juhtmetega ühendamine arusaadav. Tuleb aga meeles pidada, et ühendada tuleb ühe seadme **TX** teise seadme **RX**-ga ja vastupidi (joonis 21), kuna nagu nende nimedki ütlevad, siis ühe viigu kaudu saadetakse infot ja teise kaudu võetakse vastu, seega on loogiline eelpool kirjeldatud ühendus. Selline lahendus võib esmapilgul tunduda keeruline, kuna enamjaolt ühendatakse samade märgistustega pistikud.



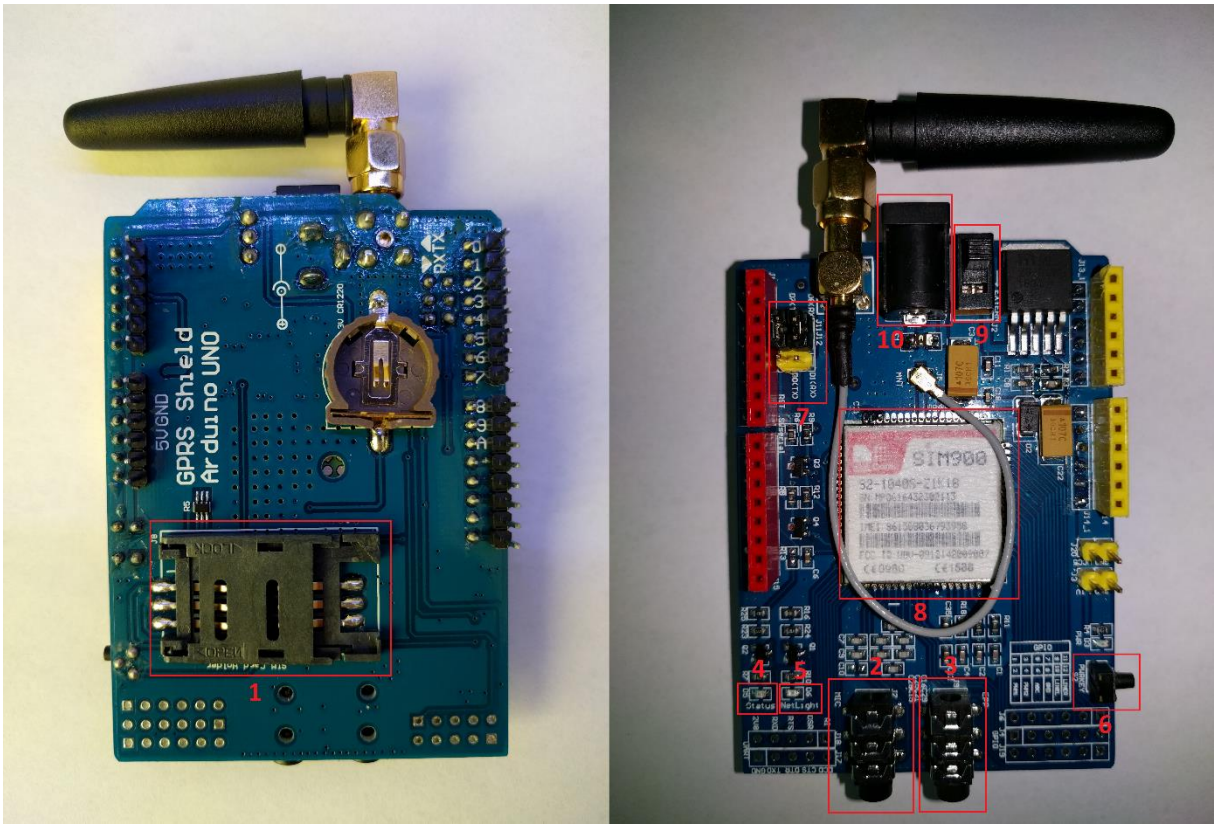
Joonis 21. RX ja TX ühendamise loogika [17].

Jadaliidesega seadmed võivad olla nii täisdupleks- kui ka poolduplekssidega. Täisdupleks tähendab, et mõlemad seadmed võivad samaaegselt nii saata kui vastu võtta andmeid. Poolduplex, aga et saatmine ja vastuvõtmine käib seadmete vahel kordamööda ehk üks seade saadab, teine võtab vastu, seejärel võib alles teine saata ning esimene vastu võtta.
On olemas ka ühe ühendusega jadasiine. Sel juhul on üks seade saatja ja teine vastuvõtja, aga mitte vastupidi. Ühendus luuakse saatva seadme **TX** ja vastuvõtva seadme **RX** vahel. Ühe ühendusega jadaedastust kasutatakse näiteks osade LCD-ekraanide puhul, kus siis mikrokontrollerist saadetakse info, mida ekraanil kuvatakse.

Jadaühendust kasutades tuleks teada, et jadaedastus on disainitud vaid kahe seadme omavaheliseks suhtluseks üle ühe siini. Kui kaks seadet üritaks saata andmeid samaaegselt üle ühe siini, siis tekivad tõrked. Näiteks tekib selline situatsioon, kui ühendada mikrokontrolleriga, mis on hetkel vastuvõtja rollis, kaks andurit, mis mõlemad on saatjad, üle ühe jadasiini. Mikrokontrolleri RX märgistusega pistik on ühendatud mõlema anduri TX pistikuga. Tekib olukord, kus mõlemad andurid võivad samaaegselt saata infot, mis tekitabki vigasid. Seega kaks seadet samaaegselt läbi ühe ühenduse andmeid saatma panna pole mõtet. Küll aga toimib asi vastupidiselt ehk kui ühendada üks saatev seade kahe vastu võtva seadmega ühe ühenduse kaudu. Aga ka selline lahendus pole väga soovitatav, kuna kui saatja saadab välja andmed, mis on mõeldud esimesele andurile, aga tänu ühisele ühendusele saab selle kätte ka teine andur, mis võib vigast sisendit saades käituda valesti.

3.2.4. Jadaedastusel töötav GPRS Shield

GPRS[6] Shield võimaldab kasutada GSM[7] mobiiltelefonivõrkudes GPRS pakettandmesideteenust. GSM ehk globaalne mobiilsidesüsteem on populaarseim mobiilsidetehnoloogia standard maailmas. 2014. aastaseisuga kasutati maailmas GSM standardit üle 90% tervest turust [20]. GPRS pakettandmesideteenus võimaldab andmeedastust allalaadimise kiirusega keskmiselt kuni 85 kbit/s ja üleslaadimist kuni 30 kbit/s. Nende kiirustega on võimalik teha tavalist audiokõnet, saata lühisõnumeid SMS[8] ja MMS[9] ning kasutada aeglast internetiühendust WAP[10] [21]. Joonisel 22 on kujutatud Arduino Unoga sobituvat GPRS Shieldi.



Joonis 22. GPRS Shield ja selle tähtsamad osad.

Joonisel 22 on näidatud punase kasti ja numbritega funktsionaalselt olulised osad. Number 1 on SIM-kaardi pesa, sinna tuleb sisestada SIM-kaart suuruses Mini. Number 2 on 3.5 mm

[6] GPRS (*General Packet Radio Service*) – üldine raadio-pakettandmeside teenus
[7] GSM (*Global System for Mobile Communication*) – globaalne mobiilsidesüsteem
[8] SMS (*Short Message Service*) – lühisõnumiteenus, mobiilisõnum
[9] MMS (*Multimedia Messaging Service*) - multimeedium-sõnumiteenus
[10] WAP (*Wireless Application Protocol*) – traadita rakenduste protokoll

liidesega mikrofonipesa. Number 3 on 3.5 mm liidesega audiopesa. Tänu mikrofoni- ja audiopesale pole vaja eraldi kõlarit ja mikrofoni arendusplaadiga ühendada. Number 4 on indikaator GPRS Shieldi tööoleku kohta. Kui indikaator põleb, siis on GPRS Shield sisse lülitatud, kui ei põle, siis on väljalülitatud. Number 5 on mobiilivõrgu indikaator. Kui 3 sekundiliste intervallidega, siis on GSM-moodul ühendatud võrku, kui indikaator vilgub 800ms intervallidega, siis võrguühendust pole, ning kui 300ms intervallidega, siis toimub GPRS suhtlus. Number 6 on toite sisse lülitamise lüliti. Number 7 on jadaedastust kasutavate pesade valija. Kui see on asendis D0/D1, siis kasutatakse jadaedastuseks pesasid D0 ja D1, kui aga asendis D7/D8, siis kasutatakse pesasid D7 ja D8. Number 8 on SIM900 kiip, mis juhib kogu plaadi tööd. Number 9 on lüliti välise toiteallika jaoks. Kui see on asendis EXTERN, siis kasutatakse toiteallikat, mis ühendatakse pesasse, mis joonisel on number 10, vastasel juhul saab plaat toite pesade 5V ja GND kaudu [22].

GPRS plaadiga töötamiseks on vaja lukustamata SIM-kaarti ehk ilma PIN-koodita. PIN-koodi saab SIM-kaardilt eemaldada mõnes mobiiltelefonis seadetes privaatsuse või SIM-kaardi sätete alt. Lukustamata SIM-kaart panna GPRS plaadi SIM-kaardi pesasse. GPRS plaati on väga kerge ühendada Arduino Uno arendusplaadiga. Arendusplaadi pesa A5 ja 0 tuleb viia kohakuti GPRS plaadi esimeste viikudega ning seejärel rahulikult kõik viigud lõpuni arendusplaadi pesadesse suruda. Osadel GPRS plaatidel pole viike kohe külge joodetud. Sellisel juhul tuleb seda ise teha.

Arduino riistvaral on sisseehitatud jadaedastuse võimekus digitaalviikudel 0 ja 1. Et saada kasutada ka teisi viikusid jadaedastusteks on Arduino IDE-ga kaasasolev teek *SoftwareSerial* [23]. GPRS Shieldiga saab kasutada viikusid D7 ja D8 jadaedastuseks. Seega on vajalik *SoftwareSerial* teegi kasutamine.

Jadaedastuse teegis on defineeritud funktsioonid, kuidas jadaedastust täpsemalt kasutada. Kõigepealt tuleb konstruktoriga *SoftwareSerial(rxPin, txPin)* luua isend, millega hakatakse jadamisi andmeid edastama. Konstruktoris on *rxPin* vastuvõttev viik ehk RX ja *txPin* edastav viik ehk TX. GPRS Shieldil kasutame viikusid 7 ja 8 ehk konstruktoriga isendi loomine näeb välja selline: *SoftwareSerial GPRSShield(7, 8)* [24]. Funktsiooniga *begin(kiirus)* pannakse paika andmeedastuse kiirus, kus *kiirus* on andmeedastuse kiirus ühikutes bps. GPRS Shieldi jaoks sobib järgnev kasutamine - *GPRSShield.begin(9600)* [25]. Funktsiooniga *available()* leitakse, kui palju andmeid on juba RX viigu puhvrisse saabunud. Funktsioon tagastab puhvri

baitide arvu [26]. Funktsioon *read()* loeb ühe baidi RX viigult [27]. Funktsiooniga *write(data)* trükitakse baidijada *data* jadapordile [28].

