TAMPEREEN TEKNILLINEN YLIOPISTO

SAMPPA TOIVIAINEN

Inertia-anturihiiri mobiililaitteelle

Diplomityö

Tarkastajat: Prof. Jarmo Takala
TkL Helena Leppäkoski
Diplomityön aihe on hyväksytty tieto- ja sähkötekniikan tiedekuntaneuvoston kokouksessa 3.10.2012

TIIVISTELMÄ

TAMPEREEN TEKNILLINEN YLIOPISTO
Signaalinkäsittelyn ja tietoliikennetekniikan koulutusohjelma

TOIVIAINEN, SAMPPA: Inertia-anturihiiri mobiililaitteelle
Diplomityö, 42 sivua
Toukokuu 2013
Pääaine: Sulautetut järjestelmät
Tarkastajat: prof. Jarmo Takala ja TkL Helena Leppäkoski
Avainsanat: kiihtyvyysanturi, gyroskooppi, hiiri, iOS

Tässä diplomityössä esitellään miten voidaan toteuttaa mobiilisovellus, jota voidaan käyttää tietokoneen hiirenä. Hiiren kursorin liikuttamiseen käytetään mobiililaitteen sisältämiä inertia-antureita. Mobiililaite, jossa hiirisovellusta ajetaan, on mahdollista kytkeä tietokoneeseen langattomasti.

Käytetty mobiililaite tässä työssä on Applen iPhone 4, joka sisältää kolmiulotteiset kiihtyvyysanturin ja gyroskoopin. Mobiilisovelluksen lisäksi tarvitaan tietokonesovellus, joka tietokoneella suorittaa hiiritoiminnot. Tietokonesovelluksesta on toteutettu versiot sekä Windows- että OS X -käyttöjärjestelmille. Työssä käydään läpi näiden sovellusten rakenne ja mobiilisovelluksen käyttöliittymä. Laitteesta löytyviä Bluetooth- ja WiFi-ominaisuuksia käytetään hyödyksi langattoman yhteyden muodostamisessa. Mobiililaitteen ja tietokoneen välille muodostetaan näiden ominaisuuksien avulla verkkoyhteys, jonka yli tietokonesovellus ja mobiilisovellus kommunikoivat UDP:n päälle rakennetun protokollan avulla.

Lisäksi tehdään antureiden ja yhteysmenetelmien suorituskykyyn liittyviä mittauksia. Eri yhteysmenetelmien aiheuttamat viiveet mitataan. Sekä gyroskoopin että kiihtyvyysanturin näytearvojen laatua tarkastellaan. Lisäksi mitataan taajuus, jolla käytetty mobiililaite pystyy tuottamaan uusia näytearvoja.

ABSTRACT

TAMPERE UNIVERSITY OF TECHNOLOGY
Degree Program on Signal Processing and Communications Engineering

TOIVIAINEN, SAMPPA: Inertial Sensor Mouse for a Mobile Device
Master of Science Thesis, 42 pages
May 2013
Major: Embedded Systems
Examiners: Prof. Jarmo Takala ja Lic. Helena Leppäkoski
Keywords: accelerometer, gyroscope, mouse, iOS

This thesis describes how a mobile application can be created that can be used as a computer mouse. This mobile application takes advantage of inertial sensors that mobile device contains. Mobile application described in this thesis can be connected wirelessly to the computer.

Apple iPhone 4 was chosen as the mobile device in this thesis. It contains 3-axis accelerometer and 3-axis gyroscope. The mobile application described in this thesis also requires application that is run on the computer user wants to control with the mobile application. This computer application performs mouse events generated by the user of the mobile application. Computer application was created for Windows and OS X operating systems. This thesis shows the structrure of these applications and the user interface of the mobile application. iPhone 4 has WiFi and Bluetooth capabilities and these are used to create a network between the mobile device and the computer. These devices communicate using a custom protocol built on UDP.

Performance measurements were also performed on the inertial sensors and connection methods used by the mobile application. Delays caused by different connection methods were measured. There are also measurements about the quality of samples provided by intertial sensors in the mobile device. In Addition, the maximum sampling rate for reading the inertial sensors was measured.

ALKUSANAT

Tämä diplomityö on tehty Tampereen teknillisen yliopiston Tietotekniikan laitokselle ja sen tavoitteena oli toteuttaa mobiilialustalle inertia-antureita hyödyntävä hiirisovellus. Tämä työ on tavallaan jatkoa kandidaatintyönä toteuttamalleni kiihtyvyysanturilla toimivalle hiirelle. Työn tarkastajina ovat toimineet prof. Jarmo Takala ja TkL Helena Leppäkoski.

Haluan kiittää Jarmo Takalaa ja Helena Leppäkoskea tähän työhön liittyneestä ohjauksesta, sekä mahdollisuudesta toteuttaa tämä diplomityö Tietotekniikan laitokselle.

