TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond

Lukas Sulg 179793

# Sudoku-masin

Bakalaureusetöö

Juhendaja: Peeter Ellervee
Professor / Ph.D

Tallinn 2020

# Autorideklaratsioon

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

Autor: Lukas Sulg

01.03.2021

# Annotatsioon

Käesolev bakalaureusetöö keskendub strateegilise numbrimängu sudoku loomisele ja lahendamisele mikrokontrolleril. Selleks realiseeritakse käesolevas töös mikrokontrollerile, Raspberry Pi Model B Plus, mängu genereerimise algoritm. Samuti, luuakse mängu sisendparameetrite sisestamiseks ühendus puutetundliku ekraani ja mikrokontrolleri vahel. Puutetundlikule ekraanile kuvatakse graafiline kasutajaliides, mille abil saab rakenduse kasutaja mängu kontrollida.

Mängu genereerimise algoritm programmeeriti mängu reeglite kohaselt. Sellest tulenevalt kasutas autor tagurdamise algoritmi ja konkreetseid algväärtuseid. Mängu keerukuse hindamiseks kasutab autor lihtsat vähendamismeetodit.

Projekt on realiseeritud programmeerimiskeeles C# kasutades keskkonda Unity. Kogu koodibaasi on võimalik näha autori GitLabi kodulehel https://gitlab.cs.ttu.ee/lusulg/sudoku.

Lõputöö tulemusena arendati mikrokontrolleril põhinev sudoku-masin koos lihtsalt kasutatava graafilise kasutajaliidesega ning koostati ka selle põhjalik analüüs. Lahendust on võimalik proovida nii androidseadmetel kui ka mikrokontrolleril. Bakalaureusetöö peetakse õnnestunuks, sest töö käigus realiseeritakse püstitatud eesmärgid.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 25 leheküljel, 6 peatükki, 24 joonist.

# Abstract

## Sudoku Machine

This bachelor's thesis focuses on creating and solving a strategic number game sudoku on a microcontroller. This is realized by connecting the microcontroller Raspberry Pi Model B Plus to a touch screen to enter game input parameters. Furthermore, a graphical user interface is displayed on the touch screen that allows the user of the application to control the game. Also, a game generation algorithm was implemented

The game generation algorithm was implemented according to the game rules. Consequently, the author used a reversal algorithm and specific initial values. To assess the complexity of the game, the author uses a simple reduction method.

The project is implemented in the C # programming language and uses the Unity environment. The entire code base can be found on the author's GitLab website https://gitlab.cs.ttu.ee/lusulg/sudoku.

As a result of the dissertation, a microcontroller-based sudoku machine with an easy-to-use graphical user interface was developed together with a thorough analysis. The solution can be tried on any android devices as well as on the authors microcontroller. The bachelor's thesis can be considered successful because the project the set goals were realized.

The thesis is in Estonian and contains 25 pages of text, 6 chapters, 24 figures.

# Lühendite ja mõistete sõnastik

| | |
|---|---|
| DPI | *Dots per inch*, punkti tolli kohta |
| AI | Arvutisüsteemide insituut |
| LCD | *Liquid-crystal display*,Vedelkristallkuvar |
| DSI | *Display Serial Interface*, Ekraani jadaliides |
| ARM | *Advanced RISC Machines*, Täiustatud RISC-masinad |

# Sisukord

# Jooniste loetelu

# 1 Sissejuhatus

Strateegiline numbrimäng sudoku koosneb harilikult 9x9 ruudustikust ehk 81 lahtrist. Mänguala on omakorda jaotatud veergudeks, ridadeks ja 3x3 regioonideks. Uut mängusessioni alustades on enamus lahtritest täidetud numbritega ühest üheksani. Mängu korrektseks lahendamiseks tuleb paigutada igasse lahtrisse number ühest üheksani sedasi, et üheski reas, veerus ja 3x3 ruudustikus ei esineks sama number mitu korra. Sudoku on ainult siis valiidne, kui mängu lahendamiseks leidub ainult üks võimalik lahend.

## 1.1 Ülesande püstitus

Antud bakalaureusetöö eesmärk on realiseerida mikrokontrolleril põhinev sudoku-masin. Selleks realiseeritakse käesolevas töös mikrokontrollerile, Raspberry Pi Model B Plus, mängu genereerimise algoritm. Genereerimise algoritm peab olema võimaline looma erineva keerukusega mänge hoides mängu valiidsena. Samuti, luuakse mängu sisendparameetrite sisestamiseks ühendus puutetundliku ekraani ja mikrokontrolleri vahel. Puutetundlikule ekraanile kuvatakse graafiline kasutajaliides, mille abil saab rakenduse kasutaja mängu kontrollida. Eesmärgi saavutamiseks hõlmab töö järgnevaid ülesandeid:

- Riistvara tellimine
- Riistvara seadistamine
- Graafilise kasutajaliidese loomine
- Mänguvälja ja väärtuste genereerimine
- Väärtuste sisestamine ja kontrollimine
- Tarkvara testimine

Töö sisuline osa jaguneb viieks. Esimesena tutvustatakse lugejale Sudoku lahendamisstrateegiaid. Teiseks tutvustab autor töös kasutatud riistvara ja selle ühendamise protsessi koos riistvaralise struktuuriga. Samuti esitleb autor alternatiivseid võimalusi sarnase tulemuse saamiseks. Kolmandana kirjeldatakse projektis kasutatud tarkvara ja selgitatakse mängu genereerimise algoritmi. Samuti kirjeldab antud peatükis

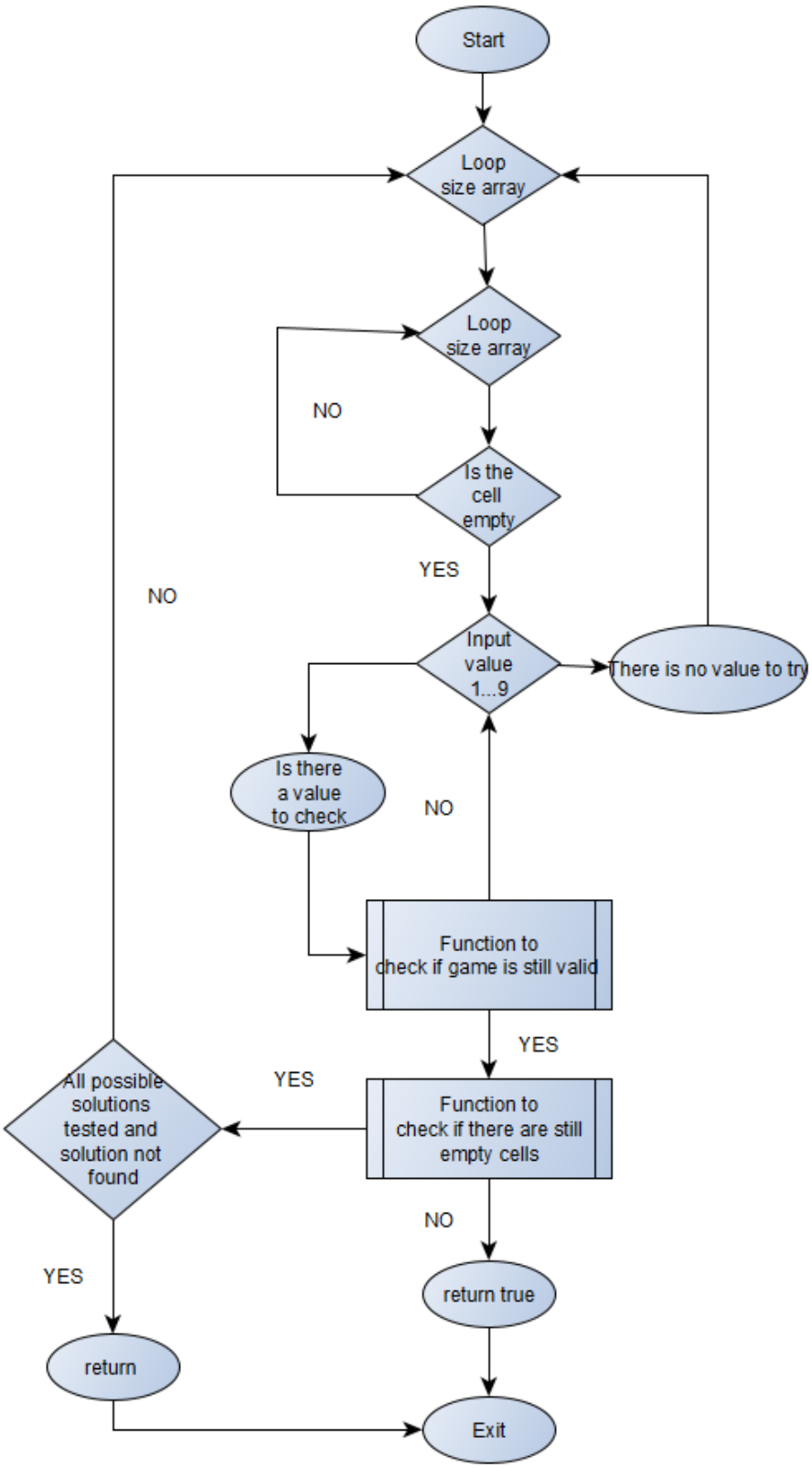
autor graafilise kasutajaliidese loomise protsessi ja visualiseerib autori poolt loodud tarkvaralise struktuuri. Järgnevalt keskendub autor erinevatele võimalustele projekti tarkvaraliselt edasi arendamiseks. Neljandaks kirjeldatakse lugejale arenduse jooksul läbiviidud teste ja testide tulemusi. Viiendas peatükis teeb autor kokkuvõtte projekti vältel tehtud tööst.