Joonisel 23 on GPRS Shieldilt andmete kättesaamine kasutades jadaedastuse teeke. Esmalt luuakse GPRS Shieldine isend viikudega 7 ja 8 ning defineeritakse vajalikud muutujad. Seejärel *setup()* funktsioonis määratakse andmeedastuse kiirused. Funktsioonis *loop()* kontrollitakse, kas andmeid on saabunud ning kui on siis loetakse need puhvrisse ning trükitakse jadapordi monitorisse ja puhver tühjendatakse.

```
#include <SoftwareSerial.h> //impordime jadaedastuse teegi

SoftwareSerial GPRSShield(7, 8); //loome jadaedastuse kaudu andmeid
saatva
unsigned char buffer[64]; //puhver info jaoks
int count = 0; //loendur puhvri jaoks

void setup(){
    GPRSShield.begin(9600); //käivitame GSM-mooduli jadaedastuse
kiiresel 9600bps
    Serial.begin(9600); //käivitame jadapordi monitori samal kiirusel
}

void loop(){
    if (GPRSShield.available() > 0){ //kui saadaval on uusi andmeid
        while(GPRSShield.available() > 0){ //siis loetakse need
puhvrisse
            buffer[count++] = GPRSShield.read();
            if(count == 64){ //kui puhver on täis siis jätkatakse
              break;
            }
        }
        Serial.write(buffer, count); //puhver kirjutatkse jadapordi
monitorisse
        for (int i=0; i<count;i++){  //puhver tühjendatakse järgmise
lugemise jaoks
          buffer[i]=NULL;
        }
        count = 0;  //loendur pannakse ka taas nulli
    }
}
```

Joonis 23. Jadaedastuse lugemine GPRS Shieldilt. Kasutatud allikat [22].

Järgmises punktis kirjeldatakse asünkroonse jadaedastuse edasiarendust, muutes see sünkroonseks.

3.3. Serial Peripheral Interface

Eelmises punktis tutvustati jadaedastust, mis on asünkroonne ehk andmeedastuse ajal ei ole kontrolli selle üle, kas mõlemad pooled (näiteks mikrokontroller ja andur) töötavad ühes rütmis. Seda seetõttu, kuna puudub ühist taktsignaali edastav juhe. *Serial Peripheral Interface* ehk SPI liidesel on eraldi taktsignaali kandev ühendus ning edastab andmeid sünkroonselt. Käesolev punkt on kirjutatud internetiallika [29] põhjal.

3.3.1. Sünkroonne liides

SPI liides on sünkroonne liides, mis tähendab, et sel on eraldi ühendused edastamaks andmeid ja taktsignaali. Taktsignaal on sünkroonne signaal ning ütleb täpselt, millal on vaja lugeda andmebitte, mida saadetakse andmeid edastavate ühenduste kaudu. Moment, millal täpselt andmeid lugeda, antakse taktsignaalis edasi väärtuse muutumisega LOW-st HIGH-ni või vastupidi. Kui väärtuse muutumine on toimunud, teab andmeid vastu võttev mikrokontroller, et on aeg lugeda andmeliinilt bitte.

Eelnevalt kirjeldatud ühendus on andmete saatmise mõttes kahjuks ainult ühepoolne tegevus. Kahepoolne andmeedastus on keerulisem. Taktsignaali dikteerib siiski vaid üks pooltest. See pool, mis genereerib taktsignaali, nimetatakse ülemaks (inglise keeles *master*). Teist poolt nimetatakse sel juhul alluvaks (inglise keeles *slave*). Alati saab olla ainult üks ülem, kuid ühel ülemal võib olla mitmeid alluvaid. Taktsignaali edastavat ühendust märgitakse tavaliselt lühendiga SCK (*Serial Clock*).

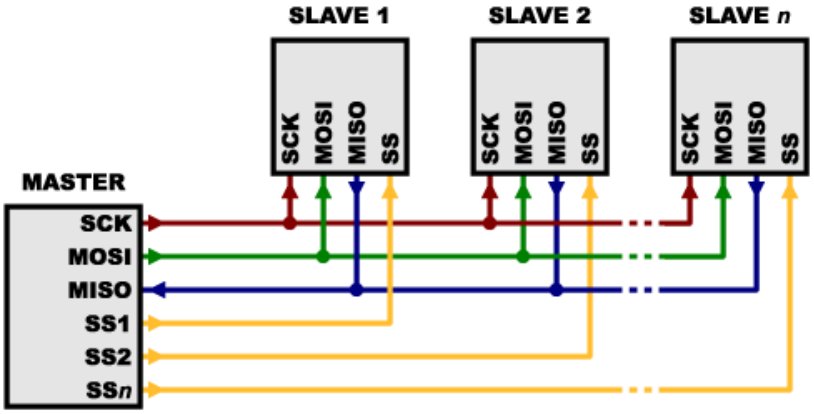
Kahepoolse andmeedastuse jaoks kasutatakse kahte ühendust. Kui andmeid saadetakse ülemast alluvasse, siis seda ühendust nimetatakse MOSI (*Master Out/Slave In*). Kui saadetakse alluvast ülemasse, siis nimetatakse MISO (*Master In/Slave Out*). Ülem peab teadma, kui suurt andmehulka alluvalt oodata on, et tagada terve informatsiooni vastuvõtmine. Kuna vaid ülem dikteerib taktsignaali, siis pärast seda, kui ta ise on andmed edastanud, genereerib ta taktsignaali tsükleid andmete vastuvõtmise jaoks, milleks tsüklite arvuks ongi vaja teada, kui mitu bitti andmeid alluv saadab. Osade anduritega suhtlemiseks on alluvatelt oodatava vastuse bittide hulk kindel, näiteks kaks bitti või kolm bitti. Bittide hulk võib olla ka varieeruv. Siis saadetakse esimese või esimese kahe bitiga info, kui suur hulk andmebitte tuleb vastu võtta.

SPI on täisduplekssidemega, kuna kasutab erinevaid ühendusi andmete saatmiseks ja vastuvõtmiseks.

3.3.2. Ühe või mitme alluvaga suhtlemine

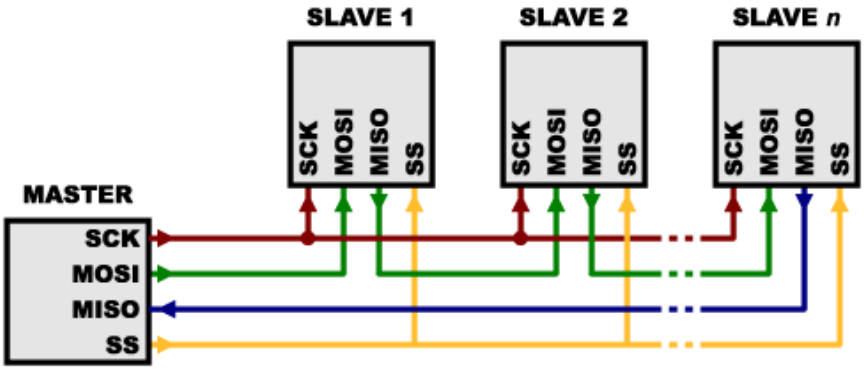
SPI liideses kasutatakse ka neljandat ühendust märgistusega SS (*Slave Select*). SS kaudu antakse alluvale märku, et ta oleks valmis andmeid vastu võtma või saatma. Samuti kasutatakse SS ühendust, kui kasutatakse mitut alluvat, et valida millise alluvaga ülem suhelda soovib. SS liini väärtust hoitakse tegevuseta olekus HIGH. Kui väärtus viiakse LOW-i, siis alluv aktiveerub ja andmeedastus võib hakata pihta.

Ülemat saab ühendada mitme alluvaga kahte moodi. Esimesel juhul on vaja iga alluv ühendada eraldi SS liiniga. Kõik alluvad jagavad aga ühist SCK, MOSI ja MISO liini (joonis 24). Et ühe alluvaga suhelda, tuleb konkreetse alluvaga ühenduses oleva SS liini väärtus viia LOW-i ning teiste alluvate omasid hoida HIGH. Nii tuleb teha selleks, et vaid üks alluv saaks saata andmeid MISO liini kaudu, kuna kõik alluvad on ühendatud ülema ühe MISO ühenduse külge. Vastasel juhul võib tekkida tõrkeid või saadakse moonutatud infot.



Joonis 24. SPI ühendus mitme alluvaga eraldi SS liinide abil [29].

Teisel juhul on kõikidel alluvatel ühine SCK ja SS liin. MOSI ja MISO liinid on ühendatud jadamisi. Ülema MOSI on ühendatud ühe alluva MOSI-ga ning selle alluva MISO on ühendatud järgmise alluva MOSI-ga ja nii edasi kuni viimase alluvani, mille MISO ühendatakse ülema MISO-ga. Joonisel 25 on esitatud ühendusskeem. Sellise ühendusskeemi korral aktiveeritakse kõik alluvad korraga, mistõttu ta ei sobi kõikide projektide jaoks. Pigem kasutatakse eelpool kirjeldatud siini ainult väljundipõhiseks suhtluseks, kus alluvad andmeid tagasi saatma ei pea.

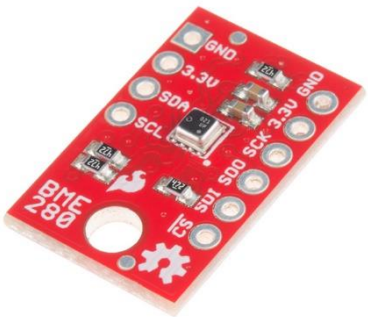


Joonis 25. SPI ühendus mitme alluvaga ühise SS liini abil [29].

Tähelepanu võiks pöörata teadmisele, et ülem peab saatma välja piisavalt palju käsklusi, et info jõuaks tahetud alluvani, sest esimesena saadetud käsklus liigub kohe viimase alluvani.

3.3.3. SPI liidesega atmosfääriandur

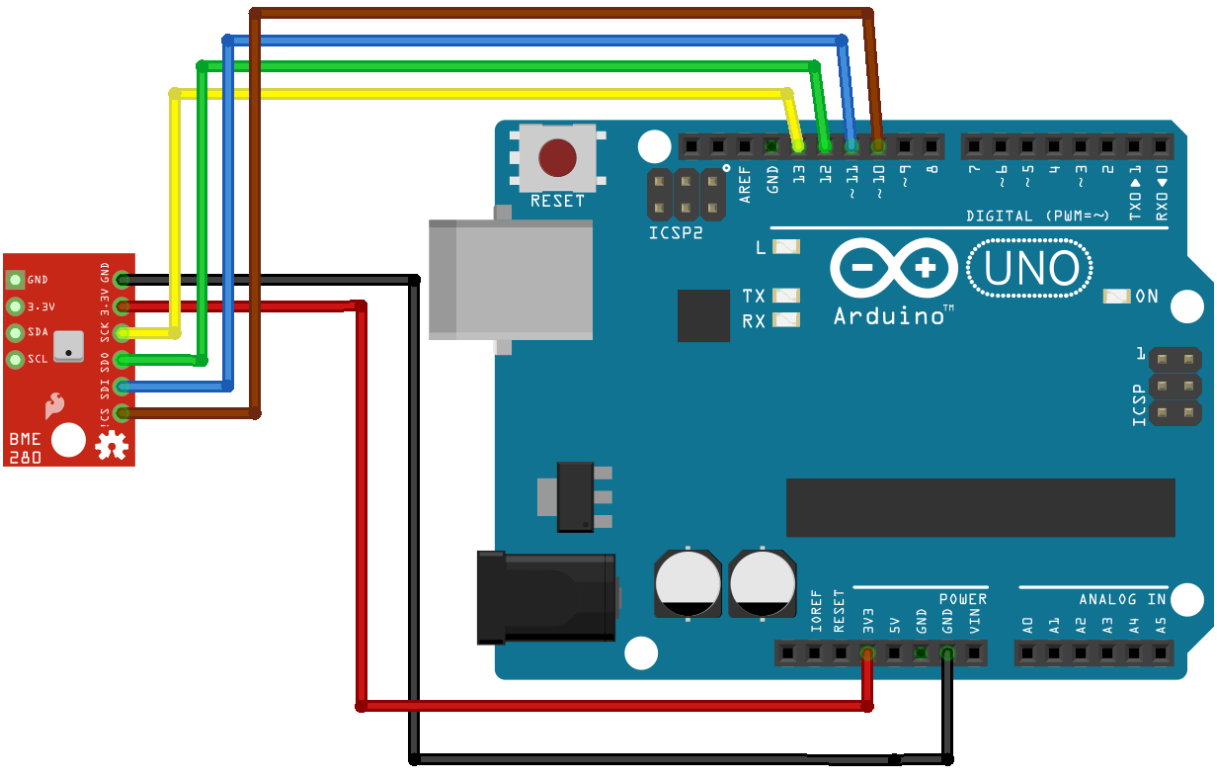
Atmosfääriandur BME280 on multifunktsionaalne andur (joonis 26), millega saab mõõta atmosfääri kolme tähtsaimat nähtust: õhurõhku, temperatuuri ning õhuniiskust. BME280 mõõdab õhurõhku vahemikus 30kPa kuni 110kPa ning selle põhjal arvutab suhtelise õhuniiskuse ning õhutemperatuuri. Andur toetab nii SPI kui ka I^2C suhtlust. SPI liides töötab 3.3 voldise pingega ning I^2C võimaldab nii 3.3 kui 5 voldise sisendpingega töötada. Korraga saab kasutada ühte liidest [30].