Tampereella 14.4.2013

Samppa Toiviainen

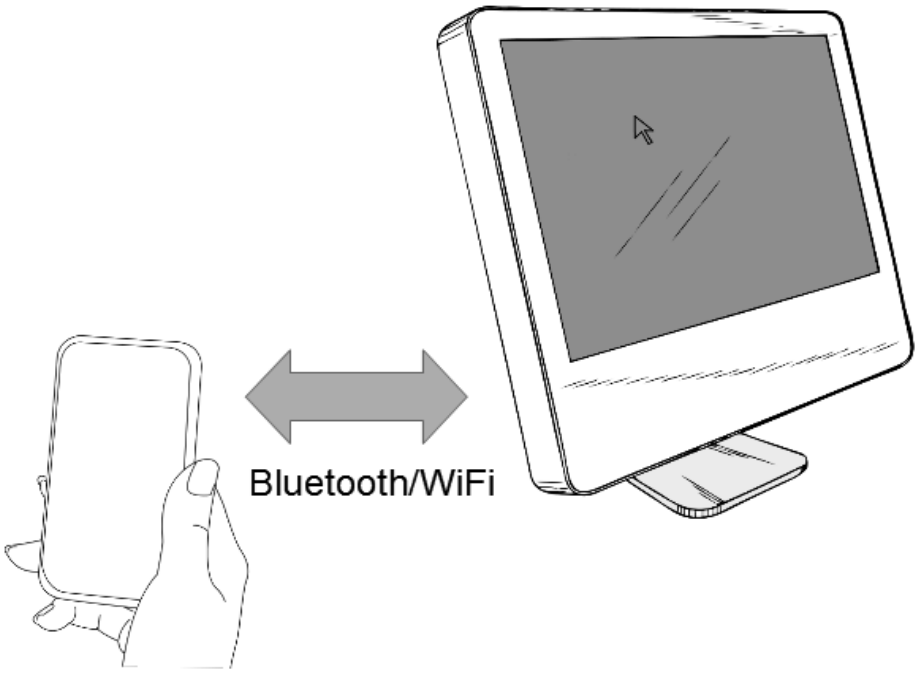
SISÄLLYS

TERMIT JA LYHENTEET

Termi	Selitys
ASCII	7-bittinen 128 merkin tietokoneiden merkistö.
Bluetooth	Langaton kommunikointi standardi.
instanssi	Olio-ohjelmoinnissa luokan edustaja.
iOS	Applen kehittämä mobiilikäyttöjärjestelmä.
IP	Internet-kerroksen protokolla, joka huolehtii IP-pakettien toimittamisesta perille IP-osoitteen perusteella.
kapselointi	Yhteen kuuluvien tietojen ja toimintojen yhdistäminen ja piilottaminen.
MVC	Ohjelmistoarkkitehtuuri, jonka tarkoituksena on erottaa käyttöliittymä ja sovellusaluetieto. (engl. Model-View-Controller)
RTT	Se aika, joka kuluu signaalin lähetyksestä lähetyksestä saatuun kuittaukseen. (engl. Round Trip Time)
SDK	SDK:lla tarkoitetaan tyypillisesti sovelluskehitykseen tarkoitettua työkalupakkia. (engl. Software Development Kit)
socket	Verkon yli tapahtuvan kommunikoinnin päätepiste.
Broadcast-osoite	Ethernet aliverkon osoite, johon lähetettävät paketit kaikki aliverkon laitteet vastaanottavat.
UDP	Yhteydetön verkkoprotokolla.
USB	Sarjaväylä (engl. Universal Serial Bus)
WiFi	Langaton lähiverkkotekniikka. (engl. Wireless Fidelity)
Xcode	Applen tarjoama ohjelmointiympäristö. Sitä käytetään iOS ja Mac OS X -kehityksessä.

1 JOHDANTO

Erilaisia liikettä tunnistavia antureita käytetään monissa ohjauslaitteissa. Yleisimpiä tälläisiä antureita ovat kiihtyvyysanturit ja gyroskoopit. Kiihtyvyysanturin tarkoituksena on mitata kiihtyvyyttä ja gyroskoopin tarkoitus on mitata pyörimisliikkeen nopeutta. Nykyään tälläisiä antureita on sisällytetty myös erilaisiin mobiililaitteisiin, kuten matkapuhelimiin ja tablet-koneisiin. Tälläisissä mobiililaitteissa liikettä tunnistavia antureita voidaan hyödyntää esimerkiksi laitteen asennon määrittämiseen tai mobiilipeleissä ohjaussyötteen keräämiseen.



Kuva 1: Työn tarkoituksena oli tehdä langaton hiirisovellus mobiililaitteelle, jonka avulla mobiililaitetta voidaan käyttää tietokoneen hiirenä.

