TALLINNA TEHNIKAÜLIKOOL
Infotehnoloogia teaduskond
Tarkvarateaduse instituut

IE40LT

Elena Borissov 155175IAPB

iOS RAKENDUS NEUROMUSKULAARSETE HAIGUSTEGA PATSIENTIDE SEISUNDI JÄLGIMISEKS

Bakalaureusetöö

Juhendaja: Alar Kuusik
PhD
(tehnikateadused)
Kaasjuhendaja Triin Kask
MSc (siirdemeditsiin)

Tallinn 2019

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: Elena Borissov

23.05.2019

Annotatsioon

Käesoleva lõputöö eesmärgiks on luua iOS mobiiliseadme rakendus neuromuskulaarse haigusega patsientide kehalise suutlikkuse jälgimiseks kodustes tingimustes. Erinevalt varasemast lahendusest kasutab antud rakendus telefonisiseseid sensoreid, milleks on aktselerомeeter ja güroskoopsensor. Mõõtetulemused arvutatakse ühendatud serveri asemel otse mobiilirakenduses.

Antud töös kirjeldatakse telefonisiseste sensorite kasutamist React Native'iga loodud rakenduses, milleks töö autor kirjutas põliskomponendi. Mõõtmisalgoritmi kirjutamisel on arvestatud sensorite ebaühtlase võendamist.

Töö tulemuseks valmis iOS rakendus, mis mõõdab telefonisiseste sensoritega patsientide füsioloogilist võimekust, arvutab võrreldavad testitulemused ja hoiustab kõik mõõtmistulemused.

Lõputöö on kirjutatud eesti keeles ning sisaldab teksti 28 leheküljel, 5 peatükki, 15 joonist.

Abstract

iOS Application for Monitoring the Condition of Patients with Neuromuscular Diseases

The aim of this thesis is to create an iOS mobile application for home monitoring of physical performance of patients with neuromuscular disease. Unlike previous solutions offered, this application uses internal mobile phone sensors, which are an accelerometer and a gyroscope sensor. In addition, the calculation of test results is provided from a standalone server to a mobile application.

The work given describes the use of internal mobile phone sensors in the application created by React Native, a native component of which was written by the author. In addition, the algorithm for obtaining test results in case of irregular sampling of sensors has been described.

The result of the work is the ready-made iOS application that measures with internal mobile phone sensors the physiological capability of patients, calculates the comparable results for tests, and retains the information about the previous results of the measurement.

The thesis is in Estonian and contains 28 pages of text, 5 chapters, 15 figures.

Lühendite ja mõistete sõnastik

Android	Google poolt loodud mobiilne operatsioonisüsteem
API	Application Programming Interface ehk rakenduse programmeerimisliides
Apple	Riist- ja tarkvara arendav ning tootev ettevõte
Bluetooth	Standard seadmete vahelise traadita side jaoks
C	Üldkasutatav programmeerimiskeel
C++	Objektorienteeritud programmeerimiskeel
Fitbit	Ettevõte mis toodab inimeste füüsilist aktiivsust jälgivaid seadmeid
Header	Põliskomponenti saadetavate andmete päisesse lisatav informatsioon
HTML	HyperText Markup Language veebilehtede programmeerimiskeel
IDE	Integrated development environment ehk keskkond tarkvara arendamiseks
iOS	Apple mobiiliseadmete operatsioonisüsteem
iPhone	Apple toodetud nutitelefon
Java	Üldkasutatav programmeerimiskeel
JavaScript	Peamiselt veebilehtede loomiseks kasutatav programmeerimiskeel
kHz	Herts – perioodilise sageduse mõõteühik
Linux	Operatsioonisüsteem
Low Energy	Madal energiakulu
Npm	Node Package Manager ehk pakkihaldusmehhanism
Objective-C	Programmeerimiskeel
Popup	Hüpikmenüü
Põliskomponent	Platvormi spetsiifiline komponent
Range of Motion	Liikumise ulatus
React Native	Facebooki poolt loodud raamistik mobiilirakenduste loomiseks
Swift	Programmeerimiskeel
tag	Elementide esitusviis HTML-is

WebStorm	IDE veebirakenduste arendamiseks
XCode	Apple poolt loodud IDE

Sisukord

Jooniste loetelu

1 Sissejuhatus

1.1 Taust

Kehalise aktiivsuse sensoreid (näiteks sammulugejaid) on juba mõnda aega kasutatud inimeste tervise jälgimiseks. Algselt kasutasid sammulugejad mehaanilist liikumisandurit ja sammude loendit, aga tehnoloogia arenguga on meetodid täiustunud. Tänapäeva sammulugejad baseeruvad mikromehaanilistel aktseleromeetritel. Esimesena võttis selle kasutusele Fitbit. Pikaajalise testimise käigus on välja töötatud algoritmid, mis suudavad eristada kõndimise muudest liikumisviisidest [1]. Samasuguseid sensoreid saab lisaks sammude lugemisele kasutada ka muu kehalise aktiivsuse mõõtmiseks ja jälgimiseks.

Liikumisandurid on muutunud niivõrd tavaliseks, et need on juba igal inimesel nutitelefonis olemas. Mobiiltelefonid sisaldasid sensoreid juba varem, kuid 2013 aastal välja tulnud Androidi versioon 4.4 muutis need sensorid vähem energiat tarbivaks, mis võimaldas lisada telefoni püsiva sammulugeja ja -detektori [2], [3].

Ka iPhone andurid ei olnud koheselt püsivas kasutuses. Kui telefoni liikumist sai jälgida iOS 4.0-st, siis sammulugejad tulid telefonidesse koos M7 protsessoriga, mis on kasutusel alates 2013 aastal välja tulnud iPhone 5S-ist [4], [5].