Joonis 26. Atmosfääriandur BME280 [30].

Hetkel on tähtis SPI liides ning selle viigud. SPI liides kasutab sellel anduril kuute viiku. Tähis GND on maandus. 3.3V nõuab 3.3 voldist sisendpinget, millel andur töötab. SCK on taktsignaali jaoks. SDO on andmete saatmise ja SDI andmete vastuvõtmise jaoks. CS on kiibi valimise jaoks. SPI ja I^2C ühenduse jaoks on anduril erinevad kiibid [31]. Arduino Uno

võimaldab SPI ühendust digitaalpesadega 10-13. Ühendusskeem Uno ja anduriga on esitatud joonisel 27.



Joonis 27. BME280 atmosfääri anduri ühendusskeem Arduino Unoga.

Atmosfäärianduri suhtlemiseks on vajalikud SPI teegi, mis on Arduino IDE-ga juba kaasas ning Sparkfun BME280 teeki. BME280 teegi saab alla laadida Sparkfuni Githubi kontolt [32] ning see Arduino IDE-sse lisada.

Tahame nüüd andurilt lugeda temperatuuri, õhuniiskuse ja õhurõhu näitu. Selleks tuleb avada Arduino IDE-s tühi programm. Kõigepealt on vaja importida alla laaditud teek *#include "SparkFunBME280.h"* ning SPI teegid *#include "Wire.h"* ja *#include "SPI.h"*. Seejärel BME280 konstruktoriga luua isend. Funktsioonis *setup()* tuleb konstruktoriga loodud isend seadistada SPI siinil töötama. Määrates *settings.commInterface = SPI_MODE* kasutab andur SPI liidest ning määrates *settings.chipSelectPin = 10* valib andur kasutamiseks SPI liidesele mõeldud kiipi. Funktsiooniga *begin()* rakendatakse eelpool seadistatud seaded.

Andurilt näite saab lugeda erinevate *read*-funktsioonidega. Temperatuuri lugemiseks kraadides Celsiuse järgi on funktsioon *readTempC()*. Õhurõhu jaoks on *readFloatPressure()*, mis mõõdab näidu kilopaskalites. Õhuniiskuse jaoks on *readFloatHumidity()*, mis loeb õhuniiskuse

protsendi. Joonisel 28 on näidiskood, mis näitab, kuidas on võimalik seda kõike reaalselt kasutada. Malli on võetud Sparkfun BME280 teegiga kaasas olevast näiteprogrammist.

```
#include "SparkFunBME280.h" //BME280 anduri teegi import

#include "Wire.h" //SPI teekide importimine
#include "SPI.h"

BME280 mySensor; //kontruktoriga anduri isendi loomine

void setup(){
  mySensor.settings.commInterface = SPI_MODE; //commInterface võib olla
SPI_MODE või I2C_MODE
  mySensor.settings.chipSelectPin = 10; //SPI kiibi valimine
  mySensor.settings.runMode = 3; //  normaalne töörežiimi valimine
  mySensor.settings.tStandby = 0; //  0.5ms-ne ooteaeg
  mySensor.settings.filter = 0; // filter välja
  Serial.begin(57600);
  delay(10);
  mySensor.begin(); //eelpool seadistatud seadete rakendamine
}

void loop(){
  Serial.print("Temperatuur: ");
  Serial.print(mySensor.readTempC(), 2); //temperatuuri näidu lugemine
  Serial.println(" C");
  Serial.print("Õhurõhk: ");
  Serial.print(mySensor.readFloatPressure(), 2); //õhurõhu näidu
lugemine
  Serial.println(" Pa");
  Serial.print("Õhuniiskus %: ");
  Serial.print(mySensor.readFloatHumidity(), 2); //õhuniiskuse näidu
lugemine
  Serial.println(" %");
  delay(1000);
}
```

Joonis 28. Atmosfäärianduri BME280 seadistamine ja näitude lugemine.

Kuid ka SPI liidesel on arenguruumi, kuna liidese kasutamiseks on vaja üpriski palju ühendusjuhtmeid ja vabu pesasid, mis näiteks Arduino Uno arendusplaadil on küllaltki piiratud. Selleks on välja mõeldud veelgi keerukam digitaalsignaali edastamise protokoll, mis ei vaja füüsiliseks ühenduseks rohkem kui kahte pesa.

3.4. The Inter-Integrated Circuit Protocol

Selles punktis selgitatakse, mis on I^2C ehk Inter-Integrated Circuit protokoll, mis põhimõtetel see töötab ning kuidas kasutatakse. Lõpus tuuakse näide OLED[11] ekraanist, mis kasutab I^2C liidest ning demonstreeritakse selle kasutamist. Antud punkt on kirjutatud internetiallika [33] põhjal.

3.4.1. I^2C põhimõte

I^2C vajab töötamiseks kahte juhet. Suurim erinevus jadaedastusega on see, et I^2C kahe juhtmega on võimalik ühel ülemseadmel olla kuni 1008 alluvuses olevat seadet. Samuti saab ühel alluval olla ka mitu ülemseadet.

I^2C koosneb kahest signaalist: SCL ja SDA. SCL on taktsignaal ning SDA on andmesignaal. Nagu eelnevalt kirjutatud, siis üks juhe on taktsignaal edastuseks ning teine juhe on andmesignaali jaoks. Taktsignaal tuleb alati siini ülemseadmelt.

I^2C-s saavad seadmed signaaliliini väärtust alandada, aga mitte iial suurendada. Seega välistatakse olukord, kus üks seade üritab väärtust alandada ja teine suurendada, mis tekitaks probleeme. Kui aga liinil on väärtus toodud alla mingi seadme poolt, aga seadmed ise pole võimelised seda jälle üles tooma, siis peab seda tegevust muude abivahenditega tegema. Selle jaoks on igal liinil spetsiaalsed väärtust üles tõstvad takistid. Need tõstavad signaali jälle kõrgeks parasjagu siis kui üksi seade seda ei ürita alla tuua.

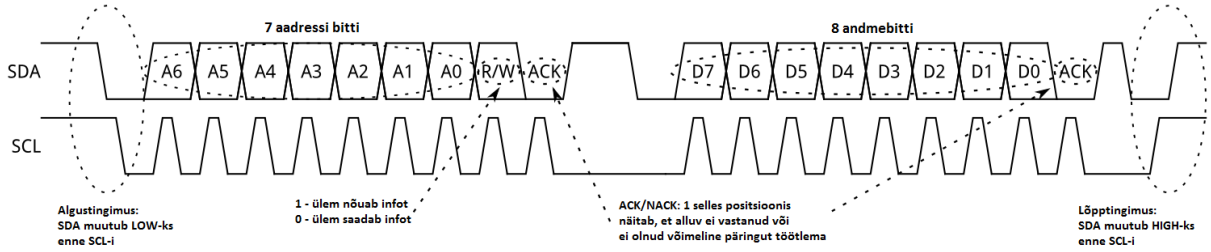
3.4.2. Protokoll

I^2C on palju keerukam kui eelnevates punktides kirjeldatud suhtlusviisid. Signaali edastamine peab järgima kindlat protokolli, et saaks tunnistada I^2C side korrektseks.

Protokolli järgi andmesignaali kaudu saadetavad sõnumid on jaotatud kahte tüüpi raamistikeks: aadressiraamistik ja andmeraamistik. Aadressiraamistikus on info, kus ülemseade näitab alluvale, mille jaoks see sõnum on saadetud. Andmeraamistikus sisalduvad reaalsed andmed, mida ülem saadab alluvale või vastupidi.

Sõnumi struktuur on esitatud joonisel 29, kust on näha, et sõnum algab aadressiraamistikuga, millele järgneb üks või rohkem andmeraamistikku.

[11] OLED (*Organic Light Emitting Diode*) – orgaaniline valgusdiood



Joonis 29. I²C-s saadetava sõnumi struktuur [33].

Et alustada aadressiraamistiku saatmisega toob ülemseade andmesignaali väärtusele LOW samal ajal kui taktsignaal on veel HIGH. See annab märku alluvatele seadmetele, et sõnumi saatmine algab. Seejärel alustatakse aadressiraamistiku saatmisega andmesignaali kaudu ja ülemseade hakkab tootma taktimpulsse taktsignaali edastaval liinil. Aadress on 7-bitti pikk (joonisel 29 A6 - A0). Sellele järgneb kaheksas bitt ehk R/W bitt (*Read/Write Bit*), mis näitab, kas tegu on kirjutamis- või lugemisoperatsiooniga ehk kas seade nõuab informatsiooni või saadab seda. Üheksas ja viimane bitt aadressiraamistikust on NACK/ACK[12] bitt. See on raamistikku lõpetav bitt nii aadressraamistiku kui ka andmeraamistiku jaoks. NACK/ACK bitt näitab kas vastu võttev seade sai informatsiooni kätte (ACK) või mitte (NACK).

Nüüd kui aadressiraamistik on saadetud, saame alustada andmeraamistike ülekandega. Ülemseade toodab endiselt regulaarset taktimpulssi ning andmeraamistikud saadetakse üle andmesignaali. Andmeraamistikke võib olla rohkem kui üks ning iga raamistik lõppeb NACK/ACK bitiga.

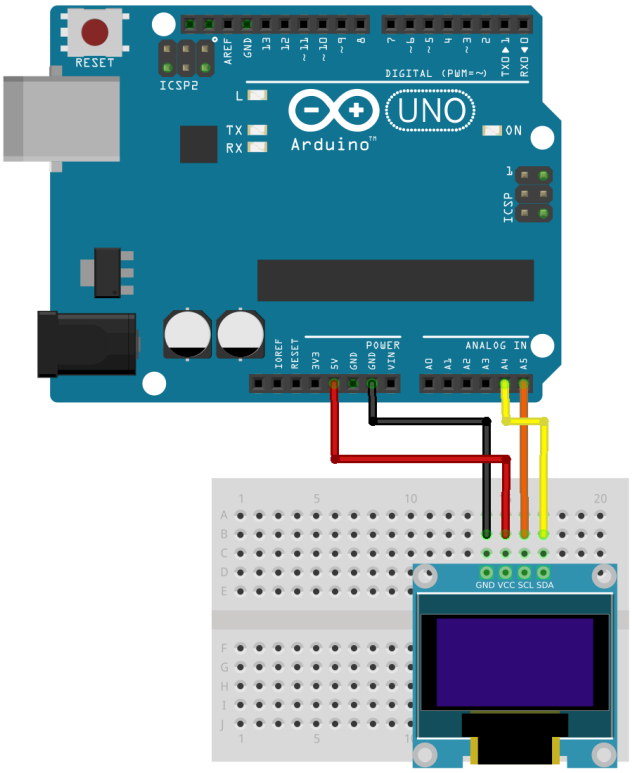
Kui kõik andmeraamistikud on saadetud, siis ülemseade genereerib lõppolukorra. Lõppolukord on defineeritud nii, et taktsignaali väärtuseks kirjutatakse HIGH ning pärast seda muutub ka andmesignaali väärtus suuruseks HIGH ja jääb sinna püsima kuni järgmiste korraldusteni.