# 2 Sudoku lahendamise meetodid

Antud peatükis kirjeldatakse lugejale sudoku lahendusmeetodeid. Lahendusmeetodeid on võimalik kirjutada algoritmiks, mille kaudu saab genereeritud mängu lahendada. Erinevaid lahendusmeetodeid saab kasutada erinevate keerukuste mängude lahendamiseks. Prioritiseerides ja kombineerides kindalid lahendusmeetodeid on võimalik mängu lahendamisprotsessi kiirendada.

## 2.1 Tagurdamise lahendusmeetod

Tagurdamise lahendusstrateegia ehk *backtracking algorithm* on kasulik algoritm, kui ülesannet on võimalik lahendada kasutades rekursiivset meetodit[8]. Järgnevalt kirjeldab autor taguramise lahendusmeetodi tööpõhimõtet. Esmalt leiab lahendusstrateegia väärtustamata ruudu. Enne väärtuse sisestamist kontrollitakse, kas väärtust on ohutu määrata. Selleks kontrollitakse, et väärtust ei esineks samas veerus, reas või 3x3 ruudustikus. Pärast väärtuse määramist kontrollitakse rekursiivselt, kas sisestatud väärtus viib lahenduseni või mitte. Kui määratud arv ei vii mängu lahenduseni, siis proovitakse teist lahendust. Juhul kui lahend puudub tagastab programm sisestatud maatriksi. Järgnevalt esitab autor tagurdamise lahendusmeetodi vooskeemi:




Joonis 1 Tagurdamise lahendusmeetodi vooskeem

## 2.2 Loogilised lahendusmeetodid

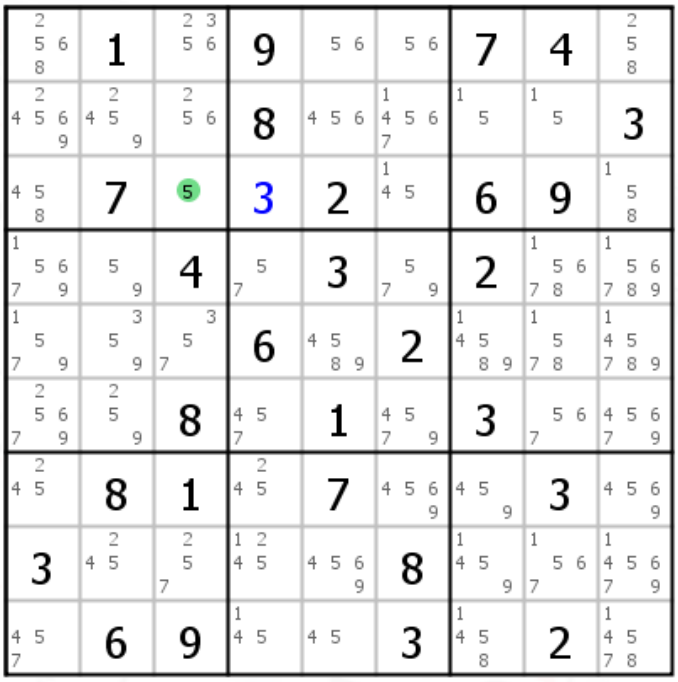
Antud peatükis kirjeldatakse loogikal põhinevaid lahendusmeetodeid. Lahendusmeetodid on jagatud alapeatükideks tulenevalt lahendusstrateegia keerukusest.

### 2.2.1 Lihtsad lahendusmeetodid

Üksikud, 3x3 ruudustikus, veerus või reas täidetud lahtrid on mängu kõige algelisemad probleemid[10]. Seetõttu ei pea lahendamiseks tegema eeldusi ja väärtus saab olla ainult ülejäänud number. Sellist lahendusmeetodit on hea kasutada mängu algul, et täita võimalikult palju lahtreid kergesti. Algoritmi teostamisel on hea idee lisada üksikute lahtrite kontroll esimeseks tegevuseks. Selline lahendamine teeb programmi kiiremaks, kuna vajab vähem tsükleid kui umbkaudne pakkumine. Üksikud lahtrid jagunevad järgnevateks probleemideks[11]:

- *Naked Single*
- *Hidden Single*

Järgnevalt illustreerib autor juhtu *Naked Single*. Pildilt on võimalik näha, et esile tõstetud lahtrit täitmiseks ei ole vaja tuletada, vaid saab koheselt täita jälgides sudoku reegleid. Illustreerival pildil oleks tegu väärtusega viis[10].



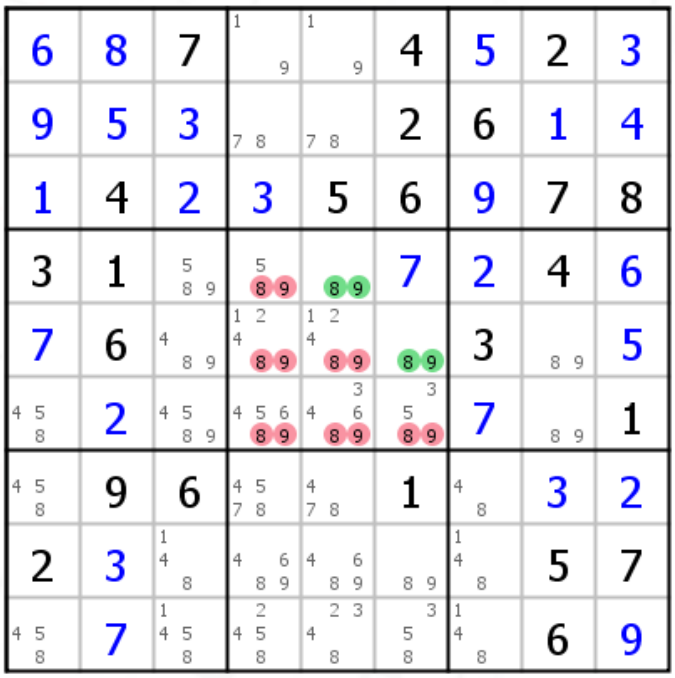
Joonis 2 Naked Single lahendusmeetod

### 2.2.2 Kerged lahendusmeetodid

Kerged lahendusmeetodid tähendavad seda, et valitud 3x3 ruudustikus, veerus või reas puudub kaks väärtust. Paaride väärtuste leidmiseks kasutatakse indekseerimist. Sellistel juhtudel on efektiivsem need väärtused sisestada enne keerukamaid meetodeid. Lihtsamate väärtuste leidmine teeb programmi efektiivsemaks ja vajab vähem rekursiooni, mille tulemusena probleemi lahend leitakse kiiremini[10]. Kerged lahendusmeetodid jagunevad järgnevateks alam probleemideks[11]:

- *Naked Pair*
- *Hidden Pair*
- *Pointed Pairs*

Järgnevalt illustreerib autor juhtu *Naked Pair*. Probleemi lähemalt vaadeldes leiame, et mõlemad esile tõstetud lahtrid saavad olla ainult väärtusega kaheksa või üheksa. Sellest tulenevalt ei ole võimalik koheselt sisestada nendesse lahtritesse väärtust. Probleemi lahendamiseks võime kasutada tagurdamise algoritmi või loogika alusel täita samas alam ruudustikus järgneva kasti, mis joonisel oleks väärtusega viis[10].