Aina taskussa mukana kulkeviin mobiililaitteisiin on mahdollista asentaa monenlaisia elämää helpottavia ja arkipäivän tehtävissä auttavia sovelluksia. Tämän työn tarkoituksena on tehdä kiihtyvyysanturilla ja gyroskoopilla varustettuun mobiililaitteeseen sovellus, jonka avulla mobiililaitetta voidaan käyttää langattomasti tietokoneen hiirenä. Sovelluksen tarkoituksena on mahdollistaa tietokoneen hiiriperustaisen käyttöliittymän käyttäminen matkapuhelimen avulla. Tälläisen inertia-

antureilla toimivan hiiren etuna perinteisiin pöytäpinnalla toimiviin hiiriin on mahdollisuus käyttää sitä paikassa tai tilanteessa, jossa ei ole tasaista alustaa käytettävissä. Tämänkaltaisia tilanteita voisivat olla esimerkiksi mediatietokoneen käyttö sohvalta tai PowerPoint -esityksen pitäminen videoprojektorilla. Lisäksi matkapuhelin kulkee periaatteessa aina käyttäjän mukana, joten tämän ansiosta sovelluksen käyttäjän ei tarvitsisi kantaa mukana erityistä tähän tarkoitukseen valmistettua hiirtä.

Kuva 2: Logitech MX Air on langaton gyroskooppia käyttävä hiiri.

Työn aloittamishetkellä gyroskoopilla varustetut mobiililaitteet olivat harvemmassa, mutta kiihtyvyysanturilla varustetut olivat yleisiä. Tämän takia työn toteuttamisalustaksi valittiin Applen iPhone 4. Kyseisestä laitteesta löytyvät molemmat anturit. Laitteesta löytyy lisäksi myös Bluetooth ja WiFi, joita voitaisiin käyttää langattoman yhteyden muodostamiseen mobiililaitteen ja tietokoneen välille. [1] Kuvassa 1 on esitettynä periaatekuva toteutetusta työstä. Mobiililaitetta kääntämällä voidaan liikuttaa tietokoneen näytöllä näkyvää kursoria.

Joitakin valmiita kaupallisia inertia-antureita hyödyntäviä hiiritoteutuksia löytyy markkinoilta. Yksi kaupallinen toteutus gyroskooppia hyödyntävästä hiirestä on Logitechin valmistama MX Air -hiiri, joka on kuvassa 2. [2] Gyroskoopin lisäksi siitä löytyy laser-anturi, joten hiirtä voi myös käyttää tasaista pintaa vasten tavanomaisen hiiren tapaan. Logitech MX Air on langaton ja se sisältää erillisen verkkovirtaan kytketyn latausaseman, jonka avulla sen akku voidaan ladata. Langaton yhteys tietokoneen ja hiiren avulla on toteutettu hiiren mukana toimitettavan USB-vastaanottimen avulla. Logitech MX Air -hiiri osaa havaita lennosta pöytäpinnan ja näin

vaihto gyro- ja laseranturin välillä tapahtuu itsestään käyttötavasta riippuen. Logitech MX Air on suunniteltu käytettäväksi gyrohiirenä ja sen sisältämä laitteisto on tätä varten suunniteltu, joten se toimii paremmin tässä tarkoituksessa kuin geneerisemmälle mobiililaitealustalle toteutettu hiirisovellus.

Kuva 3: Mobile Mouse -hiirisovelluksen käyttöliittymä.

IPhone 4 puhelimessakin käytössä olevalle iOS-alustalle on saatavilla Mobile Mouse -niminen sovellus, jonka ilmaisversio tukee puhelimen kosketusnäytön käyttämistä hiiren kursorin liikuttamiseen. Maksullisessa Pro -versiossa on lisäksi mahdollisuus käyttää matkapuhelimessa olevia antureita kursorin liikuttamiseen. Kyseisestä sovelluksesta löytyy myös versio Android-käyttöjärjestelmälle.[3] Kuvassa 3 on kuvankaappaus Mobile Mouse -sovelluksen käyttöliittymästä. Kuvastakin voi päätellä että hiiren kursorin liikuttamisen lisäksi sovelluksella voi lähettää myös näppäinkomentoja tietokoneelle. Sovelluksen ja tietokoneen välinen yhteydenpito on toteutettu käyttämällä WiFi-yhteyttä ja sen käyttäminen edellyttää erillisen sovelluksen asentamista tietokoneelle. Tästä tietokonesovelluksesta löytyy versiot Windows, OS X ja Linux -käyttöjärjestelmille. Tämä ratkaisu muistuttaa paljon sitä, johon itse päädyin toteuttaessani puhelimen ja tietokoneen välistä kommunikointia toteuttamaani hiirisovellukseen.