1.2 Probleem

Neuroloogiliste haigustega patsientidel puuduvad mugavad kaasaegsed tehnoloogilised lahendused nende tervisliku seisundi muutuste jälgimiseks. Hulgiskleroosi (*Sclerosis multiplex*), nagu ka teiste neuroloogiliste haiguste mõju patsiendile, ilmneb liikumisvõime muutuses. Tavaliselt hinnatakse patsiendi seisundit visuaalse vaatlusega, näiteks Rombergi ja Range-of-Motion teste kasutades [6]. Kaasaegsed nutitelefonid sisaldavad liikumisandureid, mis võimaldavad neuroloogiliste haigustega patsientidele iseloomulikke kehalisi muutusi tuvastada oluliselt täpsemalt kui inimsilm [7].

1.3 Eesmärk

Projekti “Home monitoring system for neurodegenerative disease patients” raames loodi eraldiseisvast liikumisandurist ja Androidi rakendusest koosnev lahendus patsiendi kehalise seisundi hindamiseks kodus [8].

Käesoleva lõputöö eesmärk on valmistada Androidirakendusest iOS versioon, mis kasutab motoorika mõõtmiseks telefoni siseseid sensoreid ja lisab rakendusele funktsionaalsust. Valminud rakendus võimaldab patsiendi liikumisvõime muutusi jälgida ilma väliseid sensoreid kasutamata ja anda kasutajale kohest tagasisidet seisundi muutuste kohta.

1.4 Ülevaade tööst

Antud töö annab üldise ülevaate patsientide jälgimise süsteemist. Kolmandas peatükis kirjeldatakse iOS operatsioonisüsteemil töötavate seadmete liikumissensorite kasutamist React Native-is kirjutatud rakenduses. Neljandas peatükis selgitatakse, kuidas sensorite andmetest saadakse testide tulemused, mis on lihtsalt võrreldavad patsiendi eelmiste tulemustega.

2 Rakenduse üldine kirjeldus

Antud peatükk annab ülevaate olemasolevast lahendusest, mis tehti projekti “Home monitoring system for neurodegenerative disease patients” raames [8]. Kirjeldatud on ka antud lõputöö ülesanne ja kasutatud tehnoloogiad käesoleva süsteemi täiustamiseks.

2.1 Olemasolev lahendus

Projekti “Home monitoring system for neurodegenerative disease patients” raames loodi lahendus kehale kinnitatavast sensorist, Androidi rakendusest ja serveritarkvarast, et jälgida neuroloogiliste haigustega patsiente kodus ja vähendada arstivisiite [8]. Hetkel on käimas kliinilised katsed Tallinnas Astra Team Clinic [9].

Väline sensor, kujutatud joonisel 1, sisaldab endas aktseleromeetrit ja güroskoopsensorit. Sensorit on võimalik sisse ja välja lülitada, samuti annab sensor teavet aku tlaetuse tasemest. Sensorit saab kontrollida ja andmeid edastada Bluetooth Low Energy ühenduse abil. Sensori kasutamiseks on varasemalt loodud Androidi rakendus, mille kaudu saab sooritada viis erinevat testi: Range of Motion, tasakaalutest ehk Rombergi test, hüppetest, nine-hole-peg test. Peale selle saab patsient edastada hinnanguid oma enesetunde kohta. Rakendus saadab sensorist tulnud info serverisse.

Joonis 1. Reiele kinnitatud eraldiseisev sensor [8].

Serverist töödeldakse rakendusest tulnud andmeid. Testide tulemused salvestatakse andmebaasi ja tehakse veebirakenduse kaudu arstidele kättesaadavaks. Pärast testi tulemuste väljaarvutamist saadetakse tulemus ka kasutajale mobiilirakendusse. Eelmiste testide tulemused on nähtavad ainult veebirakenduses ja patsiendid tulemuste ajalugu ei näe.

2.2 Lõputöö ülesanne

Antud lõputöö ülesanne on teha alternatiivne rakendus, mis on mõeldud iOS operatsioonisüsteemile. Antud rakendus kasutab väliste sensorite asemel juba mobiiliseadmesse sisseehitatud sensoreid. On tehtud kindlaks, et telefonide sensorid on piisavalt täpselt, seega kaob vajadus kasutada eraldiseisvaid sensoreid [7].

Teine ülesanne on serveris tehtud päringu asemel teostada mõõetulemuste arvutamine otse mobiilirakenduses. Selleks tuleb varem C++ programmeerimiskeeles lahendatud arvutused implementeerida JavaScriptis ja muuta algoritme. Varem kasutatud sensorid genereerisid andmeid perioodiliselt, püsiva intervalliga, aga telefoni tarkvara käsib sensoril parameetreid mõõta kutsumise peale, seega toimub nn ebaühtlane võendamine, mis muudab asendi ja kiiruse täpse mõõtmise keerukamaks.

Kuna varasem rakendus on tehtud Javas Androidi operatsioonisüsteemile, tuleb antud töös alustada implementeerimist algusest. Et vältida tulevikus sarnaseid probleeme on töö jooksul valmiv rakendus implementeeritud React Native abiga.

2.3 Kasutatud tehnoloogiad

Antud töös on kasutatud peamiselt Javascripti koos React Native raamistikuga. Kuna sensorite kasutamiseks oli vaja luua mõned põliskomponendid, sai töös kasutatud ka iOS süsteemile omaseid keeli ja arendusvahendeid nagu Swift ja XCode. Antud peatükk tutvustab neid lähemalt.

2.3.1 Raamistik

React Native on JavaScripti raamistik mobiilirakenduste kirjutamiseks, mis võimaldab ehitada nii iOS kui ka Androidi põhiseid rakendusi. React Native ei kata kõiki

operatsioonisüsteemide võimalusi, aga vajadusel on võimalik lisada põliskomponente, realiseerides need operatsioonisüsteemile sobivas keeles [10].

Rakendus ehitatakse üles samadele komponentidele, mis on kasutusel platvormile omases keeles, aga React Native genereerib komponendid vastavalt operatsioonisüsteemile. Komponente saab ühendada sarnaselt HTML-ile, kus esimese komponendi tag-ide vahele lisatakse järgmine komponent [10].