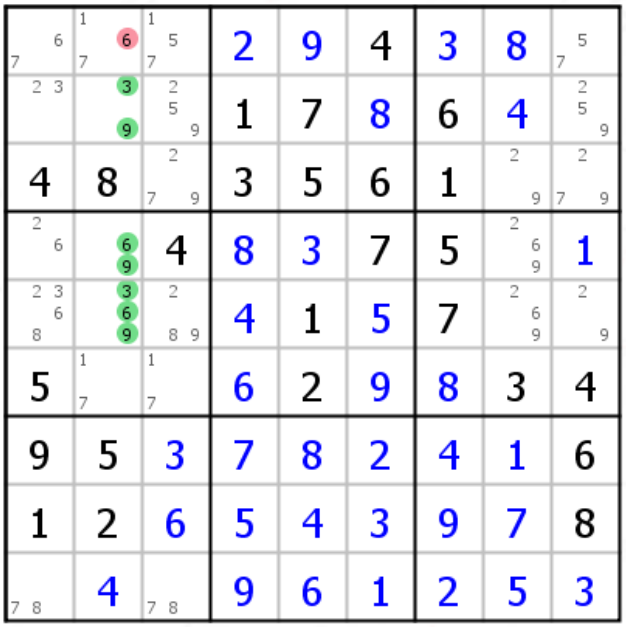
Joonis 3 Naked Pair lahendusmeetod

### 2.2.3 Mõõduka keerukusega lahendusmeetodid

Mõõduka keerukusega lahendusmeetodid on olukordadeks, kui valitud 3x3 ruudustikus, veerus või reas puudub kolm kuni neli väärtust. Mõõduka keerukusega lahendusi on viis erinevat. Selliseid olukordi tekib sudoku lahendamisel tihti. Selle klassi ülesanded sarnanevad kergete lahendus meetoditega. Lahendused algoritmide kirjutamiseks ja programmi implementeerimiseks on ajaliselt kulukas ja keeruline töö. Mõõduka keerukusega lahendusmeetodid on järgnevad[11]:

- *Naked Triple*
- *Naked Quad*
- *Pointing Triples*
- *Hidden Triple*
- *Hidden Quad*

Järgnevalt illustreerib autor juhtu *Naked Triple.* Joonist lähemalt uurides näeme, et samas veerus on kolm võimalikku kandidaadi väärtust. Sellest tulenevalt saame väita, et üleval asuv väärtus ei tohi olla kuus. Kasutades mustrit saame sisestada väärtuse vasakusse ülanurka, mille kaudu mäng laheneb kiirelt.



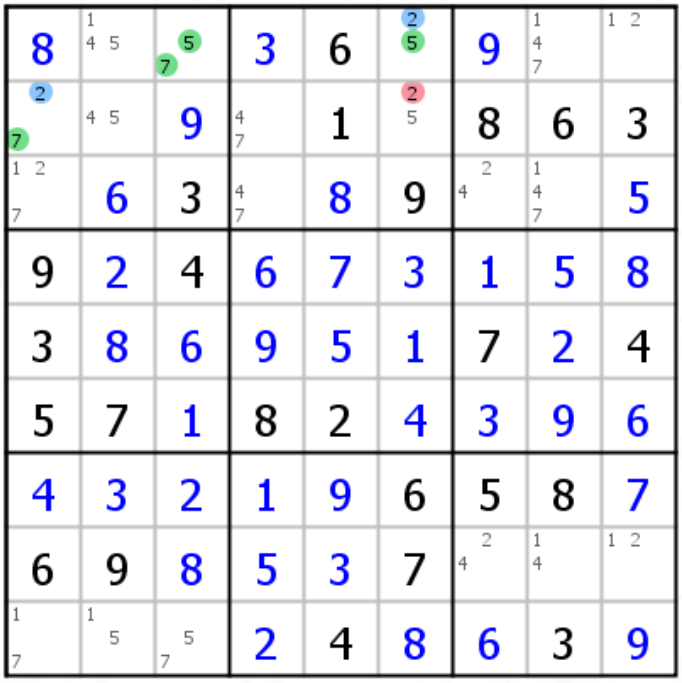
Joonis 4 Naked Triple lahendusmeetod

### 2.2.4 Keerulised lahendusmeetodid

Keerulisi lahendusmeetodeid kutsutakse kui *Wing* või *Fish* meetoditeks. Sellised meetodid kasutavad mustreid, mille järgi proovitakse leida korrektne väärtus, vaadates mitmeid lahtreid korraga. *Wing* ja *Fish* meetoditel on väga palju erinevaid variatsioone, mille tulemusena on algoritmiselt väga keeruline teostada. Keerulised lahendusmeetodid on järgnevad[11]:

- *XWing*
- *Swordfish*
- *Jellyfish*
- *XYWing*
- *XYZWing*

Üks tuntud keeruline meetod on *XWing* muster. Selline muster esineb, kui kaks rida või veergu sisaldavad mõlemad ainult kahte lahtrit, milles on vastav väärtus. Selline väärtus peab olema mõlemas reas ja jagama sama kahte veergu või vastupidi. Probleemi lahendamine toimub kandidaatide eemaldamise kaudu. Järgnevalt illustreerib autor juhtu *XWing*[10]:



Joonis 5 XWing lahendusmeetod

# 3 Riistvara

Antud peatükis on kirjeldatud projektis kasutatud riistvara. Järgnevalt kirjeldab autor riistvara seadistamise protsessi ja struktuuri. Viimaks tutvustab autor lugejale alternatiivsed riistvara lahendusi.

## 3.1 Kasutatav riistvara

Mikrokontrollerid ehk teisisõnu ühe kiibiga mikroarvuteid on tänapäeval mitmeid. Mikroarvutid sarnanevad turing masinaga, ehk vaatamata mudeli lihtsusele suudab masin simuleerida kõiki arvuti algoritme. Olgu see siis serveri loomine või sudoku-masin realiseerimine.

Üks tuntumaid mikrokontrollereid on Raspberry Pi. Autori valis mikrokontrolleriks raspberry pi 3 model plus. Tegemist on ühe võimsama mikrokontrolleriga, mille peal asub 1.4GHz ARM protsessor[1]. Lisaks on plaadil palju erinevaid väljundeid, mille kaudu saab ühendada välisseadmeid. Mikrokontrolleri valimisel pidas autor oluliseks hinda ja võimekust. Lisaks pidas autor väga tähtsaks tarkvara, mida mikrokontroller peab olema võimeline toetama.

Joonis 6 Mikrokontroller Raspberry Pi 3 Model Plus

Raspberry pi toetab laialt populaarseid operatsioonisüsteeme. Operatsioonisüsteemid annavad võimaluse kasutada varieeruvaid programmeerimiskeeli ja töökeskkondi. See teeb Raspberry heaks vahendiks, mille peale ehitada retro mänge või veebiservereid.

Puutetundliku ekraani valimisel pidas autor tähtsaks töökindlust, kvaliteeti ja ühendamise võimalusi mikrokontrolleriga. Puutetundlik ekraanile peab olema võimeline kuvama  välja töödeldud tulemust. Samuti mängib tähtsat rolli ekraani suurus. Mida suurem ekraan, seda raskem on teda kaasas kanda. Samas võib väike ekraan piirata kasutamist. Selle probleemi lahendamiseks valis autor mikrokontrollerile, Raspberry Pi Model B Plus jaoks mõeldud 7 tollise puutetundliku ekraani[2].