Tein kandidaatintyön kiihtyvyysanturilla toimivasta hiirestä, joten kiihtyvyysanturin käyttö tähän tarkoitukseen oli jo tuttu. Tässä työssä painopiste on enemmän

gyroskoopin mahdollisuuksien tutkimisessa tähän tarkoitukseen. Tämän lisäksi tarkastellaan hieman anturien yhteiskäyttöä tähän tarkoitukseen. Työssä esitellään kätetyt inertia-anturit ja periaatteet niiden käyttämisestä kursorin liikuttamiseen sekä iPhone 4 -laitteesta löytyvä iOS -alusta. Tämän lisäksi käydään läpi toteutettu mobiilisovellus ja tietokonesovellus, sekä näiden väliseen yhteydenpitoon käytetty kommunikointiprotokolla. Sovelluksista käydään läpi niiden rakenteet ja käyttöliittymä. Lopuksi esitellään mittaustietoa toteutetun hiiren toimintaan vaikuttavista asioista, kuten mobiililaitteen antureista ja laitteen ja tietokoneen välisen yhteyden aiheuttamista viiveistä.

2 PERIAATTEET JA TAUSTAT

Kappaleessa käydään työssä käytetyt inertia-anturit ja niiden sovelluksia. Tämän lisäksi esitellään käytössä olevan iPhone 4 -mobiililaitteen iOS -alustaa ja sille tapahtuvaa ohjelmistokehitystä.

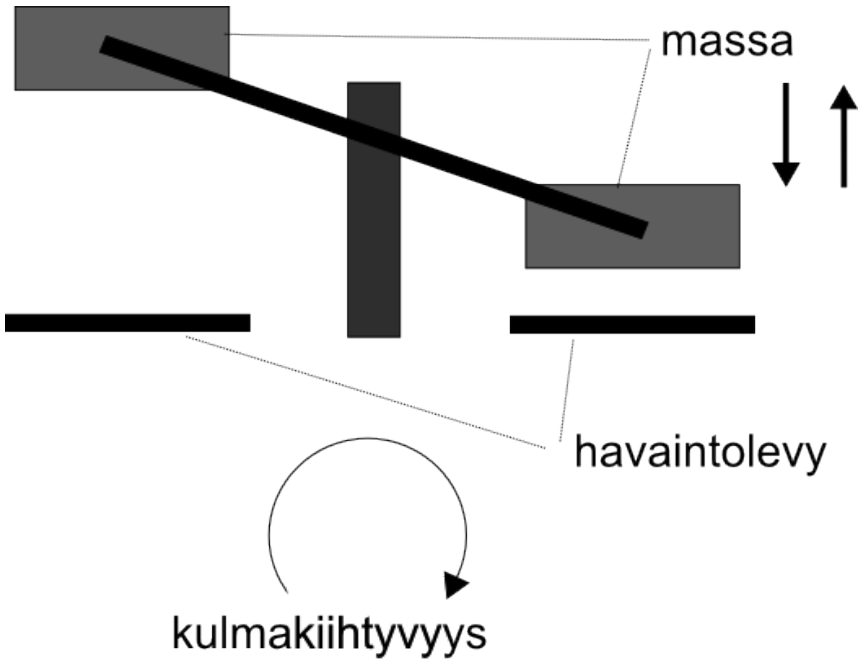
2.1 Inertia-anturit

Inertia-anturit mittaavat anturiin kohdistuvaa liikettä. Inertia tarkoittaa kappaleen taipumusta vastustaa liiketilansa muutosta ja inertia-antureiden toiminta perustuukin usein tähän. Tässä käsitellään tarkemmin gyroskoopit ja kiihtyvyysanturit, joita molempia hyödynnetään toteutetussa hiirisovelluksessa.

2.1.1 Gyroskoopit

Gyroskooppi on komponentti, joka mittaa anturin akselin ympäri vaikuttavaa kulmanopeutta. Akselin ympäri vaikuttavan kulmanopeuden mittaamiseen on monia tapoja. Perinteinen mekaaninen gyro perustuu pyörivän metallisen pyörän pyrkimykseen vastustaa kiertoa mitattavan akselin ympäri. Tämänkaltainen gyro ei ole kovin kätevä kun tarvitaan pienikokoista ja vähän energiaa kuluttavaa gyrototeutusta. [4]