3.4.3. I²C liidesega OLED ekraan

0.96 tolline 128x64 pikslit OLED ekraan on I²C liidesega. Ekraanil on neli viiku. GND ja VCC (osadel ekraanidel ka VDD) ekraanile toite saamiseks – GND ühendada arendusplaadil samuti pesaga GND ning VCC ühendada pesaga 5V. Ekraan töötab ka 3.3 voldise pingega, kuid soovitatav on kasutada 5 volti. Andmeedastuseks kasutatakse I²C liidest. Selle jaoks on viigud SDA ja SCL (osadel ekraanidel ka SCK). Need tuleks ühendada arendusplaadi pesadega, mis

[12] NACK/ACK (NotAcknowledge/Acknowledge) – mittekinnitus/kinnitus

toetavad I^2C suhtlust. Näiteks, saab ühendada SDA arendusplaadi pesaga A4 ja SCL pesaga A5. Ühendusskeem on näidatud joonisel 30. Ekraan kasutab SSD1306 draiverit [34].



Joonis 30. Ekraani ühendusskeem Arduino Unoga [34].

Ekraani töötamiseks võib kasutada u8glib teeki [35]. Ekraani töökorras olemist ja õiget seadistust saab kontrollida u8glib teegiga kaasatulevast näiteprogrammist *HelloWorld.ino*, mille leiab imporditud u8glib teegi kaustast *examples* alamkausta alt. Avanenud programmist näeme, et ekraaniga töötamiseks on vaja programmi importida u8glib teek reaga *#include "U8glib.h"*. Järgnevalt on koodis suur hulk erinevaid konstruktoreid, mis on mõeldud erinevate ekraanidega töötamiseks. Siin töös kasutatakse SSD1306 draiveriga 128x64 pikslist I2C liidesega ekraani. Seega tuleks leida nende konstruktorite seast rida *//U8GLIB_SSD1306_128X64 u8g(U8G_I2C_OPT_NONE|U8G_I2C_OPT_DEV_0); // I2C / TWI* ning see välja kommenteerida. Võib juhtuda, et kuigi kasutatakse 128x64 pikslilist ekraani aga arvuti tunnistab seda, kui 128x32 pikslist ekraani, siis tuleks välja kommenteerida rida *//U8GLIB_SSD1306_128X32 u8g(U8G_I2C_OPT_NONE); // I2C / TWI*. Ekraanile joonistamine käib järgneva loogika järgi, mis on näidatud joonisel 31.

```
void loop(void) {
  // ekraanile joonistamise tsükkel
  u8g.firstPage();
  do {
    u8g.setFont(u8g_font_unifont); //tekstifondi seadistamine
    u8g.drawStr(0, 22, "Hello World!"); //teksti "Hello World!"
kirjutamine ekraanile koordinaatidega x=0 ja y=22
  } while( u8g.nextPage() );
  delay(50);
}
```

Joonis 31. Ekraanile joonistamise tsükkel *loop()* funktsioonis. Kasutatud näiteprogrammi *HelloWorld.ino* eeskuju [35].

Kõigepealt tuleb ekraanile rakendada *firstPage()* meetodit, mis initsialiseerib ekraani, mille järgselt saab ekraanile teksti joonistada. *do*-blokis tuleb kirjeldada kõik, mida ekraanil tahetakse kuvada. *nextPage()* rakendab *do*-blokis kirjeldatud käsud. Hetkel on oluline vaid teksti kuvamine ekraanil. Selleks seadistamine tekstile fondi meetodiga *setFont(font)*. Erinevaid fonte leiab u8glib teegi alamkaustast *utility* failist *u8_font_data.c*. Meetodiga *drawStr(x, y, sõne)* kirjutatakse valitud koordinaatidega kohta sõne.

Teades nüüdseks, missuguseid signaale, protokolle ja liideseid enamik andurid ja moodulid kasutavad, tuleks õppida neid ka koos kasutama. Järgnevas peatükis ehitatakse Arduinole nuputelefon GPRS Shieldi abil kasutades juba eelnevalt tutvustatud andureid ja ekraani.

4. Nuputelefoni ehitamine GSM-mooduli abil

Antud peatükis ehitatakse Arduino Uno arendusplaadile nuppudega telefon, suures osas samade vahenditega, mida tutvustati eelmistes peatükkides. Idee on näidata, milliseid komponente nuputelefon vajab ning kuidas selle ehitamine realiseerida. Enne ehitamise juurde minemist tuuakse näiteid sarnastest olemasolevatest projektidest. Seejärel kirjeldatakse funktsionaalseid nõuded ja vajalikke komponente telefoni valmimiseks. Lõpus kirjutatakse valminud tulemustest.

4.1. Idee ning sarnased projektid

Idee on luua seade, mis tutvustaks tehnoloogiast ja prototüüpimisest huvitatud õpilastele lihtsa nuputelefoni tööpõhimõtet. Valminud seadmega, peaks saama vastu võtta kõnesid ning helistada numbrilaual valitud numbritele, sõnumeid vastu võtta ning saata hädaabisõnumit kindlale kontaktnumbrile. Samuti peab seade märku andma sissetulevast kõnest või sõnumist. Sarnaseid projekte võib leida internetist. Küll on aga enamus projekte tehtud puutetundliku ekraani põhimõttel. Näiteks on olemas õpetused valmistamaks ArduinoPhone koos vajamineva riistvara ning koodiga [36]. Samuti on sarnane õpetus Adafruit kodulehel [37].

4.2. Nõuded nuputelefonile

Antud punktis tuuakse välja, mis nõuded peavad olema täidetud.
Nuputelefonil peab olema ekraan ning numbrilaud.

1) Nuputelefon peab andma märku sissetulevast kõnest.
2) Nuputelefoniga peab saama võtta vastu sissetulevaid kõnesid.
3) Nuputelefoniga peab olema võimalik kõne lõpetada ühe nupuvajutusega.
4) Nuputelefoniga peab olema võimalik sisestatud numbrile helistada.
5) Numbrit valides peab olema võimalik olema ühe numbri kaupa kustutamine.
6) Nuputelefoniga peab saama ühe nupuvajutusega saata hädaolukorrasõnumi kindlale numbrile.
7) Nuputelefon peab andma märku uuest saabunud sõnumist.
8) Nuputelefoniga peab olema võimalik lugeda saabunud lühisõnumit.

Järgnevalt, et eelpool kirjeldatu ellu viia on vaja sobivaid komponente, mis nõuded täidaksid. Sobiv riistvara on kirjeldatud järgmises punktis

4.3. Vajalikud komponendid

Projekti elluviimiseks sobilikud komponendid on pandud tabelisse 2 koos hindadega. Arduino Uno arendusplaadile ehitatakse kogu projekt. GPRS Shield SIM900 kiibiga on vajalik telefoniühenduse loomiseks, et saata sõnumeid ja helistada. 0.96" OLED ekraan on vajalik kogu protsessi visuaalseks kuvamiseks. 4x4 numbrimaatriksit kasutatakse sisendi saamiseks, näiteks telefoninumbri trükkimiseks. RGB[13] valgusdioodi kasutatakse märguandjana erinevate olekute näitamiseks. Fototakistit on vaja ekraani väljalülitamise jaoks. Takistid on vajalikud RGB valgusdioodi ja fototakisti kasutamiseks. Maketeerimislaua ja ühendusjuhtmete abil ühendatakse komponendid omavahel.

Komponent	**eBay hind**	**Oomipoe hind**
Arduino Uno	4.43$ [38]	26.00€ [46]
GPRS Shield SIM900 kiibiga	15.78$ [39]	39.00€
0.96 tolline OLED ekraan	3.79$ [40]	13.00€ [47]
4x4 numbrimaatriks	1.01$ [41]	4.00€ [48]
Fototakisti	0.99$ [42]	1.00€ [49]
RGB valgusdiood	0.99$ [43]	1.00€ [50]
Takistid	1.84$ [44]	4x0.05€ [51]
Maketeerimislaud, ühendusjuhtmed	3.97$ [44]	3.90€ + 6.60€ [52, 53]
KOKKU	32.8$ (30.14€)	94.7€

Tabel 2. Nuputelefoni ehitamiseks vajalike komponentide loetelu ning hindade võrdlus internetipoega eBay ning Eesti elektroonikapoega Oomipood.

Loetletud komponentide kogumaksumus jääb internetisaidilt eBay tellides 30 euro kanti. Eestist ostes jääb loetletud komponentide maksumus 95 euro ringi.

Järgnevalt kirjeldatakse GPRS Shieldi kasutamist SIM900 teegiga ning tutvustatakse RGB valgusdioodi kasutamist. Ülejäänud vahendeid kirjeldati eelmistes peatükkides.

[13] RGB (*Red/Green/Blue*) – värviline valgusdiood punase, roheline ja sinise dioodiga

4.3.1. GPRS Shieldi kasutamine SIM900 teegiga

Kolmandas peatükis tutvustati GPRS Shieldi ning õpetati sellelt andmeid lugema kasutades *SoftwareSerial* teeki. Antud projektis on aga lihtsam kasutada teeki, mis on kirjutatud SIM900 kiibi jaoks. Selles teegis on erinevad kasulikud funktsioonid juba defineeritud ning implementeeritud. Näiteks helistamine ja sõnumite saatmine.

Järgnev seadistus on tehtud õpetuse järgi internetiallikast [24]. GPRS plaadiga suhtlemiseks on vajalik hankida SIM900 teek. See alla laadida GSMLib internetilehekülgelt [55] ja importida Arduino IDE-sse. Teegist tuleb teha mõned muudatused. Avada *GSM.cpp* fail ning muuta read *#define _GSM_TXPIN_ 2* ja *#define _GSM_RXPIN_ 3* vastvalt *#define _GSM_TXPIN_ 7* ja *#define _GSM_RXPIN_* 8 ning fail salvestada. Seejärel kontrollida GPRS plaadi pealt, kas jadaedastuse pesade valija on asendis D7/D8. Kui ei ole, siis tõsta sellese asendisse.

Avada fail *GSM.h* ning kommenteerida sisse rida *#define DEBUG_ON*, millega lülitatakse välja silumisrežiim. Samas failis muuta rida *#define GSM_ON 8* reaks *#define GSM_ON 9*. Pärast seda võib pesa 9 olla selles failis defineeritud veel mõneks sisendiks, kuid probleemide vältimiseks kommenteerida sisse kõik teised read, kus kasutatakse pesa 9. Seejärel fail salvestada. GSM_ON pesa kasutatakse GPRS plaadi automaatse sisselülitamise jaoks, vastasel juhul tuleks seda teha käsitsi plaadilt.

Programmi kirjutamise jaoks on vaja importida SIM900 ja SoftwareSerial teegid ridadega *#include "SIM900.h"* ja *#include <SoftwareSerial.h>*. Käsuga *gsm.begin(9600)* käivitatakse GSM-moodul ning käsuga *gsm.CheckRegistration()* kontrollitakse, kas moodul on ennast telefonivõrku ühendanud või mitte. Kui *begin(9600)* tagastab väärtuse 1, siis on moodul edukalt käivitatud, kui 0, siis käivitamine ebaõnnestus. Kui *CheckRegistration()* tagastab väärtuse 1, siis on moodul ühendatud võrku.