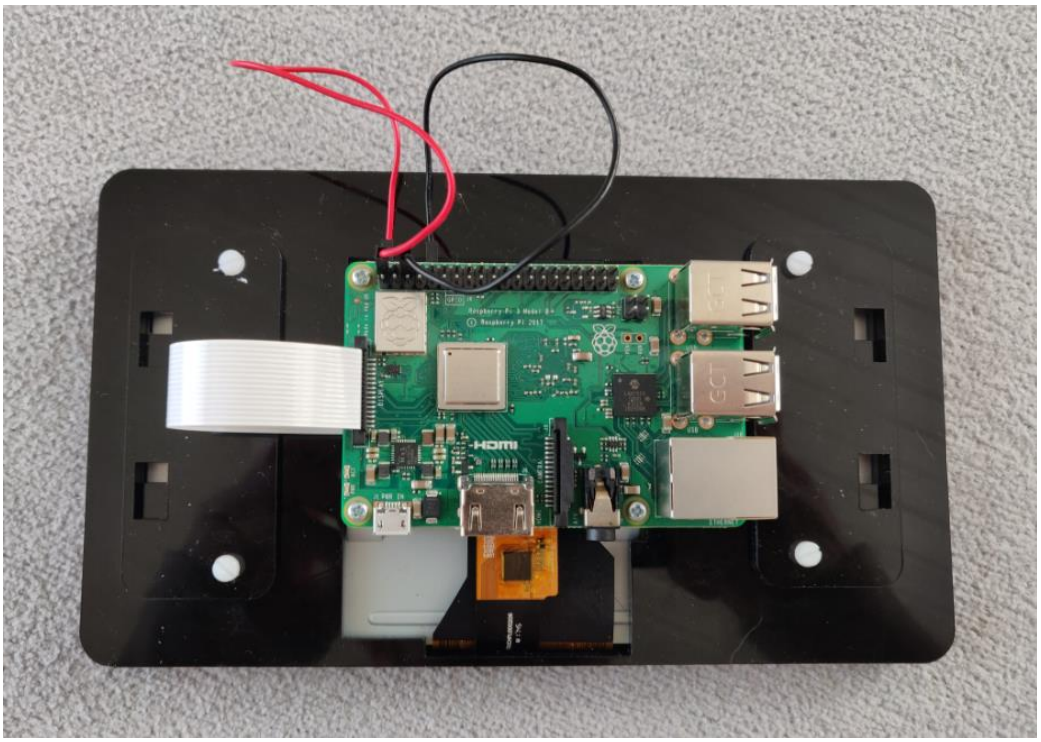
Joonis 7 Mikrokontroller Raspberry Pi 3 Model Plus

## 3.2 Riistvara seadistamine

Riistvara seadistamise protsess on autori sõnul üheselt struktureeritud. Ühendamiseks on vaja järgnevaid abivahendeid:

- DSI lintkaabel
- 2x *Jumper wire*
- Toitepinge: 200 mA, 5 V (micro USB B)

Esmalt tuleb keerata ekraan ümber ja leida tagumisel plaadil asuv DSI lintkaabli adapter. Avades adapteri korpuse saame libistada sisse lintkaabli ja korpuse mõlemal küljel asuvad mustad kinnitused kinnitada. Järgnevalt tuleb ühendada *jumper wires*. Kasutades raspberry pi andmelehte leiame korrektsed tihvid, kuhu kinnitada maandus ja sisendvool. Maanduseks kasutas autor musta juhet ja sisendvooluks punast. Viimaks tuleb mikrokontrollerile kinnitada ekraan, kasutades plaadi taga leiduvad kinnitusi. Võimalusel soovitab autor lisada mikrokontrollerile ja ekraanile katte, mille abil kaitsta riistvara välistingimuste eest. Ühendatud riistvara näeb välja järgnev:

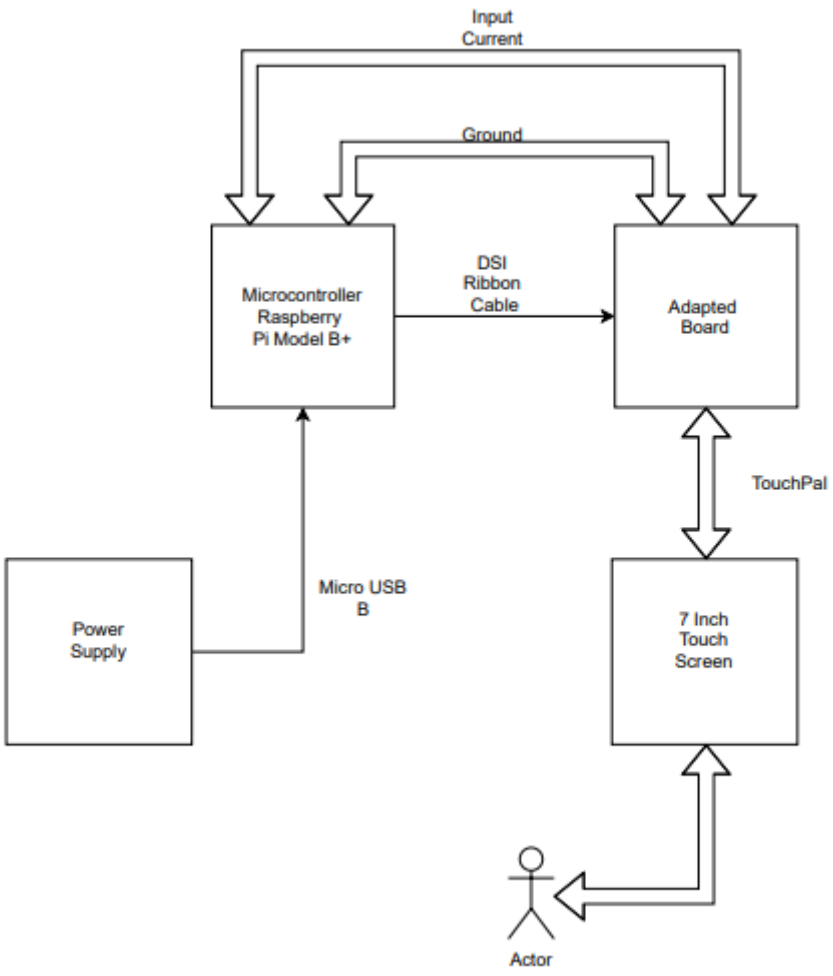


Joonis 8 Ühendatud Riistvara

Operatsioonisüsteemi paigaldamine mikrokontrolleritele erineb traditsioonilisest paigaldusest. Esmalt tuleb valida sobiv operatsioonisüsteem, mis toetab valitud mikrokontrollerit. Tulenevalt operatsioonisüsteemi mahukusest tuleb kontrollida, et kasutatav micro sd kaardi mahtuvus on vähemalt 8 Gb. Järgnevalt tuleb valitud operatsioonisüsteemi laadida micro sd kaardile. Kuna mikrokontrolleritele tehtavad operatsioonisüsteemid on tavaliselt .iso või .img failid, peab kasutama kolmandat tarkvara, et failid korrektselt kaardile lugeda. Selleks kasutas autor tarkvara nimega „Etcher“[5]. Etcher on võimas operatsioonisüsteemi pildi vilkur, mis kaitseb kasutajat kõvaketaste ülekirjutamise eest. Sisestades ettevalmistatud micro sd kaardi mikrokontrollerisse, saame enda mikro arvutile operatsioonisüsteemi. Autor otsustas kasutada „LineageOS“, mis on Android-põhine operatsioonisüsteem. LineageOS koosneb peamiselt tasuta ja avatud lähtekoodiga tarkvarast, mida on võimalik kasutajal enda vajadustele kohandada [4].

## 3.3 Riistvaraline struktuur

Riistvaraline struktuur aitab lugejal paremini mõista komponentide omavahelisi seoseid. Samuti annab mudel parema ülevaate, kuidas komponendid üksteisega ühilduvad. Lisaks annab riistvaraline arhitektuur tarkvara arendamiseks vajalikku informatsiooni. Järgnevalt toob autor esile illustreeriva pildi riistvaralisest struktuurist:

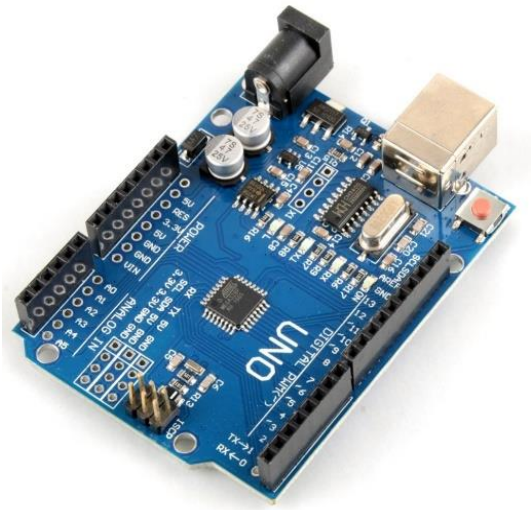



Joonis 9 Riistvara struktuur

## 3.4 Alternatiivsed võimalused

Mikrokontrollereid on lai valik ja suur osa mikrokontrolleritest ehitatakse kindla eesmärgi saavutamiseks. Näiteks Nividia jetson nanao miktrokontroller, mille peamisteks kasutusvaldkondadeks on objekti tuvastamine ja närvivõrkude töötlemine. Tihtipeale on sellised mikrokontrollerid mõeldud professionaalidele ja selle tulemusena kallis.