Mikromekaaniset gyrot ovat pieniä ja vähävirtaisia. Ne valmistetaan piilastulle samaan tapaan kuin mikropiirit. Mikromekaaniset gyrot eivät perustu massan pyörimiseen, vaan massan värähtelyyn. Kun nämä värähtelevät massat altistuvat pyörimiselle mitattavan akselin ympäri, muodostuu Coriolis-kiihtyvyys, joka on kohtisuorassa sekä värähtelyliikkeeseen että pyörimisliikkeen akseliin nähden. Coriolis-kiihtyvyys liikuttaa värähteleviä massoja havaintolevyistä etäämmäksi tai niitä lähemmäksi aiheuttaen kapasitanssin vaihtelua. Tästä kapasitanssista voidaan päätellä pyörimisnopeuden suuruus. Kuva 4 selventää tätä. [4]



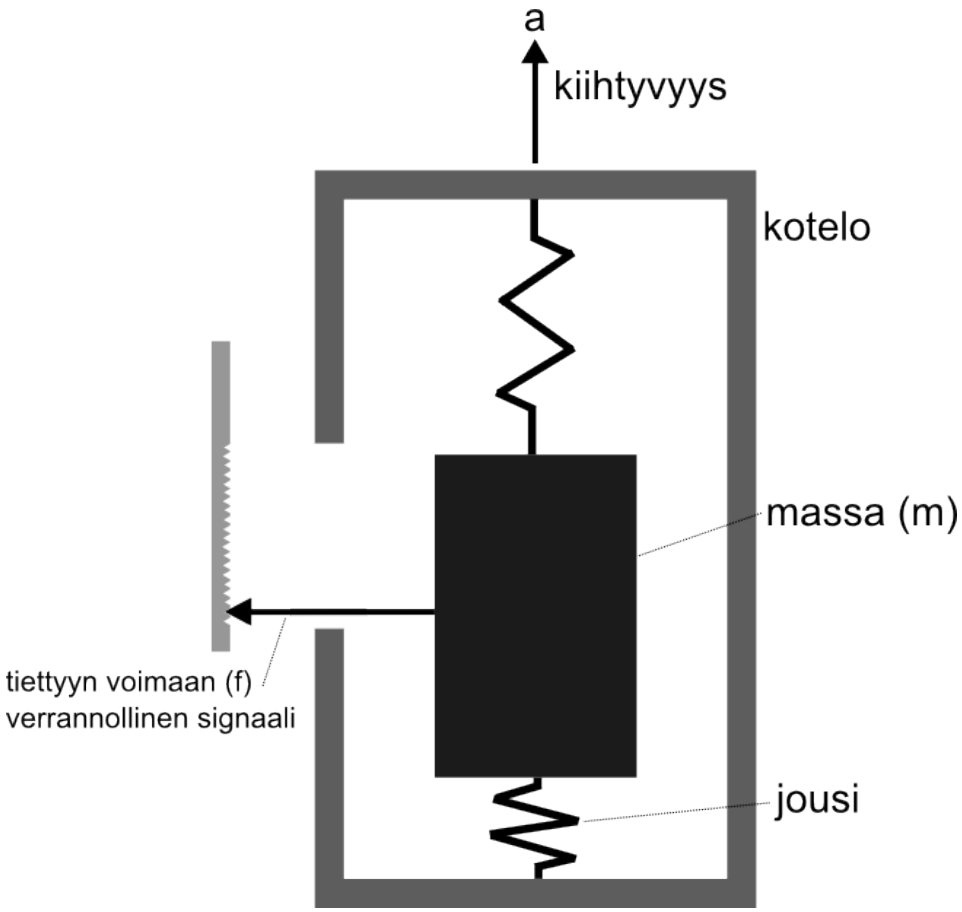
Kuva 4: Coriolis-kiihtyvyyden takia gyron värähtelevät massat aiheuttavat kapasitanssin vaihtelua mittauslevyihin nähden. [4]

Työssä käytetty iPhone 4 mobiililaite sisältää kolmiakselisen gyron, jonka ansiosta kulmanopeutta voidaan mitata kolmen akselin suhteen. Käytetyn anturin tarkka malli ei ole selvillä. Chipworks kuitenkin epäilee sen olevan STMicroelectronicsin valmistama ja hyvin lähellä STMicroelectronicsin kaupallista L3G4200D -anturia. [5]