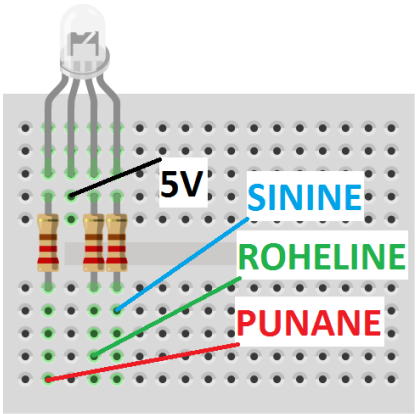
Et kasutada helistamise ja lühisõnumite saatmise võimalusi, tuleb importida programmi vastavad teegid ridadega *#include „call.h"* ja *#include „sms.h"*. Samuti tuleb luua nende klasside esindajad, näiteks *CallGSM call* ja *SMSGSM sms*. Vajalikud meetodid klassist *CallGSM* on *CallStatus()* ja *CallStatusWithAuth()*, mis ütlevad näiteks, kas hetkel on sissetulev kõne, aktiivne kõne või tegevuseta olek. Meetodiga *PickUp()* võetakse kõne vastu ja meetodiga

HangUp() pannakse kõne ära. Väljuva kõne saab sooritada meetodiga *Call(char *number_string)*, kus *number_string* on number, millele helistatakse.

Vajalikud meetodid klassist *SMSGSM* on *IsSMSPresent(byte required_status)*, kus *required_status* indikeerib, millist informatsiooni soovitakse. Kui *required_status* = *SMS_UNREAD*, siis tagastatakse esimese mitteloetud sõnumi asukoht SIM-kaardi mälust. Meetodiga *GetSMS(byte position, char *phone_number, char *SMS_text, byte max_SMS_len)* saab kätte saabunud sõnumi teksti ja saatja telefoninumbri. Argument *position* näitab, mitmes sõnum SIM-kaardi mälust loetakse. Tüüpiliselt kasutatakse sellena *IsSMSPresent(SMS_UNREAD)* tagastatud väärtust. *phone_number* muutujasse salvestatakse saatja telefoninumber, *SMS_text* muutujasse salvestakse sõnumi tekst ja *max_SMS_len* näitab, kui pikk võib sõnum olla. Tüüpiliselt on ühe SMS-sõnumi maksimaalne pikkus 160 tähemärki. Meetodiga *DeleteSMS(byte position)* saab kustutada SIM-kaardi mälus oleva sõnumi kindlalt asukohalt.

4.3.2. RGB valgusdiood

Tavalise RGB valgusdioodiga saab erinevaid värve kuvada kolme värvi (punane, roheline, sinine) erinevate kombinatsioonidega. RGB valgusdioodil on neli viiku. Viigud on järjestuses: punane, toide, roheline, sinine. Toiteviigust saab valgusdiood pinget, et põlema süttida. Teiste kaudu saab erinevaid värve tekitada. Värviviigud vajavad takisteid.



Joonis 32. RGB valgusdioodi viikude järjestus ja takistitega ühendamine.

Toiteviik tuleb ühendada arendusplaadi 5V pesaga. Värviviigud ühendada digitaalpesadega ning *setup()* funktsioonis väljundiks seadistada. Iga värviviik tekitab seda värvi, kuidas teda nimetatakse: punane punast valgust, roheline rohelist ja sinine sinist. Valgus süttib põlema, kui sellel liinil, kuhu valgusdioodi viik on ühendatud, viia väärtuseks 0. Väärtusel 1 on valgus kustus. Näiteks, kui on vaja punast valgust, siis sellel liinil, kuhu punane on ühendatud viia väärtus 0-ks ja teistel 1-ks. Värve peale punase, rohelise ja sinise saab tekitada, mitme valguse koosmõjul. Näiteks punane ja roheline valgus annavad kokku kollase valguse [56]. Joonisel 33 on näitekood pesade seadistamiseks.

```
int redPin = 9;
int greenPin = 10;
int bluePin = 11;

void setColor(int red, int green, int blue){ //meetod värvi muutmiseks
  digitalWrite(redPin, red);
  digitalWrite(greenPin, green);
  digitalWrite(bluePin, blue);
}
void setup(){
  pinMode(redPin, OUTPUT); //pesade seadistamine väljundiks
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);
}
void loop(){
  setColor(0, 1, 1);  // punane
  delay(1000);
  setColor(1, 0, 1);  // roheline
  delay(1000);
  setColor(1, 1, 0);  // sinine
  delay(1000);
  setColor(0, 0, 1);  // kollane
  delay(1000);
}
```

Joonis 33. Kood RGB valgusdioodi seadistamisest ja erinevate värvide kasutamine.

Värvide vahetamiseks on defineeritud abifunktsioon *setColor(int red, int green, int blue)*, mille argumentideks anda, missugused värvid panna süttima ja millised hoida kustus. *loop()* funktsioonis on näidatud nuputelefonis kasutatavate värvide kasutamine.

Kui keerukama riistvaraga on tutvutud, saab liikuda edasi terve nuputelefoni kokku ühendamiseks, millest kirjutatakse järgmises punktis.

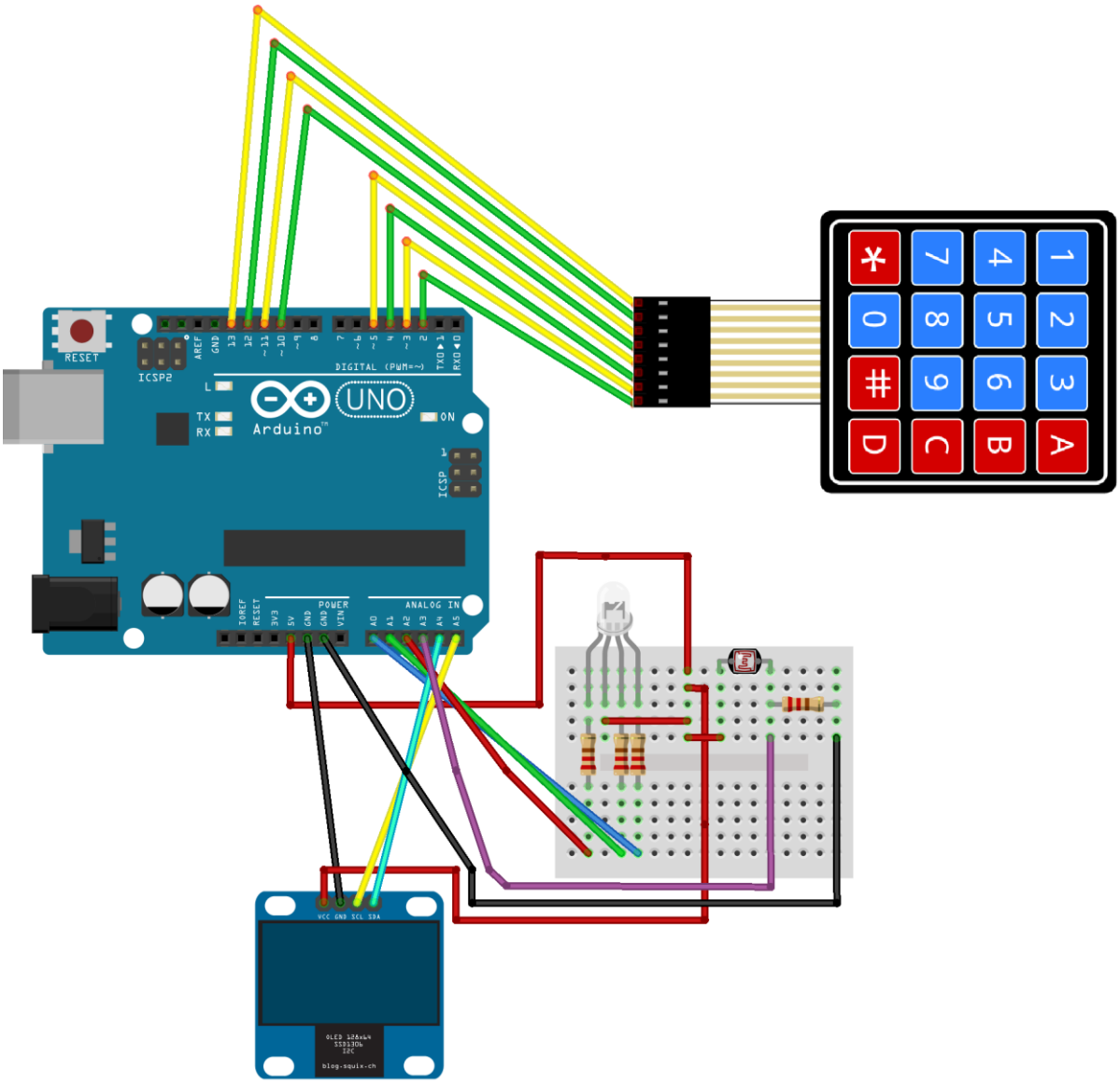
4.4. Riistvara ühendamine

Riistvara ühendatakse Arduino Uno arendusplaadi pesade külge. Punktis 4.3.1 kirjeldati GPRS Shieldi ühendamist Uno arendusplaadiga. Kuna GPRS Shieldi viigud ühendatakse täpselt kohakuti arendusplaadi omadega, siis selguse mõttes joonisel 34 pole kujutatud GPRS plaati, et oleks näha täpselt, milliste arendusplaadi pesadega on komponendid ühendatud. GPRS plaadi

võib ühendada ka arendusplaadi alla, sel juhul tuleb arendusplaadile alla joota viigud. Tuleb jälgida, et jadaedastuse pesade valija oleks asendis D7/D8 ja toiteallika lüliti ei oleks asendis EXTERN.

Numbrimaatriks on ühendatud ridadega 13-10 ja veergudega 5-2 pesadesse. Pesad 9, 8 ja 7 pole kasutatud, kuna neid kasutab GPRS plaat. Pesa 9 kasutab enda sisse lülitamise jaoks ja pesasid 7 ning 8 jadaedastuse jaoks.

RGB valgusdiood ühendada pesadesse A0, A1, A2 (programmis defineerida pesad digitaalpesade aadressidena vastavalt 14, 15, 16). Vastavalt sinine, roheline ja punane juhe.



Joonis 34. Riistvara ühendusskeem.

OLED ekraani SCL ühendada arendusplaadil pessa A5 ja SDA pessa A4. GND ühendada GND-ga ja VCC 5V-ga.

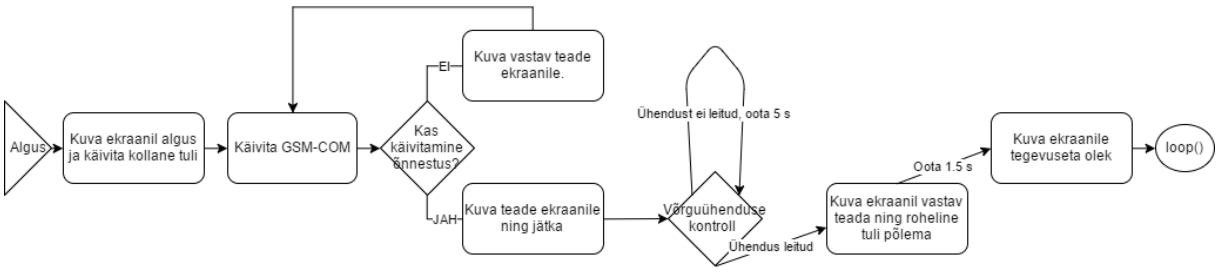
Fototakisti on ühendatud pesasse A3 ning vajab ka ühendust GND ja 5V pesaga. Kuna nii valgusdiood, ekraan kui ka fototakisti vajavad 5V, siis viime selle ühenduse läbi maketeerimislaua.

Kui füüsilised ühendused on tehtud, saab asuda riistvara programmeerimise juurde.

4.5. Nuputelefoni programmeerimine

Programmi kirjutamiseks avada Arduino IDE-s tühi sketš. Esmalt importida kõikide komponentide töötamiseks vajalikud teegid, defineerida ühendatud pesad ja luua vajalikud objektid.