2.3.2 Keeled

JavaScript on peamiselt kasutusel veebilehekülgede loomiseks, aga sel on ka palju teisi kasutusalasid. Kuna React Native on JavaScripti raamistik, siis oli see antud töös peamiselt kasutatav arenduskeel. JavaScriptis on kirjutatud nii kasutajaliides kui ka äriloogika. Antud keelt on lihtne laiendada npm (Node Package Manager) pakihalduriga, mis võimaldab taaskasutada juba varem teiste arendajate poolt loodud mooduleid [11].

Swift on programmeerimiskeel, mis on loodud kasutamiseks nii veebi- ja mobiilirakendustes kui ka serverites ning pilveteenustes. Linuxiga ühilduv Swift on hetkel kasutusel kõigis Apple platvormides [12]. Swifti on antud töös kasutatud iOS seadmete sensorite teabe kogumiseks, sest ReactNative ei laienda antud funktsionaalsust iOS APIst (Application Programming Interface).

Objective-C on loodud C täiustatud versiooniks, mis võimaldaks kasutada objektorienteeritud programmeerimise põhimõtteid. [13]. Kuni Swifti tulekuni oli Objective-C peamine Apple arenduskeel ja seda kasutatakse siiani palju [14]. Antud töös on Objective-C-d kasutatud Swifti ja JavaScripti ühendamiseks.

2.3.3 Tööriistad

WebStorm on JetBrainsi poolt veebiarenduseks loodud IDE, mis toetab ka JavaScripti. Lisaks toetab see kõiki JavaScripti levinumaid raamistikke. Antud IDE pakub arendamiseks mugavaid lisavõimalusi, nagu koodi soovitused, refaktoreerimine ja koodi lõpetamine[15]. Töö autor valis selle peamiseks arenduskeskkonnaks, sest tal oli varasem kogemus JetBrainsi IDE-dega ja hindas ta selle kasutajamugavust järgnevalt tutvustatavast XCodest paremaks.

XCode on OS operatsioonisüsteemis kasutatav IDE. XCode-iga saab kirjutada rakendusi kõikidele Apple platvormidele. XCode sisaldab simulaatoreid kõigile iOS mudelitele [16]. Antud töös kasutati seda rakenduse jooksutamiseks seadmes ning Swifti ja Objective-C-ga põlismooduli loomiseks.

2.3.4 Kasutatud moodulid

React Native Sound on React Native lisamoodul, mis võimaldab mängida helifaile nii Androidi, Windowsi kui ka iOS keskkonnas [17]. Antud töös kasutati seda tasakaalu testis helisignaalide mängimiseks.

React Native Stopwatch Timer on moodul stopperi ja taimeri jaoks. Antud moodul annab ette lihtsa stopperi, mille kujundust saab sätetega muuta [18]. Käesolevas töös kasutati seda Nine Hole Peg testi jaoks.

React Native Actionsheet on moodul popup valikute jaoks [19]. Antud töös kasutati seda tagasisideküsimustiku osas, kus osadele küsimustele on valikvastused ette antud.

Foundation on Swifti raamistik, mis sisaldab endas põhilisi elemente teksti hoiustamiseks ja töötlemiseks, kellaaja ja kuupäeva töötlemiseks ning sorteerimiseks ja filtreerimiseks [20].

Core Motion on Swifti raamistik, mille kaudu on võimalik ligi pääseda seadme keskkonda mõjutavatele sensoritele. Antud raamistik annab ligipääsu aktseleromeetrile, güroskoopsensorile, pedomeetrile, magnetomeetrile ja kõrgusele [4].

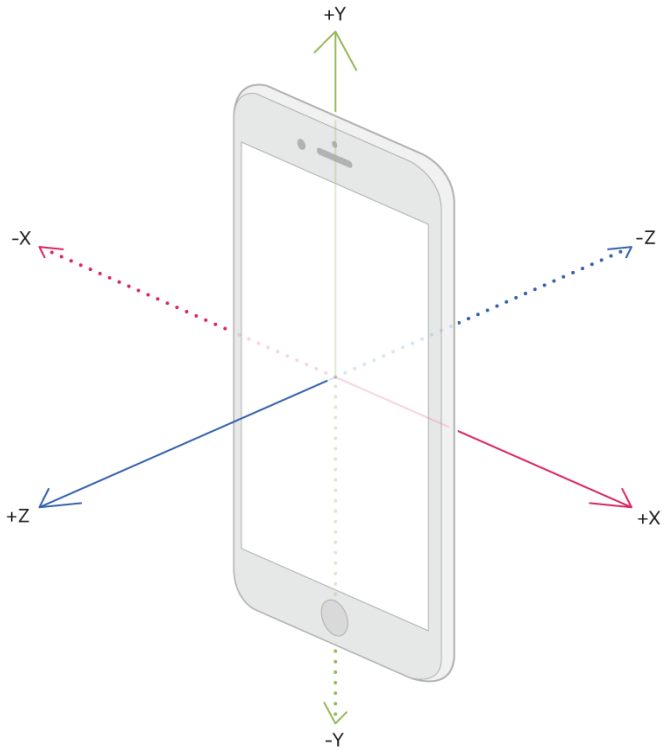
3 Nutitelefoni andurite kasutamine rakenduses

Antud peatükk selgitab, kuidas React Native'i rakendus saab kätte telefoni inertsiaalsensorite andmed. Esimene osa keskendub riistvara poolt mõõdetud andmete kättesaamisele iOS API kaudu. Teine osa selgitab kuidas React Native saab ühendada operatsioonisüsteemile spetsiifilise tarkvaraga.