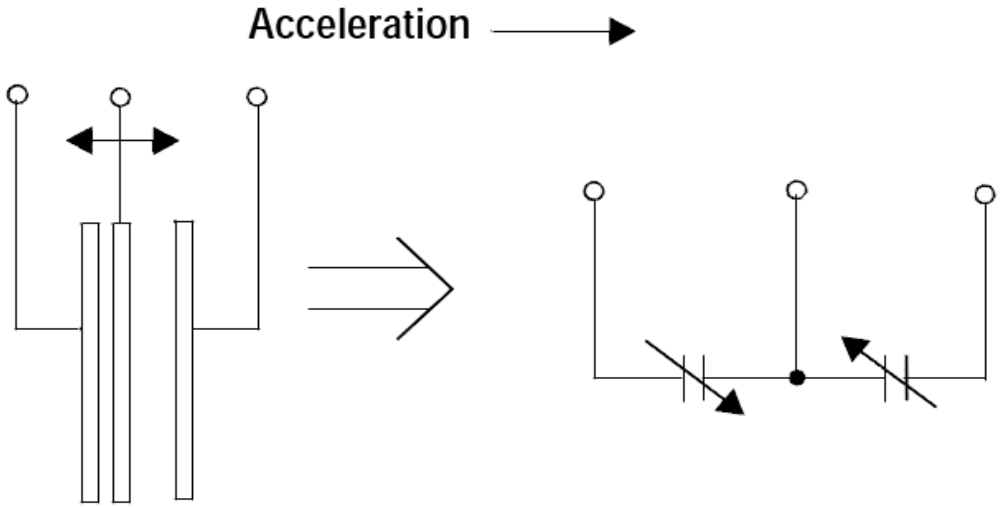
2.1.2 Kiihtyvyysanturit

Kiihtyvyysanturi on komponentti, joka mittaa anturiin vaikuttavaa kiihtyvyyttä. Kuvassa 5 on esitetty yksinkertainen kiihtyvyysanturi. Kiihtyvyysanturin toiminta perustuu, kuten inertia-antureille on tyypillistä, massan taipumukseen vastustaa liiketilansa muutosta. [4] Kiihtyvyysanturin avulla voidaan mitata mm. värinää, yhteentörmäystä, kallistuskulmaa tai havaita vapaa putoaminen. Toteutetussa hiirisovelluksessa käytetään hyväksi mahdollisuutta mitata kallistuskulmaa.

Tässä työssä käytetty iPhone 4 -mobiililaite sisältää kiihtyvyysanturin, joka mittaa kiihtyvyyttä kolmen eri akselin suhteen. Jotkin anturit saattavat mitata vain kahden tai jopa yhden akselin kiihtyvyyttä. iPhone 4 mobiililaite sisältää STMicroelectronicsin LIS331DLH kiihtyvyysanturin [5]. Tämä mikromekaaninen kiihtyvyysanturi käyttää kiihtyvyyden mittaukseen kapasitanssin vaihtelua. Anturiin kohdistuva kiihtyvyys liikuttaa sen sisäisiä massoja aiheuttaen kapasitanssin vaihtelua. Periaate on esitetty kuvassa 6. [6]



Kuva 5: Yksinkertainen kiihtyvyysanturi.[4]



Kuva 6: LIS331DLH -kiihtyvyysanturin toiminta perustuu vaihtelevien kapasitanssien mittaukseen.

2.2 Inertiamittausten sovelluksia

Inertiamittausten sovelluskohteita on paljon. Inertia-antureita käytetään esimerkiksi teollisuudessa, navigoinnissa ja viihde-elektroniikassa. Erityisesti vähävirtaiset ja pienikokoiset mikromekaaniset anturit ovat lisänneet inertia-antureitten käyttömahdollisuuksia.

Viihde-elektroniikassa ehkä menestyksekkäin sovellus löytyy Nintendo Wii:n ohjaimesta, joka on esitetty kuvassa 8. Langaton Wii Remote -ohjain hyödyntää kolmiulotteista kiihtyvyysanturia, jota käytetään pelaajan liikkeiden tunnistamiseen.[7] Ohjaimeen on saatavilla myös MotionPlus -laajennus, joka tuo ohjaimeen myös kolmiulotteisen gyron. Gyroskoopin ja kiihtyvyysanturin avulla saadaan tunnistettua entistä monimutkaisempia pelaajan liikkeitä.

Kuva 7: Wii Remote ohjaimessa on kolmiulotteinen kiihtyvyysanturi. Kuvassa ohjaimen alaosaan on kiinnitetty MotionPlus -laajennus, joka lisää ohjaimeen kolmiulotteisen gyron.

Inertiamittauksia hyödynnetään laajalti navigaatiojärjestelmissä. Tälläisille navigaatiojärjestelmille on sekä siviili- että sotilaallisia käyttökohteita. Ensimmäinen inertiaohjausjärjestelmä esiteltiin vuonna 1942 saksalaisissa V2 -ohjuksissa. Niitä lötyy nykyään risteilyohjuksien lisäksi esimerkiksi lentokoneista ja sukellusveneistä. Kiihtyvyysantureiden avulla voidaan saada integroimalla arvio laitteen paikasta ja nopeudesta. Gyroskooppien avulla järjestelmään saadaan lisäksi tieto laitteen kulmanopeudesta, josta voidaan selvittää laitteen asento. Teollisuudessa

inertiamittauksista on hyötyä myös esimerkiksi teollisuusrobottien hallinnassa. [8]

Mobiililaitteissa, kuten matkapuhelimissa tai tablet-koneissa, inertiamittauksia voidaan hyödyntää usealla tavalla. Inertia-antureilla on mahdollista saada selville laitteen asento, jolloin esimerkiksi näyttöruutu voidaan asemoida sopivasti tai kääntää kameralla otettu kuva oikeaan asentoon. Tämän lisäksi näitä antureita voidaan hyödyntää esimerkiksi mobiilipeleissä ohjaukseen. Mobiililaitteille löytyy myös navigaatiosovelluksia, joten mobiililaitteesta löytyviä inertia-antureita voitaisiin käyttää myös navigaatiotarkoituksiin.