Programmi *setup()* funktsiooni panna kõik, mis peab tehtud saama seadme sisse lülitamisel. Selleks seadistada RGB valgusdioodi ühendused ja GPRS plaadi viik 9 väljundiks ning alustada GSM-mooduli käivitamisega. *setup()* funktsiooni vooskeem on kujutatud joonisel 35.



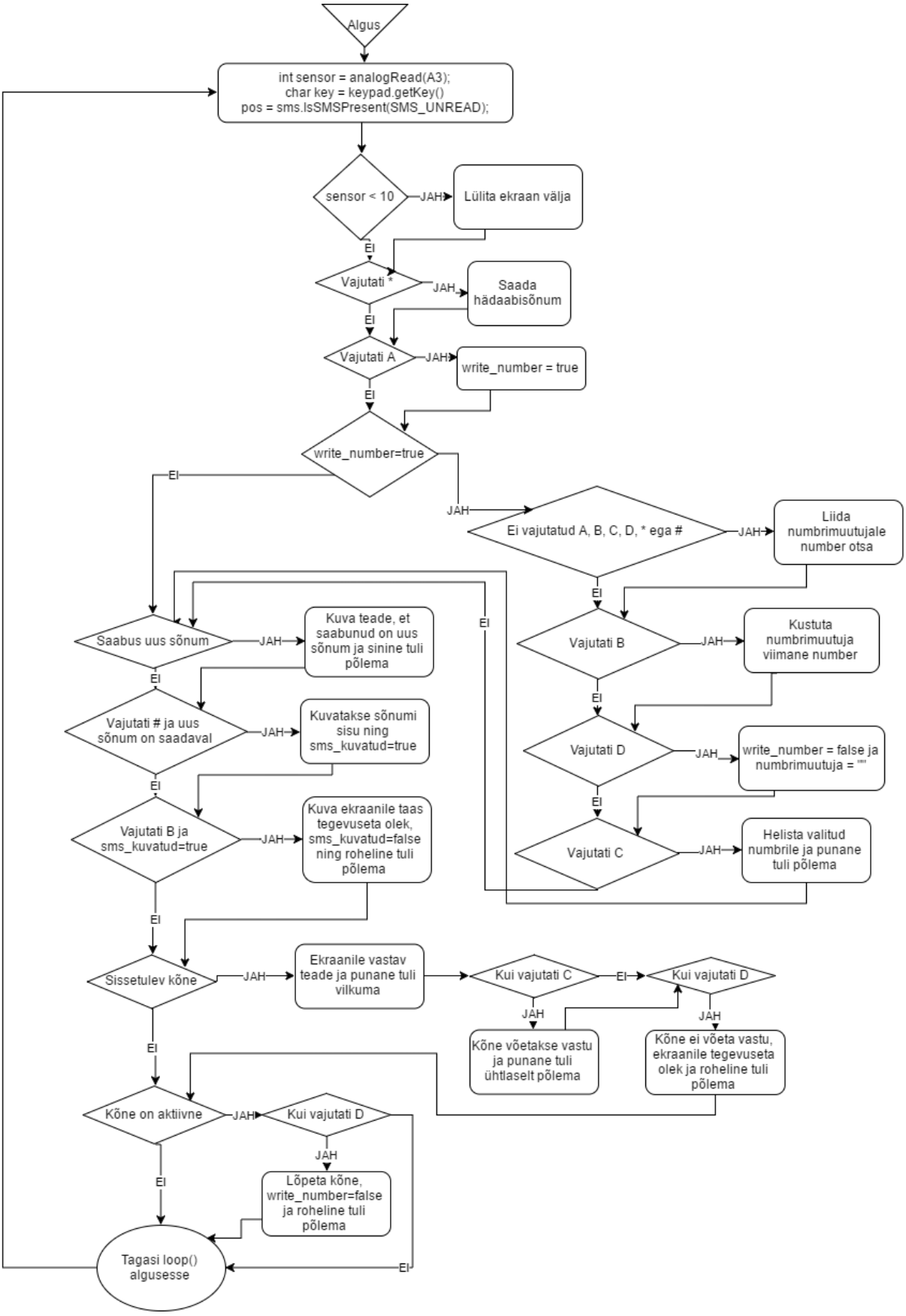
Joonis 35. Funktsiooni *setup()* vooskeem.

Kõigepealt tuleb ekraanile joonistada alguskuva, süüdata kollane tuli ning käivitada GSM-moodul. Kui käivitamine õnnestus, siis kuvada ekraanile vastav teade ning liikuda edasi, vastasel juhul üritada uuesti GSM-moodulit käivitada. Kui GSM-mooduli käivitamine õnnestus, tuleks kontrollida kas GSM-moodul on ühendanud ennast mobiilivõrku. Seda võiks teha tsükliga, mis iga viie sekundi tagant kontrolliks mobiilivõrku ühendatust. Kui ühendus on leitud, siis kuvada see ka ekraanile 1.5 sekundiks ning seejärel minna tegevuseta olekusse ehk ooteseisundisse. Süüdata ka kollase tule asemel roheline tuli, mis märgib, et kõik on laabunud probleemideta ning seade on valmis edasisteks tegevusteks.

Edasi jõuab programm *loop()* funktsioonini. Funktsioonis *loop()* on põhiidee registreerida erinevaid nupuvajutusi ja kontrollida mobiilivõrgu olekuid ning uute sõnumite olemasolu. Selle informatsiooni põhjal teha erinevaid otsuseid. Samuti fototakistilt saadud näidu põhjal

otsustada, kas ekraan lülitada välja või mitte. *loop()* funktsiooni vooskeem on kujutatud joonisel 36.

Algul loetakse sisse kogu vajalik informatsioon: fototakisti näit, nupuvajutuse tulemus ja uue sõnumi olemasolu. Fototakisti näit loetakse sisse *analogRead(),* nupuvajutus *getKey()* ja uue sõnumi olemasolu *IsSMSPresent(SMS_UNREAD)* funktsiooniga. Seejärel hakatakse nende andmete põhjal otsuseid tegema. Kui fototakisti näit on väga väike, alla kümne. See tähendab, et on väga pime ning sellist olukorda on lihtne tekitada andurile näpu peale panekuga. Sel juhul ekraan lülitatakse välja, vastasel juhul mitte. Vajutades * saadetakse hädaabisõnum koodis defineeritud kindlale telefoninumbrile. Sõnumi saatmisest antakse ekraanil märku. Nuppu A vajutades aktiveerub telefoninumbri valimise protsess. Pärast seda, kui A on vajutatud, saab numbreid valides trükkida ekraanile telefoninumbri. Nupuga B saab trükitud telefoninumbri viimasena sisestatud numbri kustutada. Nupuga C helistatakse valitud numbrile ning süttib punane valgus. Nupule D vajutades katkestatakse telefoninumbri trükkimine ja minnakse tagasi tegevuseta olekusse. Kui uus sõnum on saabunud antakse sellest ekraanil märku ning süttib sinine valgus. Kui uus sõnum on saabunud ja vajutatakse nuppu #, siis kuvatakse sõnumi sisu ja saatja telefoninumber ekraanile.



Joonis 36. Funktsiooni *loop()* vooskeem.

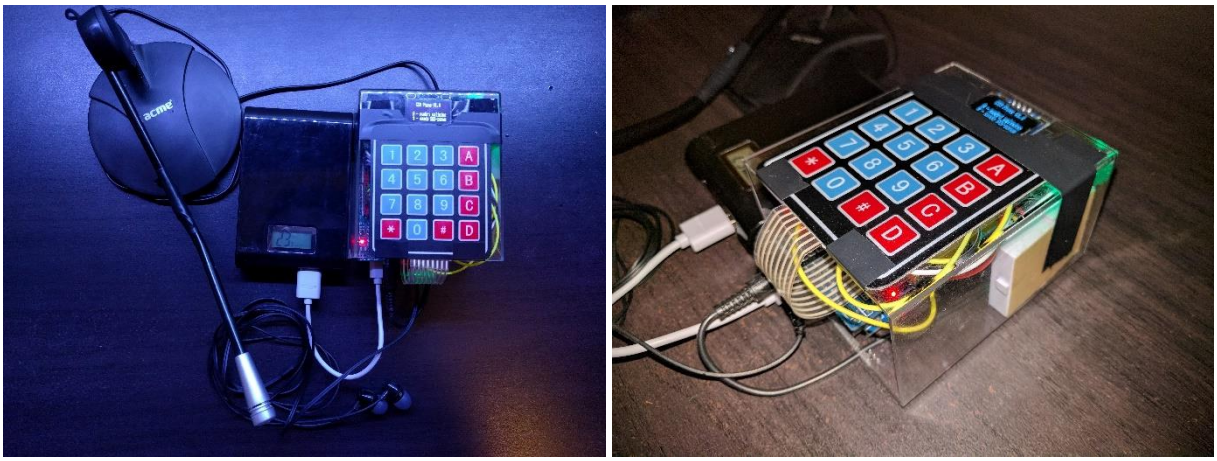
Kui sõnum on kuvatud ekraanile ja vajutatakse nuppu B, siis kuvatakse ekraanile taas tegevuseta olek ja süttib roheline valgus. Kui tuvastatakse funktsiooniga *CallStatusWithAuth(phone_number, 0, 0) == CALL_INCOM_VOICE_AUTH* sissetulev kõne, siis hakkab punane valgus vilkuma ning ekraanile kuvatakse teade helistaja numbriga. Kui selle ajal vajutatakse nuppu C, siis kõne võetakse vastu ja punane tuli jääb põlema ühtlaselt, kui aga nuppu D, siis kõnest loobutakse ning ekraanile kuvatakse tegevuseta olek ning süttib roheline valgus. Kui funktsiooniga *CallStatus() == CALL_ACTIVE_VOICE* tuvastatakse, et kõne on aktiivne, siis põleb punane tuli. Kui aktiivse kõne ajal vajutatakse nuppu D, siis kõne lõpetatakse ja ekraanile kuvatakse tegevuseta olek ja roheline valgus süttib. Järgnevalt alustab *loop()* funktsioon algusest peale kuni seadme väljalülitamiseni.

Antud töö autori poolt kirjutatud täielik kood on kättesaadav lisast I.

4.6. Tulemused

Töö tulemusena valmis prototüüp, millega saab helistada, kõnesid ja lühisõnumeid vastu võtta ning saata hädaabisõnumit. Telefon annab erinevat värvi tuledega märku, mis olekus seade hetkel on. Kollane valgus näitab, et telefon on sisse lülitatud, kuid pole veel võrku ühendatud. Roheline valgus tähendab, et telefon on võrku ühendatud ja tegevuseta olekus. Sinine valgus annab märku saabunud lühisõnumist. Punane vilkuv valgus annab märku sissetulevast kõnest ning punane ühtlane valgus, et kõne on aktiivne. Kõiki tegevusi kajastatakse ekraanil ning telefoni töö toimub nupuvajutustele reageerimise põhjal. Telefonil on fototakisti, mille abil saab ekraani välja lülitada, asetades käe või näpu anduri kohale. Tekkinud pimeduse tõttu lülitatakse ekraan välja ning ekraan aktiveerub taas uue oleku saabudes.

Valminud prototüüp on paigutatud karpi (joonisel 37). Prototüübi töötamiseks pole ühte kindlat toiteallikat. Sellele võib külge seadistada kompaktse aku või kasutada vahelduvvoolu adapterit, kui telefoni mobiilsus pole oluline. Antud töös kasutati akupanka.



Joonis 37. Nuputelefon karpi paigutatuna koos akupanga ning mikroni ja kõrvaklappidega.

Kõnedele audioväljundi ja –sisendi saamiseks tuleb GPRS plaadi 3.5mm pesadesse ühendada kõrvaklapid või kõlar ja mikrofon. Valminud seadmest saab teha edasiarendusi just mobiilsuse ja kompaktsuse poolest. Samuti võib väikse ekraani ning nupud asendada puutetundliku ekraaniga ja valmis teha nutitelefon.