Alternatiivseks valikuks soovitab autor kasutada Arduino Uno R3 mikrokontrollerit [6]. Tegemist on soodsama lahendusega, millega on võimalik saavutada autoriga sarnane lõpptulemus. Arduino Uno R3 on arduino perekonna kõige enam kasutatud ja dokumenteeritud mikrokontroller. Arduino kahjuks ei toeta laialt kasutatud töökeskondi, mille tulemusena tavaliselt ehitatakse projekt c++/c programmeerimiskeeles.




Joonis 10 Arduino Uno R3 mikrokontroller

Puutetundlikuks ekraaniks soovitab autor kasutada Arduino mega hdmi puutetundlikku ekraani -7 tolline LCD[7]. Ühendamise protsess on hästi dokumenteeritud, mille tulemusena protsess muutub lihtsamaks.

Joonis 11 Arduino Mega HDMI puutetundlik ekraan

# 4 Tarkvara

Antud peatükis on kirjeldatakse projektis kasutatud tarkvara. Seejärel tutvustatakse lugejale mängu genereerimise algoritmi ja lahendusmeetodit. Järgnevalt tutvustab autor lugejale tarkvalaist struktuuri. Viimaks kirjeldab autor edasiarengu võimalusi. Autori poolt loodud tarkvaralist lahendust on võimalik näha autori GitLabi kodulehel https://gitlab.cs.ttu.ee/lusulg/sudoku.

## 4.1 Kasutatud tarkvara

Kasutatud tarkvara ja riistvara määravad tarkvaraloomeprotsessi. Autor otsustas kasutada arendusplatvormi Unity, täpsemalt Unity 2019.3.13f1. Unity on integreeritud loomevahend videomängude ja muu säärase loomiseks. Unity toetab arenduseks programmeerimiskeelt C#, mis on kaasaegne objektorienteeritud programmeerimiskeel. Objektorienteeritud programmeerimine annab meile võimaluse kasutada agiilset tarkvaraarenduse meetodit, mis teeb arenguloomeprotsessi kiiremaks ja töökindlamaks. Lisaks on unity skriptimine hästi dokumenteeritud, mille tulemusena uutel arendajatel on lihtsam õppida unity arenduskeskonda ja temaga kaasnevat programmiliidest.. Programm on kirjutatud integreeritud arenduskeskkonnas Visual Studio.

## 4.2 Mängu genereerimine

Sudoku loomiseks tuleb esmalt tagada mängu valiidsus. Sudoku on ainult siis valiidne, kui leidub unikaalne lahend [8]. Tulenevalt mängu loomusesest pidas autori tähtsaks mängu robustsust ja kiirust. Selle realiseerimiseks peame algväärtustama algse mänguvälja sedasi, et üheski veerus, reas ja 3x3 regioonis ei esineks numbrite kordust. Protsess toimub väärtuste nihutamise kaudu. Järgnev pilt illustreerib loodud alust Sudokule.

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |
| 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 |
| 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 |
| 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 |
| 5 | 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 |
| 8 | 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 |
| 3 | 4 | 5 | 6 | 7 | 8 | 9 | 1 | 2 |
| 6 | 7 | 8 | 9 | 1 | 2 | 3 | 4 | 5 |
| 9 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 |

Joonis 12 Sudoku alus

Seejärel tuleb mänguvälja segada, et tagada mängu varieeruvus. Selleks kasutab autor kahte segamismeetodit - ridade ja veergude vahetus nende vastavates 3x3 regioonides. Eesmärk on hoida mängu valiidsust ja lahendust korrektsena. Lisaks teeb see mängu töökindlamaks. Segamise tulemusel saame järgneva mänguala:

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| 2 | 1 | 3 | 6 | 5 | 4 | 7 | 8 | 9 |
| 8 | 7 | 9 | 3 | 2 | 1 | 4 | 5 | 6 |
| 5 | 4 | 6 | 9 | 8 | 7 | 1 | 2 | 3 |
| 3 | 2 | 4 | 7 | 6 | 5 | 8 | 9 | 1 |
| 6 | 5 | 7 | 1 | 9 | 8 | 2 | 3 | 4 |
| 9 | 8 | 1 | 4 | 3 | 2 | 5 | 6 | 7 |
| 4 | 3 | 5 | 8 | 7 | 6 | 9 | 1 | 2 |
| 1 | 9 | 2 | 5 | 4 | 3 | 6 | 7 | 8 |
| 7 | 6 | 8 | 2 | 1 | 9 | 3 | 4 | 5 |

Joonis 13 Segatud sudoku mänguala

Kolmandaks sammuks on väärtuste eemaldamine mängualalt. Eemaldatud väärtuste arv määrab mängu keerukuse, kuid puuduvad kindlad reeglid, mille alusel keerukust hinnatakse. Eemaldamine toimub mängualalt pseudojuhuslikult. See tähendab, et rakendusele on eelnevalt määratud mängualal asetsevate ridade ja veergude arv, mille tulemusena programm võib eemaldada sama väärtust mitu korda. Keerukuse määramiseks otsustas autor kasutada järgnevaid väärtuseid:

- Kerge – Mänguväljalt eemaldatakse 20 või vähem väärtust
- Keskmine – Mänguväljalt rohkem kui 25 ja vähem kui 30 väärtust
- Raske – Mänguväljalt eemaldatakse rohkem kui 35 väärtust

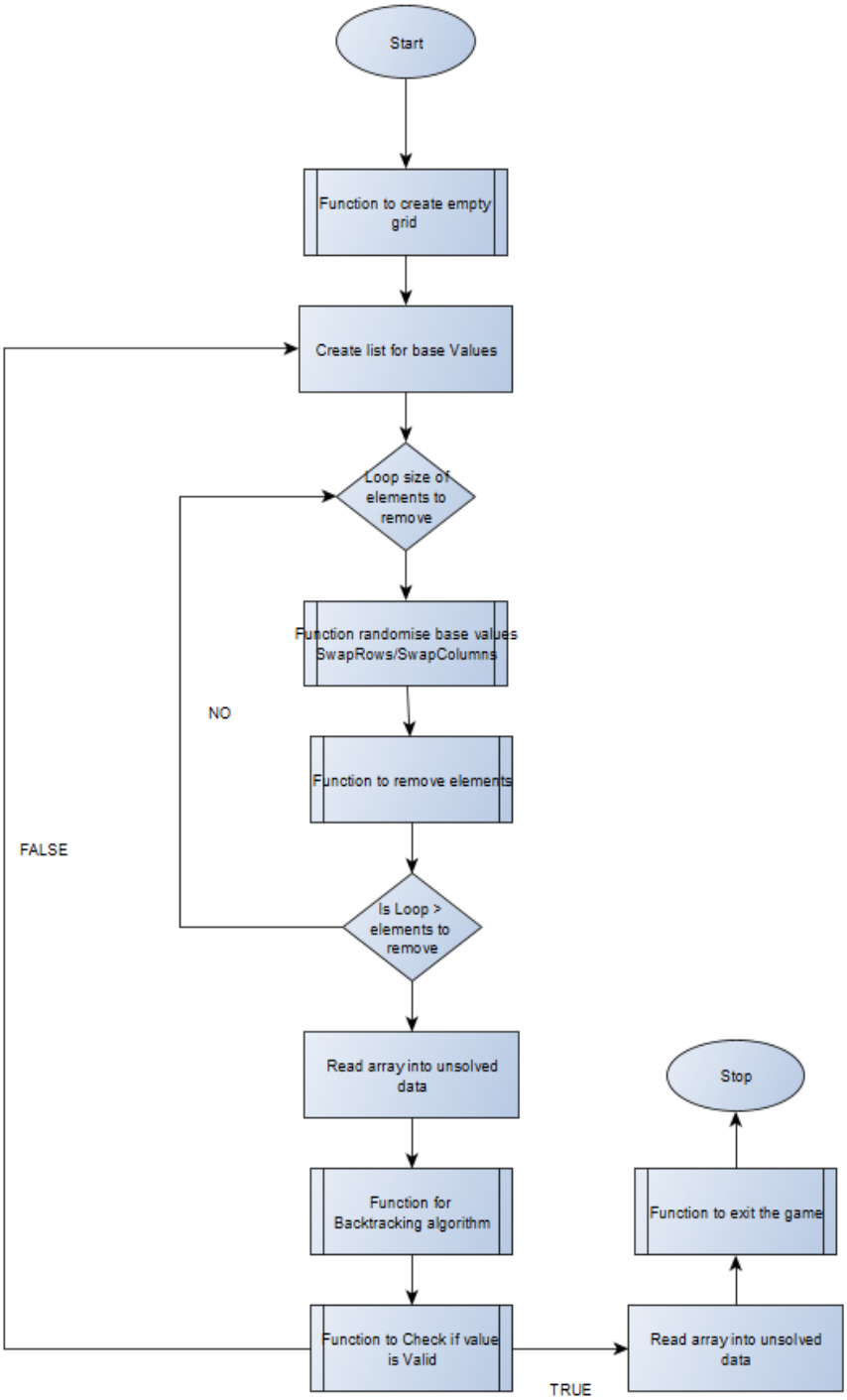
Väärtuste eemaldamisel tuleb lisaks jälgida järgmisi reegleid:

1. Mängualale peab jääma ainult üks võimalik lahend
2. Eemaldamine peab olema võimeline looma erineva keerukusega mänge
3. Raskusaste peab olema ligikaudu sama mängija poolt määratud raskusastmega
4. Eemaldamise protsess peab olema kiire

Järgnevalt tuleb luua algoritm, mis lahendab ja kontrollib mängu väärtuseid. Selleks otsustas autor kasutada lihtsat tagurdamise algoritmi. Tegemist on ühe kõige primitiivsema ja robustsema lahdendusstrateegiaga, kus kasutatakse jõudu probleemi lahendamiseks. Lahendusmeetod kasutab rekursiivset meetodit sudoku lahendi leidmiseks. Algoritmi realiseerimiseks loome järgnevad funktsioonid:

- Funktsioon, mis kontrollib kas ruudustik muutub ebaturvaliseks
- Rekursiivne funktsioon, mis võtab sisse ruudustiku
- Funktsioon, mis leiab määratama ruudu ja sisestab numbri ühest üheksani
- Funktsioon, mis kontrollib kas ruudustikus on veel määratama ruute

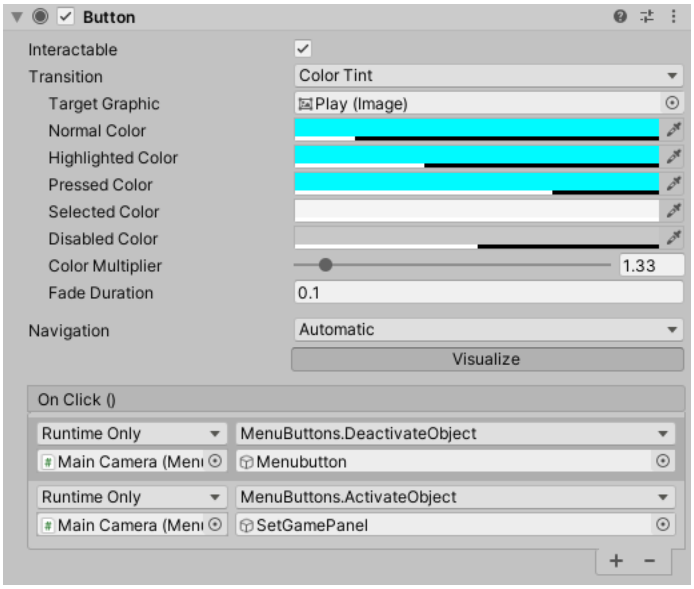
Autor otsustas kasutada tagurdamise algoritmi tema lihtsuse pärast. Autor leiab, et valitud mikrokontroller on piisavalt võimas, et kasutada lihtsamat algoritmi teades, et tegemist ei ole kõige efektiivsema lahendusmeetodiga. Viimaks toob autor esile mängu genereerimise vooskeemi:




Joonis 14 Mängu genereerimise vooskeem

## 4.3 Graafiline kasutajaliides

Liidese realiseerimiseks kasutab autor unity poolt pakutavat keskkonda koos abistavate skriptidega. Unity annab meile võimaluse luua programmeeritavaid nuppe, mille kaudu autor loob mängule kasutajaliidese.



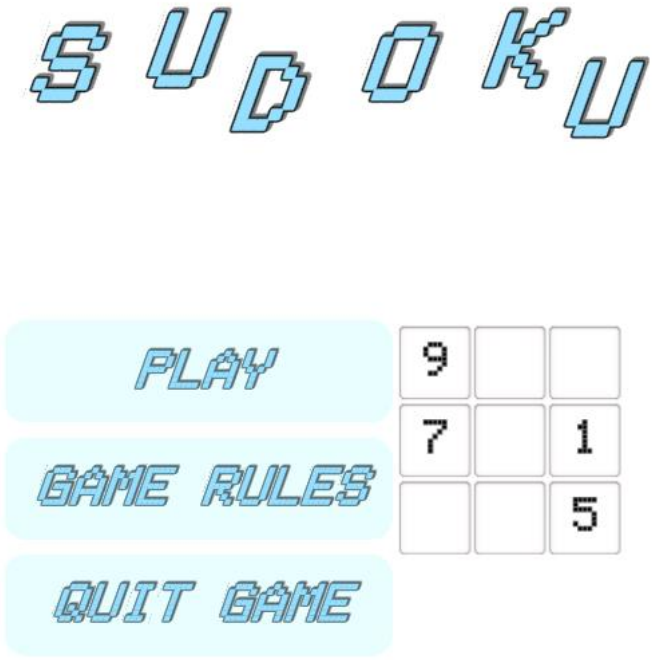

Joonis 15 Nuppu loomise menüü

Liidese disainimisel kasutas autor järgnevaid põhimõtteid:

1. Kasutajaliides peab olema lihtne.
2. Kasutajaliides peab kasutama konstantseid elemente
3. Kasutajaliidese elementidel peab olema eesmärk
4. Kasutajaliides peab kasutama ühtset värvi ja tekstuuri.

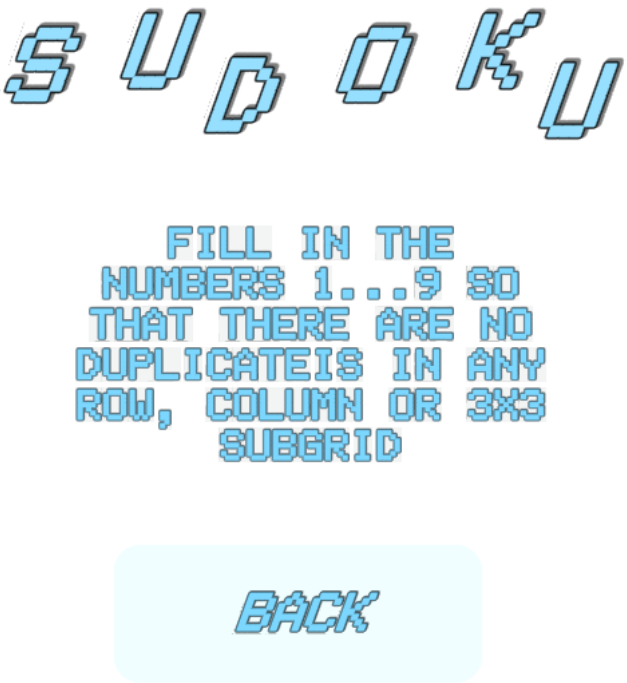
Tulenevalt etteantud tingimustest otsustas autor kasutada läbivalt helesinist värvi ja pikseleeritud tekstuuri. Kasutatavad nupud on animeeritud ja nuppude vajutamisel muutub nupu värv, eesmärgiga tõsta esile inimese poolt valitud tegevus. Autor pidas oluliseks liidese lihtsust ning leiab, et töös kasutatavad nupud täidavad eesmärki ja on

ühtselt arusaadavad. Samuti leiab autor, et liides peab olema kõigile inglise keelt kõnelevatele isikutele üheselt arusaadav.