3.1 Aktseleromeetri ja güroskoopsensori andmete kogumine

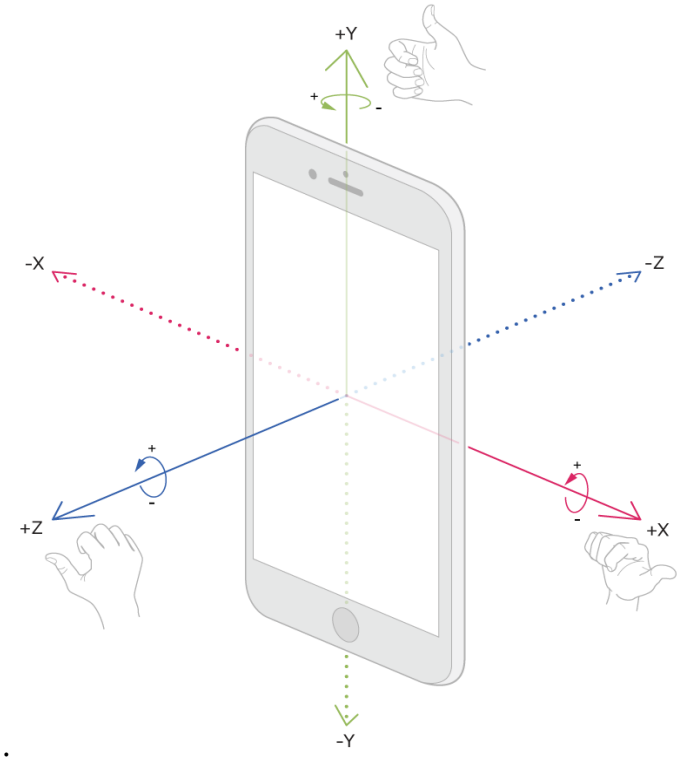
iOS seadmete anduritele saab ligipääsu Core Motion raamistiku kaudu. Liikumisandurite kohta saab infot nii algsel kujul kui ka juba eeltöödelduna, näiteks aktseleromeetri mõõtetulemusest on välja jäetud gravitatsioonist tulenev joonkiirendus [4].

Aktseleromeetrid mõõdavad joonkiirendust kolme telje suhtes, nagu näidatud joonisel 2. Andmeid esitatakse tüüpiliselt gravitatsioonilise kiirenduse sammuga, kus 1.0 väärtus tähistab kiirendust 9.8 meetrit sekundi ruudu kohta. Andmeid on võimalik küsida nii konkreetse aja kohta või paluda andmeid saata regulaarsete intervallide järel [21]. Käesoleva töö seisukohast on oluline, et mitte-reaalajaline operatsioonisüsteem ei ole siiski võimeline mõõteintervali piisava täpsusega (<1ms) konstantsena hoidma.



Joonis 2. Aktseleromeetri kolm telge [21].

Güroskoopsensor mõõdab nurkkiirust, millega seade pöörleb ümber oma telje. Enamus iOS seadmetel jälgitakse pöörlemist kolme telje suhtes, näidatud joonisel 3. Rotatsiooni mõõdetakse väärtuses radiaani sekundis. Mõõdetavad väärtused võivad olla positiivsed või negatiivsed, olenevalt seadme pöörlemise suunast [22].



Joonis 3. Güroskoopsensori kolm telge[22].

Raamistikust aktseleromeetri ja güroskoopsensori andmete küsimine toimub samal põhimõttel. Mõlema sensori andmete küsimine on toodud välja joonisel 4. Kõigepealt on vaja paluda raamistikul hakata andurite andmeid jälgima. Jälgimiseks peab paika panema, mis sagedusega andmeid uuendada. Võimalik maksimaalne sagedus sõltub konkreetsest sensorist ja seadme püsivarast (operatsioonisüsteemist). Inertsiaalsensorite maksimaalne mõõtesagedus on tavaliselt 1kHz, enamasti osutub piiravaks seadme püsivara. Kui andmete uuendamise sagedus osutub võimalikust suuremaks, kasutab raamistik suurimat võimalikku sagedust. Selleks et saada regulaarseid uuendusi anduri andmete kohta, peab kasutama taimerit, mis küsib raamistikult andmeid sama sagedusega, kui neid salvestatakse. Taimerit võib käivitada ka väiksema sagedusega, aga see raiskaks arvutusvõimsust, sest riistvara genereerib andmeid sagedamini, kui tegelikult vaja [20, 21].

```
@objc
func startSensors() {
      if self.motion.isAccelerometerAvailable &&
                  self.motion.isGyroAvailable {
            self.motion.accelerometerUpdateInterval =
                  1.0/60.0
            self.motion.gyroUpdateInterval = 1.0 / 60.0
            self.motion.startAccelerometerUpdates()
            self.motion.startGyroUpdates()
            DispatchQueue.main.async {
                  self.timer =
                  Timer.scheduledTimer(
                        withTimeInterval: (1.0/60.0),
                        repeats: true) {
                        [weak selfx] timer in
                  if let dataA =
                              self!.motion.accelerometerData {
                              if let dataG =
                                    self!.motion.gyroData {
                                    //Use data
                              }
                        }
                  }
            }
      }
}
```

Joonis 4. Aktseleromeetri ja Güroskoopsensori andmete kogumine Swiftis [21].

3.2 Swift ja React Native

React Native ei toeta kõiki platvormi API funktsionaalsusi. Selleks, et neid oleks võimalik ikkagi kasutada, saab neid funktsionaalsusi kasutada läbi platvormile omase keele [23]. iOS operatsioonisüsteem töötab Objective-C baasil, aga tuleb ka toime Swiftiga, mis on loodud spetsiaalselt C baasi keelte lihtsama alternatiivina [12].

Selleks, et React native klassid saaksid ligi Swifti koodile, peab need siduma Objective-C-s kirjutatud silla kaudu. Sild on Objective-C klass, mis implementeerib RCTBridgeModule protokolli, nagu näidatud joonisel 5. Swifti klassis on vaja kõigile

väljastpoolt klassi väljakutsutavatele meetoditele lisada @objc modifikaator. Lisaks on vaja lisada header fail, mis seoks omavahel Swifti ja Objective-C klassid. Näide header failist on joonisel 6.

```
#import <Foundation/Foundation.h>
#import<CoreMotion/CoreMotion.h>
#import "React/RCTBridgeModule.h"
@interface RCT_EXTERN_MODULE(Sensors, NSObject)
RCT_EXTERN_METHOD(startSensors)
RCT_EXTERN_METHOD(stopTimer)
RCT_EXTERN_METHOD(getData: (RCTResponseSenderBlock)callback)
@end
```

Joonis 5. Objective-C silla klassi sisu.

```
#import "React/RCTBridgeModule.h"
```

Joonis 6. Header faili sisu.