Kokkuvõte

Käesoleva bakalaureusetöö eesmärgiks oli uurida Arduino platvormiga ühilduvaid andureid ja seadmeid, tutvustada erinevaid liideseid ja protokolle, neid kasutama õppida ning valmis ehitada nuputelefon. Esmalt tutvustati Arduino platvormi üldisemalt ning kirjeldati täpsemalt ühte Arduino platvormi arendusplaati Arduino Uno, millel toimus edasine töö ning valmis nuputelefon. Seejärel kirjutati anduritest ning nende jagunemisest väljundsignaali järgi kahte gruppi: analoogandurid ja digitaalandurid. Tutvustati nelja digitaalsignaalil põhinevat suhtlusprotokolli ja liidest: paralleelühendus, jadaedastus, SPI ja I^2C. Toodi näiteid nende liideste kasutamisest. Lõpus kirjeldati nuputelefoni ehitamise ideed, nõudeid ja komponente ning anti ülevaate töö tulemusest. Valminud bakalaureusetöö on sobiv materjal ühepäevastele robootika töötubadele koolidesse ja ülikoolidesse.

Töö käigus valmis nuputelefon, millega on võimalik valida number ning teha valitud numbrile kõnesid, sõnumeid ja kõnesid vastu võtta ning saata hädaabisõnum kindlale numbrile. Nuputelefoni ehitamine oli katse näidata, milliseid erinevaid komponente ning mis tehnoloogiat on vaja kasutada, et töötav telefon valmis ehitada. Eesmärk ei olnud ehitada kompaktne ning väga töökindel seade. Seega on telefoni stabiilsusega kohati probleeme.

Autori jaoks pakkus suurimat väljakutset erinevatest protokollidest arusaamine. Mõttetööd tuli teha palju. Praktilise osa ehk nuputelefoni ehitamine oli kõige huvitavam ning protokollide ja liideste osa kõige harivam.

Viidatud kirjandus

[1] Arduino. What is Arduino? https://www.arduino.cc/en/Guide/Introduction (25.01.2017)

[2] Arduino. Arduino Software (IDE). https://www.arduino.cc/en/Guide/Environment (12.04.2017)

[3] Arduino. Arduino UNO & Genuino UNO. https://www.arduino.cc/en/Main/ArduinoBoardUno (25.01.2017)

[4] Tawil Y. Understanding Arduino UNO Hardware Design, 2016. https://www.allaboutcircuits.com/technical-articles/understanding-arduino-uno-hardware-design/ (20.02.2017)

[5] Arduino. Access the Online IDE. https://www.arduino.cc/en/Main/Software (25.01.2017)

[6] Arduino. loop(). https://www.arduino.cc/en/reference/loop (25.01.2017)

[7] Arduino. Libraries. https://www.arduino.cc/en/reference/libraries (17.04.2017)

[8] Arduino. Writing a Library for Arduino. https://www.arduino.cc/en/Hacking/LibraryTutorial (17.04.2017)

[9] Tartu Ülikool. Andurid. *Robootikast puust ja punaseks*. https://sisu.ut.ee/robot/51-andurid (20.04.2017)

[10] WiseGEEK. What is an Analog Signal?. http://www.wisegeek.org/what-is-an-analog-signal.htm (27.02.2017)

[11] Sparkfun. Analog vs. Digital. https://learn.sparkfun.com/tutorials/analog-vs-digital (27.02.2017)

[12] Metshein. Arduino – Projekt 13: ilmajaam. http://www.metshein.com/unit/arduino-projekt-13-ilmajaam/ (10.05.2017)

[13] Resistorguide. Photo resistor. http://www.resistorguide.com/photoresistor/ (10.05.2017)

[14] Study.com. What are Digital and Analog Signals? - Definition & Explanation. http://study.com/academy/lesson/what-are-digital-and-analog-signals-definition-lesson-quiz.html (11.04.2017)

[15] ElectroSome. Alcohol Sensor Module – MQ3. https://electrosome.com/shop/alcohol-sensor-module-mq3/ (30.04.2017)

[16] DX. FC-22-H Combustible Gas Detector Sensor Module for Arduino (Works with Official Arduino Boards). http://www.dx.com/p/fc-22-h-combustible-gas-detector-sensor-module-for-arduino-179310#.WQXdYhPyjIV (30.04.2017)

[17] Sparkfun. Serial Communication. https://learn.sparkfun.com/tutorials/serial-communication (11.04.2017)

[18] Parallax, 4x4 Matrix Membrane Keypad (#27899). https://www.parallax.com/sites/default/files/downloads/27899-4x4-Matrix-Membrane-Keypad-v1.2.pdf (06.05.2017)

[19] Arduino, Keypad. http://playground.arduino.cc/Code/Keypad (06.05.2017)

[20] GSMA. GSM. https://www.gsma.com/aboutus/gsm-technology/gsm (03.05.2017)

[21] Search Mobile Computing. GPRS (General Packet Radio Services). http://searchmobilecomputing.techtarget.com/definition/GPRS (03.05.2017)

[22] Seeed. GPRS Shield V1.0. http://wiki.seeed.cc/GPRS_Shield_v1.0/ (03.05.2017)

[23] Arduino. SoftwareSerial Library. https://www.arduino.cc/en/Reference/softwareSerial (10.05.2017)

[24] Arduino. SoftwareSerial(rxPin, txPin, inverse_logic). https://www.arduino.cc/en/Reference/SoftwareSerialConstructor (10.05.2017)

[25] Arduino. SoftwareSerial: begin(speed). https://www.arduino.cc/en/Reference/SoftwareSerialBegin (10.05.2017)

[26] Arduino. SoftwareSerial: available(). https://www.arduino.cc/en/Reference/SoftwareSerialAvailable (10.05.2017)

[27] Arduino. SoftwareSerial: read. https://www.arduino.cc/en/Reference/SoftwareSerialRead (10.05.2017)

[28] Arduino. SoftwareSerial: write(data). https://www.arduino.cc/en/Reference/SoftwareSerialWrite (10.05.2017)

[29] Sparkfun. Serial Peripheral Interface (SPI). https://learn.sparkfun.com/tutorials/serial-peripheral-interface-spi (27.04.2017)

[30] Sparkfun. SparkFun Atmospheric Sensor Breakout - BME280. https://www.sparkfun.com/products/13676 (10.05.2017)

[31] Sparkfun. SparkFun BME280 Breakout Hookup Guide. https://learn.sparkfun.com/tutorials/sparkfun-bme280-breakout-hookup-

guide?_ga=2.60709610.677286678.1494454885-768907747.1483948315 (10.05.2017)

[32] Github. Sparkfun_BME280_Arduino_Library. https://github.com/sparkfun/SparkFun_BME280_Arduino_Library (10.05.2017)

[33] Sparkfun. I2C. https://learn.sparkfun.com/tutorials/i2c (12.04.2017)

[34] Arduino-er, Hello World 0.96 inch 128X64 I2C OLED, on Arduino Uno, using u8glib library. http://arduino-er.blogspot.com.ee/2015/04/hello-world-096-inch-128x64-i2c-oled-on.html (06.05.2017)

[35] Google Code Archive, u8glib. https://code.google.com/archive/p/u8glib/ (06.05.2017)

[36] Seeed Studio. ArduinoPhone. http://www.instructables.com/id/ArduinoPhone/ (06.05.2017)

[37] Adafruit. Arduin-o-Phone. https://learn.adafruit.com/arduin-o-phone-arduino-powered-diy-cellphone/overview (06.05.2017)

[38] eBay. UNO R3 MEGA328P ATMEGA16U2 Development board for Arduino + USB Cable SGHS. http://www.ebay.com/itm/172176663772 (10.05.2017)

[39] eBay. SIM900A 900/1800 MHz GPRS/GSM Development Board Module For Arduino New. http://www.ebay.com/itm/172172591722 (10.05.2017)

[40] eBay. 0.96in I2C IIC Serial 128X64 White OLED LCD LED Display Module SSD1306 Arduino. http://www.ebay.com/itm/222446871975 (10.05.2017)

[41] eBay. 4 x 4 AVR 16 Key Matrix Array For Arduino HOT Membrane Switch Keypad Keyboard. http://www.ebay.com/itm/331913224309 (10.05.2017)

[42] eBay. 20PCS Photoresistor GL5528 LDR Photo Resistors Light-Dependent TS. http://www.ebay.com/itm/172168836832 (10.05.2017)

[43] eBay. 10pcs x 5mm 4 pin RGB Diffused Common Anode cathode LED Red Green Blue. http://www.ebay.com/itm/122390251464?var=422868432447 (10.05.2017)

[44] eBay. Electronic Parts Pack Kit for Arduino Resistors LEDs Potentiometers Button Cap. http://www.ebay.com/itm/351899772864 (10.05.2017)

[45] eBay. Breadboard 400 point tie and 65pcs tie line Wire cable for Arduino DA . http://www.ebay.com/itm/182565305452 (10.05.2017)

[46] Oomipood. Arduino Uno Rev3. https://www.oomipood.ee/product/a000066_arduino_uno_rev3 (10.05.2017)

[47] Oomipood. OLED displei 0.96" valge 6 klemmi. https://www.oomipood.ee/product/ug_2864ksweg01_6pin_oled_displei_0_96_valge_6_klemmi (10.05.2017)

[48] Oomipood. Klahvistik maatriks 4x4. https://www.oomipood.ee/product/keyp_num_klahvistik_maatriks_4x4 (10.05.2017)

[49] Oomipood. Fototakisti 4mm 2-20k LDR. https://www.oomipood.ee/product/ldr04_fototakisti_4mm_2_20k_ldr?q=fototakisti (10.05.2017)

[50] Oomipood. LED5 RGB 5mm 5000mcd 4-viiku CA. https://www.oomipood.ee/product/500rgb4e_led5_rgb_5mm_5000mcd_4_viiku_ca?q=rgb (10.05.2017)

[51] Oomipood. 100K 1/4W. https://www.oomipood.ee/product/1_4w_100k_100k_1_4w (10.05.2017)

[52] Oomipood. Maketeerimislaua juhtmete komplekt 65tk. https://www.oomipood.ee/product/brdb_jump_maketeerimislaua_juhtmete_komplekt_65tk (10.05.2017)

[53] Oomipood. Maketeerimislaud 176*46mm 700 punkti. https://www.oomipood.ee/product/pps0700_maketeerimislaud_176_46mm_700_punkti (10.05.2017)

[54] Arduino Project Hub, GSM Home Alarm V1.0. https://create.arduino.cc/projecthub/Tiobel/gsm-home-alarm-v1-0-41e4dc (07.05.2017)

[55] GSMLib. http://www.gsmlib.org/download/GSM_GPRS_GPS_IDE100_v307_1.zip (07.05.2017)

[56] Adafruit. Arduino Lesson 3. RGB LEDs. https://learn.adafruit.com/adafruit-arduino-lesson-3-rgb-leds/arduino-sketch (06.05.2017)