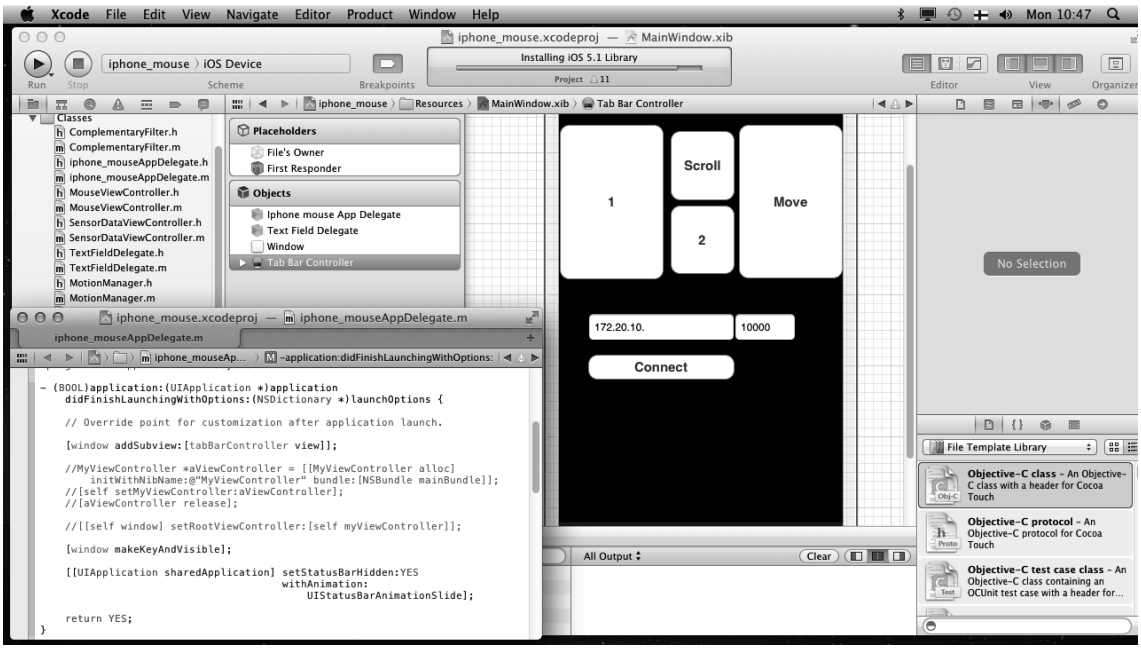
2.3 iOS

iOS on Applen mobiililaitteissa käytössä oleva mobiilikäyttöjärjestelmä. Kyseinen käyttöjärjestelmä löytyy Applen iPhone- ja iPad-tuoteperheiden laitteista ja joistain iPod -malleista. Käyttöjärjestelmän ensimmäinen, kaupallisesta tuotteesta löytynyt, versio 1.0 julkaistiin vuonna 2007. Tällöin julkaistiin ensimmäinen iPhone -puhelin. Tämän tekstin kirjoittamishetkellä käyttöjärjestelmän uusin versio on 6.0.2.

iOS-käyttöjärjestelmän ensimmäinen versio ei vielä sallinut kolmannen osapuolen tekemien sovelluksien käyttämistä laitteella. Vasta samaan aikaan iPhone 3G -laitteen julkaisun kanssa julkistettu iOS 2.0 esitteli App Storen, jonka avulla laitteen käyttäjien on mahdollista asentaa kolmannen osapuolen tekemiä sovelluksia iOS -käyttöjärjestelmälle. App Store on Applen tarjoama sovelluksien kauppapaikka, jonka kautta sovelluskehittäjien on julkaistava sovelluksensa. Tämä tarkoittaa sitä että Applella on täysin hallinnassa se mitä ohjelmia muokkaamattomalla iOS -alustalla voidaan ajaa. Ennen ohjelman julkaisua sovellus tulee lähettää Applelle tarkistettavaksi.