4 Kogutud andmete töötlemine

Valminud tarkvararakendus sisaldab kolme testi, mis kasutavad liikumisandureid. Hüppetest kasutab aktselerомeetri tulemusi, tasakaalu ja jala liikuvuse test kasutab güroskoopsensori andmeid. Konkreetsed testid on välja valitud neuroloogide ja füsioterapeutide poolt ning neid on kasutatud Lääne-Tallinna Keskhaigla, Astra Team Clinicu ja TalTechi uurimisprojektides [8].

4.1 Jala liikuvus - Range of Motion

Antud testi käigus mõõdetakse patsiendi puusaliigese (tehniliselt ka teiste liigeste) liikuvust kraadides ja liikumise maksimaalset nurkkiirust. Test sooritatakse toolil istudes, telefon kinnitatakse reiele, nagu näidatud joonisel 7, testi käigus tõstab patsient kolm korda põlve võimalikult kõrgele. Test sooritatakse mõlema jalaga eraldi. Ühtlase tulemuse saavutamiseks on tähtis hoida kõigi testide ajal sama asendit.

Joonis 7. Testi ajaks reiele kinnitatud telefon.

Esimene samm reie liikumisnurga leidmiseks on aru saada, mis hetkel hakata jala liikumist mõõtma. Güroskoop mõõdab nurkkiirust + või - märgiga, olenevalt kummas

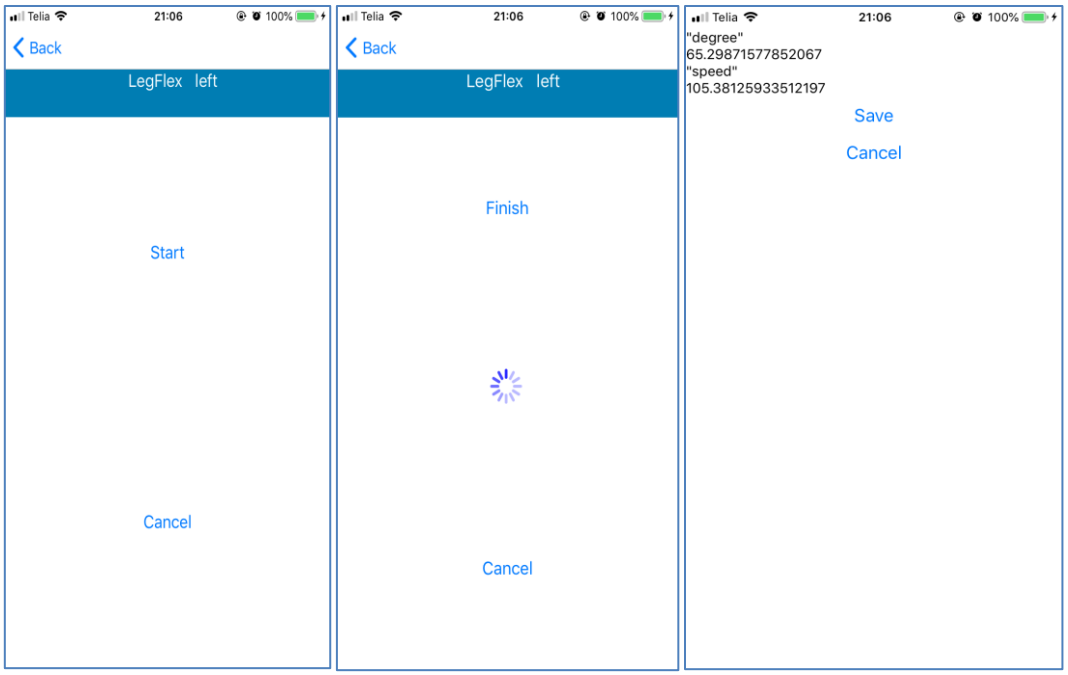
suunas sensor pöörleb. Kui jala liikumine muudab suunda, siis kiirus muutub kas positiivsest negatiivseks või vastupidi. Kuna sensorite müra võib ajutiselt anda veidraid tulemusi, siis ei arvestata suuna muutust, kui liikumise ulatus kahe suuna vahel on väiksem kui 1cm.

Selleks et leida jala tõusunurka, tuleb alustada läbitud nurgast kõigil kolmel teljel. Et leida kahe mõõdetud punkti vahel läbitud nurk, peab leidma keskmise kiiruse ja korrutama selle kulunud ajaga, nagu on näidatud joonisel 8. Kogu liikumise jooksul läbitud nurga leidmiseks peab liitma kokku kõigi mõõtepunktide vahel läbitud nurgad. Testi tulemuseks on suurim leitud nurk. Keskmise nurkkiiruse leidmiseks peab leitud nurga jagama kogu liikumisele kulunud ajaga.

```
var time = data[ix].time - data[i-1x].time;
var speedX =(data[i-1x].x + data[ix].x)/2.0;
var distanceX = speedX * time;
```

Joonis 8. Vahemaa leidmine kahe mõõtepunkti vahel x-teljel.

Autor katsetas testi enda peal ja see andis realistlikud, nurga visuaalse hindamise teel kontrollitud mõõtetulemused, mis on näidatud joonisel 9.



Joonis 9. Jala liikuvuse testi kulg realiseeritud rakenduses.

4.2 Hüppetest

Hüppetestiks kinnitatakse telefon pahkluu ümber, nagu näidatud joonisel 10. Patsient hüppab katse jooksul kolm korda ühel jalal. Katset korratakse mõlema jalaga. Katse jooksul leitakse õhus oldud aeg sekundites. Õhus püsitud aeg iseloomustab hüppekõrgust ja selle kaudu saab hinnata hüppaja sportlikku võimekust. Seetõttu on hüppetest laialdaselt kasutusel sportlaste testimisel, kuid võimaldab hinnata ka neuromuskulaarse haigusega patsientide seisundit.