Lisad

I. Nuputelefoni kood

```
// GSM nuputelefon
// Autor: Marvin Maran
// GSM-mooduli abil ehitatud nuputelefon,
// millega on  võimalik helistada, sõnumeid vastu võtta
// ning hädaabisõnumit saata.
// 2017

#include "SIM900.h" //GSM-mooduli teekide import
#include <SoftwareSerial.h>
#include "call.h"
#include "sms.h"

#include <Key.h> //numbrimaatriksi teekide import
#include <Keypad.h>

#include "U8glib.h" //OLED ekraani teeg import

#include <string.h> //sõnedega ümberkäimise jaoks vajalike teekide import
#include <stdio.h>

U8GLIB_SSD1306_128X32 ekraan(U8G_I2C_OPT_NONE); //OLED ekraan

int redPin = 16; //RGB valgusdioodi viigud
int greenPin = 15;
int bluePin = 14;

//klassid helistamise ja SMSi saatmiseks
CallGSM call;
SMSGSM sms;

char user_phone_number[] = "56206133"; //number hädaabi SMSi jaoks
char SOS_message[]="Olen h2das!"; //hädaabi SMSi sisu
char phone_number[15]; //muutuja telefoninumbri jaoks
boolean write_number = false;
String number2 = "";
char number[15];

char pos; //muutujad SMSi jaoks
int pos2;
char smsnumber[15];
char message[160];
boolean sms_kuvatud = false;

const byte ROWS = 4;
const byte COLS = 4;
char keys[ROWS][COLS] = {
  {'1','2','3', 'A'},
  {'4','5','6', 'B'},
  {'7','8','9', 'C'},
  {'*','0','#','D'}
};
byte rowPins[ROWS] = {13, 12, 11, 10}; //numbrimaatriksi ridade pesad
byte colPins[COLS] = {5, 4, 3, 2}; //numbrimaatriksi veergude pesad

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

```
//numbri joonistamine ekraanile
void drawnumber(const char *s) {
  ekraan.sleepOff();
  ekraan.drawStr(17, 7, "Nuputelefon V1.0");
  ekraan.drawStr(20, 23, s);
}

//väljuva kõne kuvamine ekraanil
void drawcall(const char *s) {
  ekraan.sleepOff();
  ekraan.drawStr(17, 7, "Nuputelefon V1.0");
  ekraan.drawStr(40, 15, "Helistan");
  ekraan.drawStr(40, 23, s);
}
//siseneva kõne kuvamine ekraanil
void drawcalling(const char *s) {
  ekraan.sleepOff();
  ekraan.drawStr(17, 7, "Nuputelefon V1.0");
  ekraan.drawStr(40, 15, "Helistab");
  ekraan.drawStr(30, 23, s);
}
//uue SMSi saabumise teadaanne
void newSMS(){
  ekraan.sleepOff();
  ekraan.drawStr(17, 7, "Nuputelefon V1.0");
  ekraan.drawStr(20, 15, "Saabus uus SMS!!!");
  ekraan.drawStr(10, 23, "Lugemiseks vajuta #");
}
//SMSi kuvamine ekraanil
void drawsms(const char *s, const char *m) {
  ekraan.sleepOff();
  ekraan.drawStr(17, 7, "Nuputelefon V1.0");
  ekraan.drawStr(20, 15, s);
  String message = (String) m;
  //kui SMS on pikem kui üks rida ekraanil siis peab sõnumi poolitama
  if(message.length() >= 18){
    char m1[20];
    char m2[20];
    strncpy(m1, m, 20);
    strncpy(m2, &m[20], 20);
    ekraan.drawStr(5, 23, m1);
    ekraan.drawStr(5, 31, m2);
  }
  else{
    ekraan.drawStr(5, 23, m);
  }
}
//SOS-sõnumi saatmise kuvamine
void sendSOS(){
  ekraan.sleepOff();
  ekraan.drawStr(17, 7, "Nuputelefon V1.0");
  ekraan.drawStr(5, 15, "SOS-sonum saadetud");
  ekraan.drawStr(30, 23, user_phone_number);
  ekraan.drawStr(30, 31, SOS_message);
}
//tegevuseta olekus ekraan
void drawidle(){
  ekraan.sleepOff();
  ekraan.drawStr(17, 7, "Nuputelefon V1.0");
  ekraan.drawStr(10, 23, "A - numbri valimine");
  ekraan.drawStr(10, 31, "* - saada SOS-sonum");
```

```
}
//RGB valgusdioodi värvi muutmine
void setColor(int red, int green, int blue){
  digitalWrite(redPin, red);
  digitalWrite(greenPin, green);
  digitalWrite(bluePin, blue);
}

void setup() {
  pinMode(9, OUTPUT); //GSM-mooduli toite sisse/välja lülitamise jaoks
defineeritud viik

  //RBG pinide seadistamine väljundiks
  pinMode(redPin, OUTPUT);
  pinMode(greenPin, OUTPUT);
  pinMode(bluePin, OUTPUT);

  digitalWrite(9, HIGH); //GSM-mooduli käivitamiseks vajalik protseduur
  delay(1200);
  digitalWrite(9, LOW);
  delay(3000);

  ekraan.setFont(u8g_font_6x12); //ekraanil tekstifondi valimine

  setColor(0, 0, 1);  // alustades põleb RGB kollaselt

  ekraan.firstPage();
  do{
      ekraan.drawStr(17, 7, "Nuputelefon V1.0");
      ekraan.drawStr(5, 15, "Yhendan GSM COMi...");
      } while( ekraan.nextPage() );

  if (gsm.begin(9600)) { //GSM modemi käivitamine
    ekraan.firstPage();
    do{
      ekraan.drawStr(17, 7, "Nuputelefon V1.0");
      ekraan.drawStr(5, 15, "Yhendan GSM COMi...");
      ekraan.drawStr(45, 23, "Valmis!");
      } while( ekraan.nextPage() );
  }
  else{
    do{
      ekraan.drawStr(17, 7, "Nuputelefon V1.0");
      ekraan.drawStr(5, 15, "Yhendan GSM COMi...");
      ekraan.drawStr(5, 23, "Ei saanud yhendada GSM modemit");
      } while( ekraan.nextPage() );
  }
  //võrguühenduse ootamine
  while (gsm.CheckRegistration() != 1) {
    delay(5000);
  }
 //kui võrk on leitud, siis teadaanne ja RGB roheliseks
 if(gsm.CheckRegistration() == 1) {
    ekraan.firstPage();
    do{
      ekraan.drawStr(17, 7, "Nuputelefon V1.0");
      ekraan.drawStr(5, 15, "Yhendan GSM COMi...");
      ekraan.drawStr(45, 23, "Valmis!");
      ekraan.drawStr(30, 31, "Vork leitud!");
      } while( ekraan.nextPage() );
 }
```

```
 setColor(1, 0, 1);  // roheline
 delay(1500);

 //ekraanile kuvatakse tegevuseta olek
 ekraan.firstPage();
    do{
      drawidle();
      } while( ekraan.nextPage() );
}

void loop() {
  int sensor = analogRead(A3); //loeb fototakisti näidu
  char key = keypad.getKey(); //tuvastab nupuvajutuse keypadilt
  pos = sms.IsSMSPresent(SMS_UNREAD); //kas uus SMS on kohal

  if(sensor < 10){ //kui sensori näit on alla 10 ehk kottpime (näiteks kui
näpp sensorile peale panna)
    ekraan.firstPage();
    do{
      ekraan.sleepOn(); //lülitatakse ekraan välja
    } while(ekraan.nextPage());
  }

  //hädasõnumi saatmine
  if(key == '*'){
    ekraan.firstPage();
    do{
      sendSOS();
    } while( ekraan.nextPage() );

    sms.SendSMS(user_phone_number, SOS_message);
    delay(1500);
    ekraan.firstPage();
    do{
      drawidle();
    } while( ekraan.nextPage() );
  }
  // kui vajutatakse A siis saab hakata numbrit trükkima
  if(key == 'A'){
    write_number = true;
    ekraan.firstPage();
    do{
      drawnumber("");
    } while( ekraan.nextPage() );
  }
  if(write_number){
    // kui numbri valimine on aktiivne siis kuvatakse sisestatud numbrit
    if(key != 'A' and key != 'B' and key != 'C' and key != 'D' and key !=
'*' and key != '#'){
      number2 += key;
      number2.toCharArray(number, 15);
      ekraan.firstPage();
      do{
        drawnumber(number);
      } while( ekraan.nextPage() );
    }
    // numbri trükkimine on aktiivne ja vajutatakse B siis kustutatakse
viimasena sisestatud number
    if(key == 'B'){
      //kustuta üks täht
      number2 = number2.substring(0, number2.length() - 1);
```

```
      number2.toCharArray(number, 15);
      ekraan.firstPage();
      do{
        drawnumber(number);
      } while( ekraan.nextPage() );
    }
    //katkesta numbri valimine
    if(key == 'D'){
      number2 = "";
      write_number = false;
      ekraan.firstPage();
      do{
        drawidle();
      } while( ekraan.nextPage() );
    }
    // helistamine valitud numbrile
    if(key == 'C'){
      //helista antud numbrile
      number2.toCharArray(number, 15);
      ekraan.firstPage();
      do{
        drawcall(number);
      } while( ekraan.nextPage() );
      if (call.CallStatus() != CALL_ACTIVE_VOICE) {
        call.Call(number);
      }
      setColor(0, 1, 1);  // punane
    }
  }
  if((int)pos > 0 && (int)pos <= 20){
    pos2 = pos;
    setColor(1, 1, 0);  // sinine
    ekraan.firstPage();
    do{
      newSMS(); //kuvatakse uue SMSi teadaanne
    } while( ekraan.nextPage() );
  }

  if(key == '#' && (int)pos2 > 0 && (int)pos2 <= 20){
    sms.GetSMS((int) pos2, smsnumber, message, 160);
    sms_kuvatud = true;
    ekraan.firstPage();
    do{
      drawsms(smsnumber, message);
      } while( ekraan.nextPage() );
    sms.DeleteSMS(pos2);
  }
  if(sms_kuvatud && key == 'B'){
    ekraan.firstPage();
    do{
      drawidle();
    } while( ekraan.nextPage() );
    setColor(1, 0, 1);  // roheline
    sms_kuvatud = false;
  }
  if(call.CallStatusWithAuth(phone_number, 0, 0) == CALL_INCOM_VOICE_AUTH)
{
    //siis kui kone tuleb sisse hakkab led vilkuma
    ekraan.firstPage();
    do{
      drawcalling(phone_number);
```

```
    } while( ekraan.nextPage() );
    setColor(0, 1, 1);  //punane vilkuma
    delay(30);
    setColor(1, 1, 1);

    //kõne vastu võtmine
    if(key == 'C'){
      //kui kõne võetakse vastu siis led põleb
      setColor(0, 1, 1); //punane
      call.PickUp();
    }
    //kõnest keeldumine
    if(key == 'D'){
      ekraan.firstPage();
      do{
        drawidle();
      } while( ekraan.nextPage() );
      call.HangUp();
      setColor(0, 0, 0);
      delay(50);
      setColor(1, 0, 1);  // roheline
    }
  }
  //kui kõne on aktiivne
  if(call.CallStatus() == CALL_ACTIVE_VOICE) {
    //kõne lõpetamine
    if(key == 'D'){
      call.HangUp();
      setColor(0, 0, 0);
      delay(50);
      setColor(1, 0, 1);  // roheline
      number2 = "";
      write_number = false;
      ekraan.firstPage();
      do{
        drawidle();
      } while( ekraan.nextPage() );
    }
  }
}
```

II. Litsents

Lihtlitsents lõputöö reprodutseerimiseks ja lõputöö üldsusele kättesaadavaks tegemiseks

Mina, **Marvin Maran**,

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

Arduino platvormiga ühilduvad andurid ja liidesed ning nuputelefoni ehitamine, mille juhendajad on Alo Peets, Anne Villems ja Taavi Duvin.

1.1. reprodutseerimiseks säilitamise ja üldsusele kättesaadavaks tegemise eesmärgil, sealhulgas digitaalarhiivi DSpace-is lisamise eesmärgil kuni autoriõiguse kehtivuse tähtaja lõppemiseni;

1.2. üldsusele kättesaadavaks tegemiseks Tartu Ülikooli veebikeskkonna kaudu, sealhulgas digitaalarhiivi DSpace´i kaudu kuni autoriõiguse kehtivuse tähtaja lõppemiseni.

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.

Tartus, **11.05.2017**