Joonis 17 Graafilise kasutajaliidese peamenüü


Joonis 16 Mängu keerukuse menüü



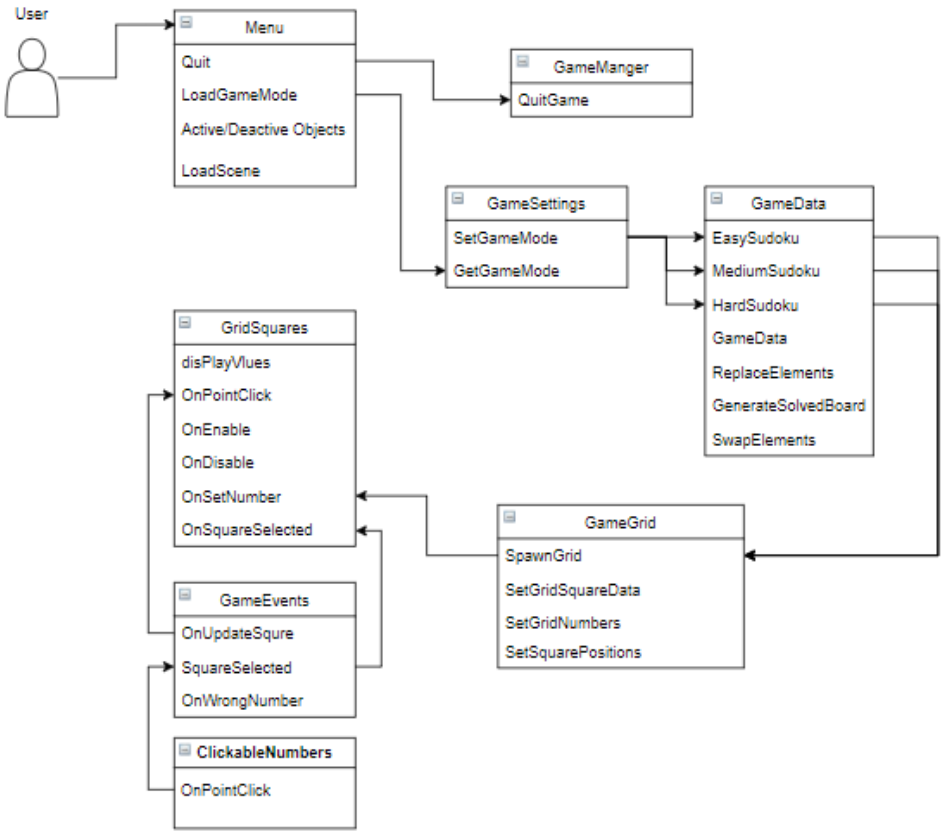

Joonis 19 Mängu reeglid


Joonis 18 Mängu vaade

## 4.4 Tarkvaraline struktuur

Programmi arhitektuur põhineb unity tarkvaraarendusmustril. Unity tarkvaraarenduses luuakse iseseisvaid skripte, mille abil seotakse skriptid mängu elementidega. Sellise arendusmeetodi kasutamise eeliseks on see, et komponendid on üksteistest sõltumatud ning seetõttu saab neid eraldi arendada.

Programm hoiab mälus erinevaid muutujaid. Kõik muutujad hoitakse mälus programmi sulgemiseni. Rakenduse avades luuakse uued väljad, kus hoitakse andmeid. Uue mängu genereerimisel on vanad väärtused ülekirjutatud, mille tulemusena ei ole võimalik taastada varem genereeritud mängu väärtuseid. Ülekirjutamine aitab hoida mikrokontrolleri mälu kontrollitud. Selle tulemusena mikrokontrolleri mälu ei täitu kunagi täies mahus, mis aitab hoida programmi töökindlana. Järgnevalt esitab autor illustreeriva pildi projekti struktuurist:




Joonis 20 Tarkvaraline struktuur

## 4.5 Edasiarenduse võimalused

Töös valminud rakendus täidab hetkel ainult sudoku kõige olulisemaid baasfunktsioone. Vaadeldes varem loodud sudoku rakendusi võiks lisada masinale vihjete andmise. Vihjed teevad mängu kasutajasõbralikumaks ja teeb mängu õppimise kergemaks. Samuti peaks programmile lisama võimaluse salvestada juba tehtud mänge ja neid uuesti laadida. See annaks kasutajale võimaluse mängu jätkata ja neid hiljem uuesti lahendada. Mängule võiks ka luua võimaluse saata genereeritud mänge teistele mängijatele läbi generatsiooni koodi või mõne muu lahendusega. Samuti võiks luua masinale erinevaid lahendusmeetodeid ja generatsiooni võimalusi. Lisades programmile erinevaid generatsiooni meetodeid on võimalik muuta mäng keerulisemaks. Loodes erinevaid generatsiooni meetmeid annab ka võimaluse genereerida unikaalseid mänge. Viimaks leiab autor, et mängule oleks mõistlik lisada rohkem visuaale, mis aitaks kasutajal kiiremini tuvastada ruute. Valitud ruudu visualiseerimine aitab kasutajal kergemini tuvastada puuduvat väärtust.

# 5 Testimine ja tulemused

Antud peatükis kirjeldatakse tarkvaralist testimisprotsessi. Testide eesmärk oli kontrollida töökindlust. Lisaks tutvustab autor lugejale testide tulemusi.

## 5.1 Algoritmi testimine

Mänguvälja genereerimiseks ja lahendamiseks kasutab autor kolmandat tarkvara[9]. Rakendus võimaldab sisestada autori poolt genereeritud mängu ja see lahendada. Autor leiab, et genereerimise testimine on üks tähtsamaid tegevusi, kuna genereerimine määrab mängu keerukuse. Samuti kontrollib genereerimise testimine mängu valiidsust. Algoritmi testimisel kontrollitakse järgnevaid tegevusi:

- Mänguväljalt väärtuste eemaldamine
- Mänguvälja genereerimine
- Sudoku lahendamine

## 5.2 Testimine mikrokontrolleril

Mikrokontrolleril testimise eesmärk oli anda autorile parema arusaam mängu elementide proportsioonist ja kasutamise võimalustest. Testimise käigus avastati, et elementide mõõtkava ei sõltu ekraani suurusest ega implemeeritud koodist. Lisaks kontrolliti, kuidas mikrokontroller käsitleb erinevad operatsioonisüsteeme. Mikrokontrolleri peal sai testitud linuxi ja androidil põhinevad operatsioonisüsteeme. Tulenevalt operatsioonisüsteemist pidi ka kontrollima, kuidas need operatsioonisüsteemid käsitlevad unity poolt väljastatud faili tüüpe. Testimisele kuulusid .apk ja .exe failid.

## 5.3 Testide tulemused

Testimiste tulemusena otsustas autor kasutada androidil põhinevat operatsioonisüsteemi, kuna linuxil põhinevad süsteemid muutsid mängu aeglasemaks. Samuti oli failide ehitamine linuxi süsteemidele keerukam, kuna ei võimalda sisestada programmi otse läbi unity keskkonna. Mänguelementide propotsiooni parandamiseks otsustas autor kasutada kindlate suurustega dünaamilisi elemente. Selle tulemusena selgus, et kui soovitakse kasutada rakendust tesitel platvormidel, tuleb käsitsi muuta elementide mõõtkava. Autori poolt läbiviidud testid mängu genereerimiseks olid läbivalt valiidsed, ehk loodud mängudel oli vaid üks lahend.

# 6 Kokkuvõte

Töö eesmärk oli realiseerida mikrokontrolleril põhinev sudoku-masin. Eesmärgi täitmiseks realiseeriti mikrokontrollerilie, Raspberry Pi Model B Plus, puutetundlik ekraan. Samuti, loodi mängu genereerimise algoritm ja graafilise kasutajaliidesega kasutades C# programmeerimiskeelt.

Mängu genereerimiseks loodi 9x9 ruudustik, mis täideti väärtustega ühest üheksani. Hiljem eemaldati ruutudest väärtuseid ja seejärel rakendati tagurdamise algoritmi, mille abil mäng lahendada. Generatsiooni käigus kasutati kõiki võimalusi, et tagada mängu töökindlust.

Graafiline kasutajaliides loodi, et tehtud rakendust kasutada. Liidesel on funktsionaalsust, mille abil saab valida mängu keerukust. Kasutajaliidese disainimisel jälgiti kindlaid reegleid, mille tulemusena on kasutajaliides ühtse disainiga.

Töö käigus õppis autor tarkvara arendust unity platvormil kasutades agiilset tarkvaraarendus meetodit, et programmi arendus oleks võimalikult sujuv ning arusaadav. Samuti õppis autor sudoku lahendusmeetodite rakendamist ning realiseerimist algoritmiliselt.