Joonis 10. Testi ajaks pahkluu ümber kinnitatud telefon.

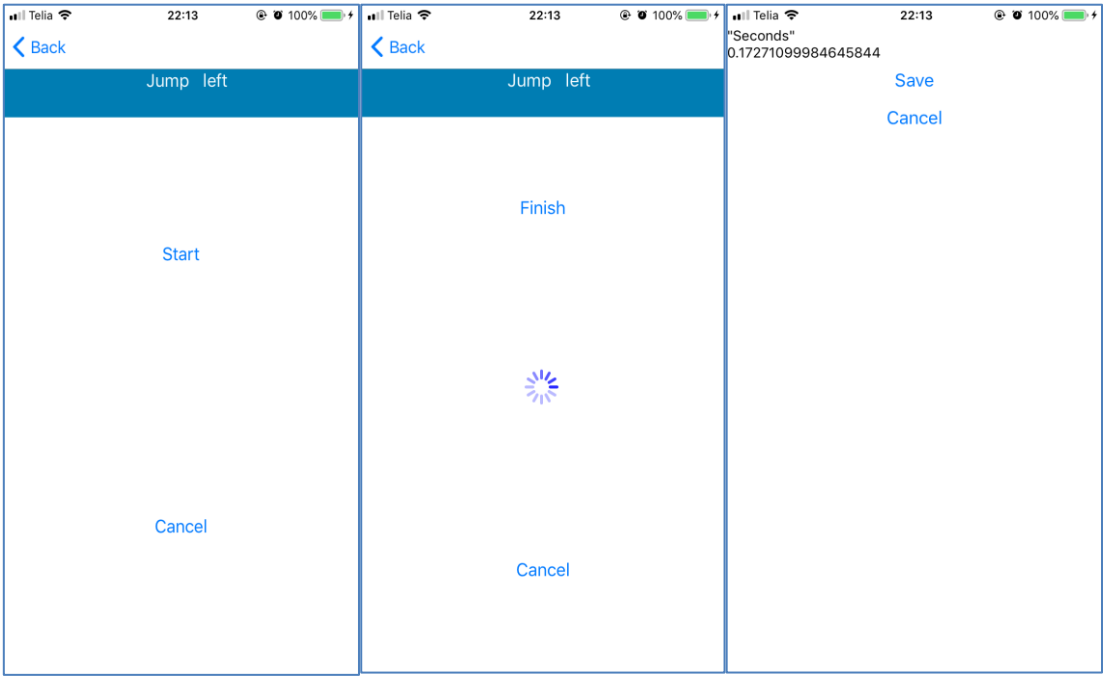
Kõige raskem osa hüppetestist on tuvastada hüppe alguse ja lõpu momenti. Kõigepealt peab selgitama välja, millise telje järgi hüpet vaadata. Selleks selgitatakse välja telg, mille suunas keskmine kiirendus on suurim. Seejärel hakatakse läbi käima valitud teljel testi jooksul salvestatud tulemusi. Esimene tingimus hüppe alguseks on, et kiirendus peab langema alla määratud piiri. Testis kasutatud numbrilised piirid on eksperimentaalselt

määratud teadusprojekti “Home monitoring system for neurodegenerative disease patients” raames [8]. Teine tingimus on, et enne hüppe algust peab olema vähemalt neli mõõtepunkti. Kolmas tingimus on, et korraga saab olla ainult üks alguspunkt. Kui viimasest alguspunkti leidmisest on vähe aega möödunud, arvestatakse see kokku eelmise alguspunktiga. Kui läbitud on 40 mõõtepunkti, loetakse see uueks hüppe alguseks ja vana unustatakse.

Hüppe lõppu otsitakse ainult siis, kui hüppe algus on juba leitud. Esimene tingimus on, et kiirendus peab tõusma üle määratud piiri. Järgmisena peab vähemalt neli järgnevat mõõtepunkti tulemust ületama veel kõrgema määratud piiri.

Hüppe alguse ja lõpu vaheline aeg ongi hüppe kestvus. Testi tulemusena salvestatakse parim tulemus kolmest.

Autor katsetas realiseeritud testi enda mobiiltelefoniga. Test andis realistlikud mõõtetulemused, mis on näidatud joonisel 11.

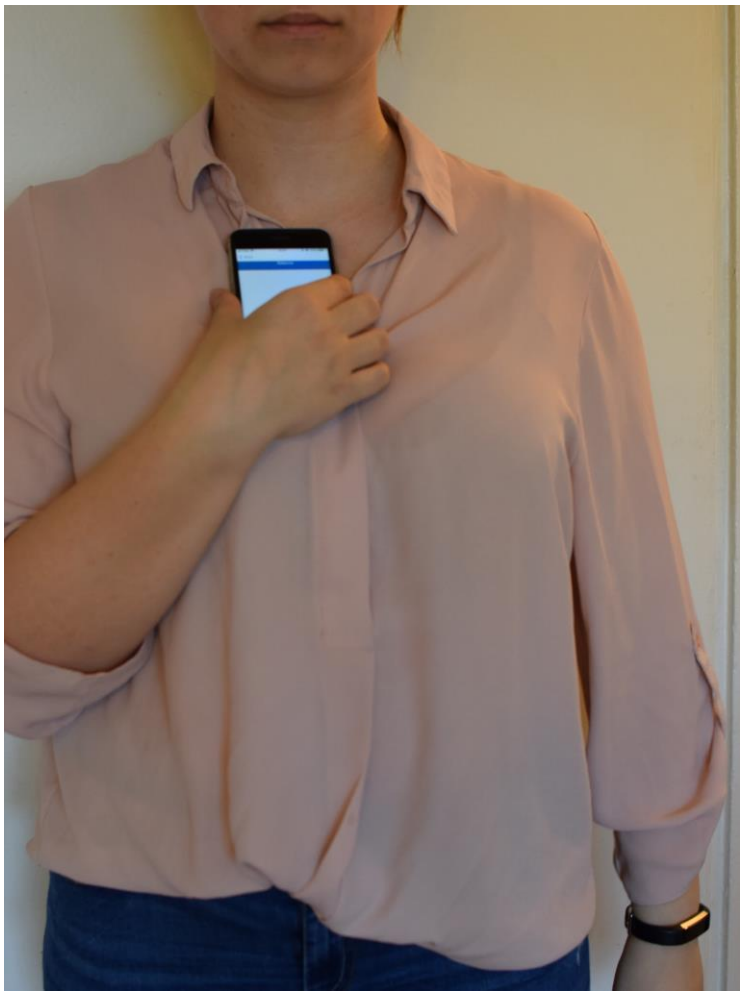


Joonis 11. Hüppetesti kulg realiseeritud rakenduses.

4.3 Tasakaal - Rombergi test

Rombergi test on lihtne katse tuvastamaks patsiendi tasakaaluhäireid. Test sooritatakse püsti seistes, käed külgedel, patsient peab seisma nii stabiilselt kui võimalik. Testi

alustatakse seistes avatud silmadega ja poole testi peal silmad suletakse [24]. Telefon sensoriga asub patsiendi rinna peal, nagu näidatud joonisel 12.



Joonis 12. Testi ajal rinnal asuv telefon.

Tasakaalutesti puhul ei ole vaja testi algust ja lõppu otsida signaalist, sest testi mõõtmine sooritatakse varem kindlaksmääratud ajavahemikus. Patsiendile annab testi kulgemisest märku helisignaal. Esimene signaal tähistab testi algust, teine helisignaal kõlab testi keskel kui katsealune peab silmad sulgema, kolmas signaal tähistab katse lõppu. Testi tulemus arvutatakse eraldi “silmad lahti” ja “silmad kinni” katseosa jaoks.

Katse tulemuse leidmiseks peab kõigepealt alustama iga güroskoopsensori telje keskmise väärtuse leidmisega. Kuna telefonisisene güroskoopsensor tagastab väärtuseid radiaani/sekundis, tuleb väärtuste keskmise leidmiseks korrutada näidud 57.2957795, et saada tulemus kraadi/sekundis. Lisaks peab tulemuse korrutama läbi 65.5-ga, et

saavutada sama lõpptulemus olemasolevas lahenduses kasutatavate sensoritetega, nagu näidatud joonisel 13.

```
//value = signal result
var i = 1;
var avgX = 0;
var avgY = 0;
var avgZ = 0;
for (i=1; i<value.length; i++) {
   var data = value[ix];
   avgX += data.xg * 57.2957795 * 65.5 ;
   avgY += data.yg * 57.2957795 * 65.5 ;
   avgZ += data.zg * 57.2957795 * 65.5 ;
}
avgX = avgX / value.length;
avgY = avgY / value.length;
avgZ = avgZ / value.length;
```

Joonis 13. Iga telje keskmise väärtuse leidmine.

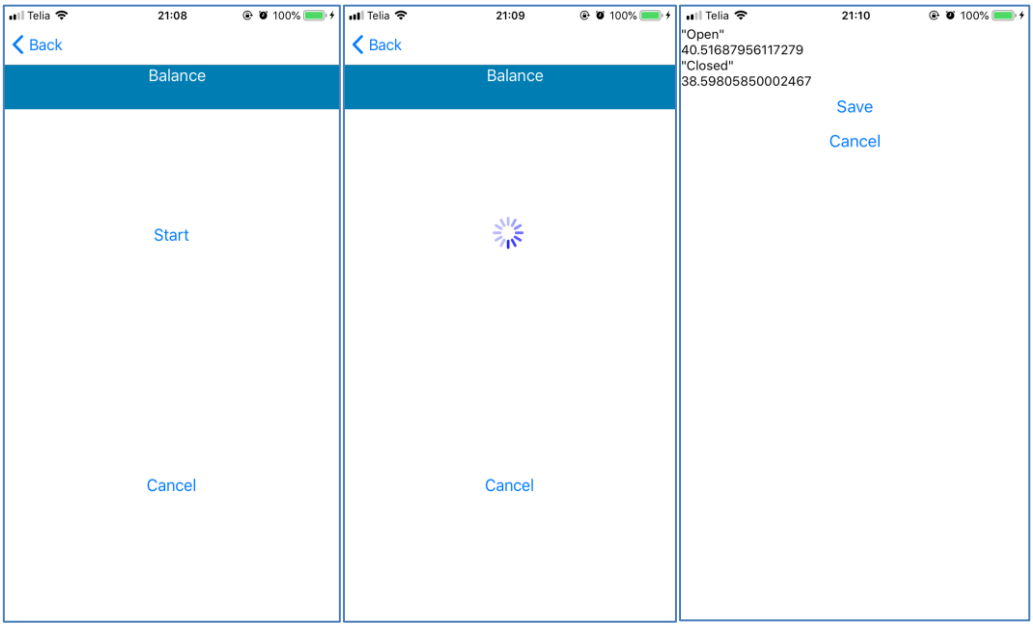
Testi tulemuse leidmiseks tuleb leida keskmine kõigi mõõtepunktide väärtuse ja telje keskmise väärtuse vahest, nagu näidatud joonisel 14. Suuna (sensori väärtuse märgi) mõju elimineerimiseks kasutatakse ruutkeskmiste arvutust. Lõpptulemus on kolme telje tulemuste keskmine.

```
var x = 0;
var y = 0;
var z = 0;
var j;
for (j=1; j<value.length; j++) {
   var data = value[jx];

   x += ((data.xg * 57.2957795 * 65.5) - avgX) *
      ((data.xg * 57.2957795 * 65.5) - avgX);
   y += ((data.yg * 57.2957795 * 65.5) - avgY) *
      ((data.yg * 57.2957795 * 65.5) - avgY);
   z += ((data.zg * 57.2957795 * 65.5) - avgZ) *
      ((data.zg * 57.2957795 * 65.5) - avgZ);
}
x = x / value.length;
y = y / value.length;
z = z / value.length;
var result = (Math.sqrt(x) + Math.sqrt(y) + Math.sqrt(z)) / 3;
```

Joonis 14. Tasakaalu testi tulemuse leidmine.