# Kasutatud kirjandus

[1] Raspberry Pi 3 Model plus andmeleht ja muu informatsioon [Võrgumaterjal] https://www.raspberrypi.org/products/raspberry-pi-3-model-b-plus/

[2] Raspberry Pi Puutetundliku ekraani andmeleht ja muu informatsioon[Võrgumaterjal] https://www.raspberrypi.org/products/raspberry-pi-touch-display/

[3] Unity dokumentatsioon. [Võrgumaterjal] https://docs.unity3d.com/ScriptReference/

[4] Androidi põhine operatsioonisüsteem[Võrgumaterjal]https://lineageos.org/

[5] Operatsioonisüsteemi kandmine mikrokontrollerile kasutatud tarkvara [Võrgumaterjal] https://sourceforge.net/projects/etcher.mirror/

[6] Arduino Uno Rev3 andmeleht ja muu informatsioon [Võrgumaterjal]https://store.arduino.cc/arduino-uno-rev3

[7] Arduino Mega HMI Touch Screen andmeleht ja muu informatsioon [Võrgumaterjal] https://www.electroniclinic.com/arduino-mega-hmi-touch-screen/

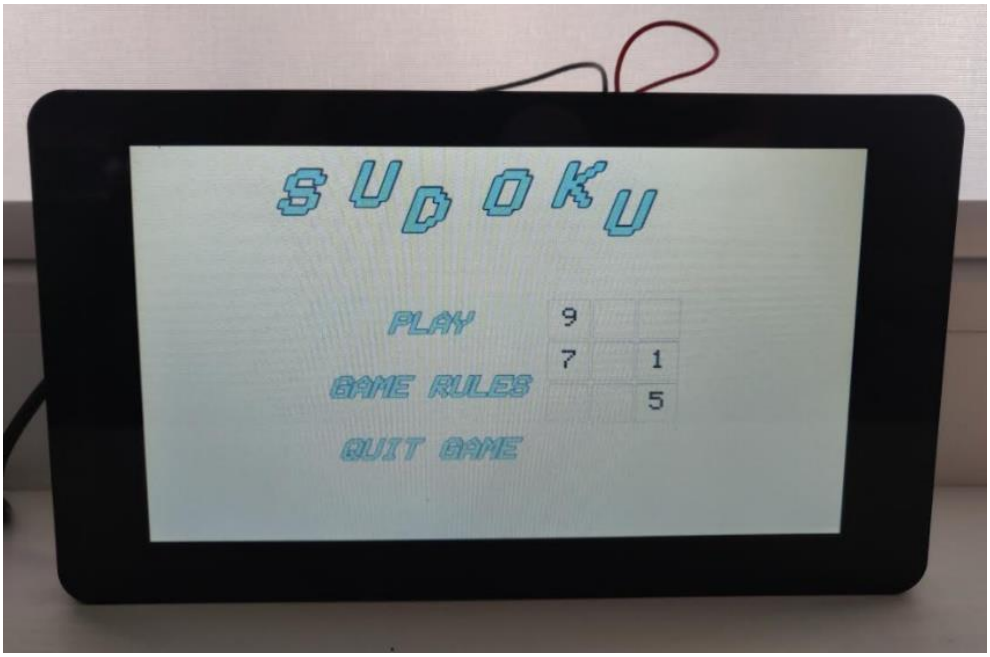
[8] Gary McGuire, Bastian Tugemann, Gilles Civario, „There is no 16-Clue Sudoku: Solving the Sudoku Minimum Number of Clues Problem via Hitting Set Enumeration," [Võrgumaterjal]https://arxiv.org/pdf/1201.0749.pdf

[9] Donald Knuth's Dancing Links algorithm to solve several Sudoku implementations [Võrgumaterjal] https://sudokuspoiler.azurewebsites.net/

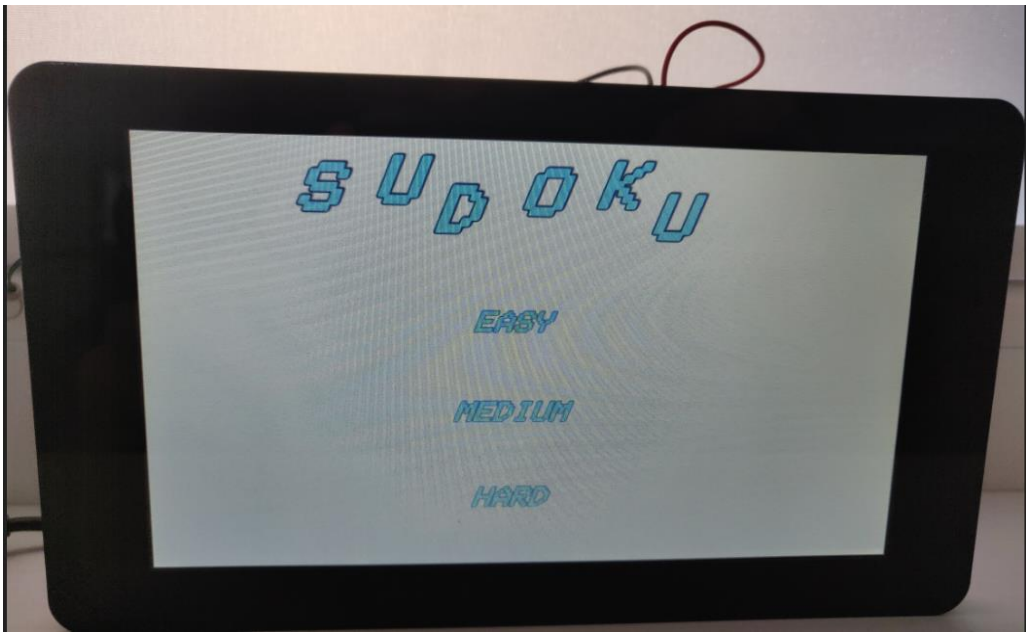
[10] Hodoku, a Sudoku generator/solver/trainer/analyzer [Võrgumaterjal] http://hodoku.sourceforge.net/en/index.php

[11] Sudoku mängu generaator ja lahendusmeetodid[Võrgumaterjal] https://sudoku-solutions.com/

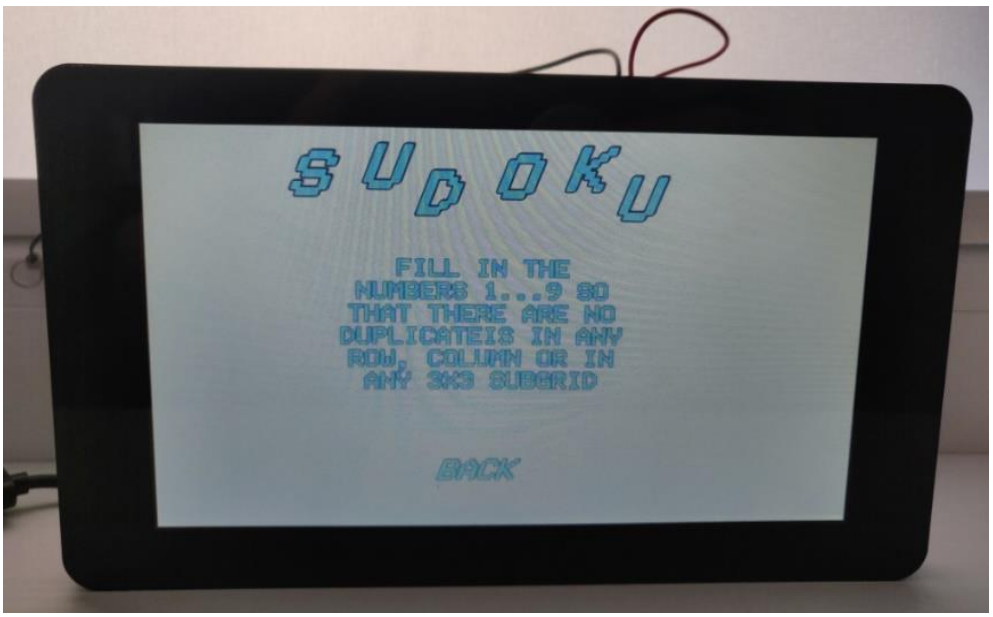
# Lisa 1 - Projekti väljundid




Joonis 21 Peamenüü mikrokontrolleril




Joonis 22 Keerukuse menüü mikrokontrolleril




Joonis 24 Mängu reeglite tutvustus mikrokontrolleril


Joonis 23 Mängu vaade mikrokontrolleril