Realiseeritud tasakaalutesti katsetati iPhone 6s seadmel ja see andis õiged (referentssensoriga võrreldavad) mõõtetulemused, nagu näidatud joonisel 15. Sellest võib järeldada, et tarkvara töötab korrektselt.



Joonis 15. Tasakaalu testi kulg realiseeritud rakenduses.

5 Kokkuvõte

Käesoleva lõputöö eesmärgiks oli valmistada neuromuskulaarse haigusega patsientidele iOS rakendus motoorika jälgimiseks, mis kasutab liikumise mõõtmiseks telefoni siseseid sensoreid ja lisab senisele rakendusele funktsionaalsust.

Antud töö tulemusena valmis iOS rakendus, mis kasutab telefonisisest aktselerомeetrit ja güroskoopsensorit. Lisaks analüüsitakse Range of Motion, hüppetesti ja Rombergi testi puhul sensoritest tulnud info koheselt rakenduses. Patsiendil on võimalik ka vaadata enda eelmiste testide tulemusi. Sellega on töö eesmärk täidetud.

Töö autor kirjeldas töös täpsemalt, kuidas ta kasutas telefoni siseseid sensoreid väliste sensorite asemel, luues Swiftis põliskomponendi ja ühendades selle React Native'iga. Realiseeriti algoritm, mis võimaldab saavutada korrektsed mõõtetulemused mittereaalajalisest operatsioonisüsteemist tuleneva ebaühtlase andmete võendamise korral.

Antud tööd saaks edasi arendada ühendades selle praegu kasutusel oleva lahendusega. Samuti võiks juurde teha rakendusest Androidi versiooni. Kuna töös kasutati React Native'it oleks vaja juurde luua ainult Androidi põliskomponent sensorite lugemiseks. Lisaks võiks parandada kasutajaliidest nii kasutajamugavuse kui ka kujunduse poole pealt.

Kasutatud allikad

[1] R, MacMagnus, „Trackers“ David Bateman Ltd, 2014, pp 19-21 [Online] Available: https://www.amazon.com/Trackers-technology-helping-monitor-improve-ebook/dp/B00QUEZKAA.

[2] „KitKat overview“ [Online] Available: https://developer.android.com/about/versions/kitkat.html#44-sensors Accessed: 27.01.2019.

[3] “Sensors Overview“ [Online] Available: https://developer.android.com/guide/topics/sensors/sensors_overview Accessed: 27.01.2019.

[4] „Coremotion“ [Online] Available: https://developer.apple.com/documentation/coremotion Accessed: 22.03.2019.

[5] “Validity of the IPhone M7 Motion Co-Processor as a Pedometer for Able-Bodied Ambulation.” *Journal of Sports Sciences*, vol. 34, no. 23, 2016, pp. 2160–2164.

[6] A. Kuusik, K. Gross-Paju , H. Maamägi and E.Reilent, “Comparative study of four instrumented mobility analysis tests on neurological disease patients” presented at 2014 11th International Conference on Wearable and Implantable Body Sensor Networks Workshops, Jun. 16-19, 2014.

[7] T. Kask and A. Kuusik, “Performance comparison of smartphones and a wearable motion sensor for patient m-assessment” presented at 2018 16th Biennial Baltic Electronics Conference (BEC), Oct. 8-10, 2018.

[8] A. Kuusik, M. Alam, T. Kask, and K. Gross-Paju, “Wearable massessment system for neurological disease patients,” , IEEE 4th World Forum on Internet of Things (WF-IoT), Singapore, 2008, pp 201-206).

[9] [Online] Available: https://www.astrakliinik.ee/projektid/ Accessed: 23.04.2019.

[10] „Build native mobile apps using JavaScript and React“ [Online] Available: https://facebook.github.io/react-native/ Accessed: 14.04.2019.

[11] „What is JavaScript?“, 2019 [Online] Available: https://developer.mozilla.org/en-US/docs/Web/JavaScript/About_JavaScript#What_is_JavaScript Accessed: 05.05.2019.

[12] „About Swift“ [Online] Available: https://swift.org/about/ Accessed: 05.05.2019.

[13] „The Objective-C programming language“, 2013 [Online] Available: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ObjectiveC/Introduction/introObjectiveC.html Accessed: 05.05.2019.

[14] „About Objective-C“, 2013 [Online] Available: https://developer.apple.com/library/archive/documentation/Cocoa/Conceptual/ProgrammingWithObjectiveC/Introduction/Introduction.html Accessed: 05.05.2019.

[15] [Online] Available: https://www.jetbrains.com/webstorm/ Accessed: 05.05.2019.

[16] „Xcode 10“ [Online] Available: https://developer.apple.com/xcode/ Accessed: 05.05.2019.

[17] „React-native-sound“ [Online] Available: https://github.com/zmxv/react-native-sound Accessed: 06.05.2019.

[18] „React Native Stopwatch Timer“ [Online] Acailable: https://www.npmjs.com/package/react-native-stopwatch-timer Accessed: 06.05.2019.

[19] „React-native-actionsheet“ [Online] Available: https://www.npmjs.com/package/react-native-actionsheet Accessed: 06.05.2019.

[20] „Foundation“ [Online] Available: https://developer.apple.com/documentation/foundation Accessed: 06.05.2019.

[21] „Getting Raw Accelerometer Events“ [Online] Available: https://developer.apple.com/documentation/coremotion/getting_raw_accelerometer_events Accessed: 22.03.2019.

[22] „Getting Raw Gyroscope Events“ [Online] Available: https://developer.apple.com/documentation/coremotion/getting_raw_gyroscope_events Accessed: 22.03.2019.
[23] „Native Modules“ [Online] Available: https://facebook.github.io/react-native/docs/native-modules-ios Accessed: 22.03.2019.
[24] Khasnis A, Gokula R M., “Romberg's test,” J Postgrad Med 2003, pp. 49-169.