2.3.1 Ohjelmistokehitys iOS-alustalle

Sovellukset iOS -käyttöjärjestelmälle kehitetään Objective-C -kielellä, joka on C++:n kaltainen oliolaajennus C-kieleen. IOS -sovelluksen tekeminen täysin ilman Objective-C kieltä on mahdotonta[9]. Itse kehitystyöhön Apple tarjoaa XCode -kehitysympäristön ja sille tarkoitetun iOS SDK:n. XCode:n käyttöliittymä on esitetty kuvassa 8. XCode tarjoaa työkalut ohjelmakoodin kirjoittamiseen, käyttöliittymän toteuttamiseen, testaukseen, optimointiin ja sovelluksen julkaisuun App Storessa.[10]

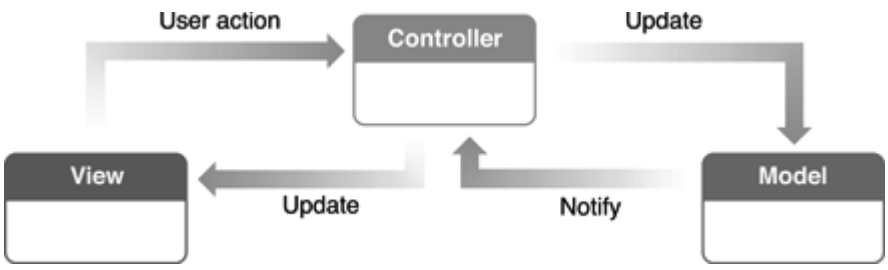


Kuva 8: XCode -kehitysympäristö.

iOS -sovellus koostuu kehittäjän omasta ohjelmakoodista ja Applen tarjoamista rajapinnoista. Nämä rajapinnat ovat luokkakirjastoja, jotka sisältävät valmiita metodeja ja rakenteita. Sovelluskehittäjä voivat hyödyntää omissa ohjelmissaan näitä luokkakirjastoja. iOS -käyttöjärjestelmä voidaan jakaa neljään kerrokseen. Jokainen rajapinta kuuluu johonkin näistä iOS -järjestelmän kerroksista. iOS -järjestelmän kerrokset on esitetty kuvassa 9. Kehittäjän on suositeltavaa suosia mahdollisimman korkean tason rajapintoja, sillä nämä tarjoavat helppoja valmiiksi testattuja tapoja käyttää alemman tason ominaisuuksia. Käyttöjärjestelmän alemmat kerrokset tarjoavat yksinkertaisia alemman tason ominaisuuksia, joiden päälle korkeammat kerroksien monimutkaisemmat abstraktiot on rakennettu. [12] iOS -käyttöjärjestelmän alin kerros on Core OS, joka sisältää kaikista alhaisimman tason metodeja. Core OS kerrokseet kuuluu muunmuassa muistihallintaan ja tiedostojärjestelmän käyttöön liittyviä metodeja. Core Services -kerros sisältää hieman korkeamman tason ominaisuuksia kuin Core OS. Tämä kerros sisältää iOS -kehityksessä tärkeän Foundation -rajapinnan, joka tarjoaa esimerkiksi erilaisia tietorakenteita, työkaluja merkkijonojen käsittelyyn ja rajapinnan säikeiden käyttöön. Mediakerros sisältää alemman tason rajapintoja äänen, grafiikan ja animaatioiden esittämiseen. iOS -käyttöjärjestelmän korkein Cocoa Touch -kerros on avainasemassa iOS -ohjelmien kehityksessä. Se sisältää ohjelmien perusinfastruktuurin, tarjoaa ohjelmalle kosketysnäytön syötteen ja muita usein käytettyjä järjestelmän korkean tason palveluita. Ohjelmistokehittäjän tulisi ensisijaisesti käyttää tämän kerroksen rajapintoja ja jos sopivaa ominaisuutta ei löydy, niin sitten vasta turvautua alemman kerroksen rajapintoihin. [11]

Kuva 9: iOS:n kerrokset ja joitakin tärkeimpiä ohjelmistokehyksiä. [11]

Sovelluskehitys iOS-alustalle on suurelta osin eri rajapintojen yhdistämistä kehittäjän omalla koodilla. Nämä käytetyt rajapinnat edellyttävät jossain määrin sitä että kehittäjän koodin tulee sopeutua rajapintojen rakenteisiin. Suunnittelumallit ovat olio-ohjelmoinnissa työkaluja, joilla voidaan ratkaista tiettyjä usein toistuvia geneerisiä ongelmia. iOS -ohjelmoinnissa tärkeä ja paljon käytetty suunnittelumalli on MVC (Model-View-Controller). Suunnittelumalli on esitetty kuvassa 10. Malli koostuu kolmesta oliosta, jotka ovat malli, näkymä ja kontrolleri. Tämä malli määrittää ohjelman olioiden roolit ja tavan, jolla ne kommunikoivat. Malli on olio, joka kapseloi ohjelman datan ja prosessoi sitä. Näkymä on olio, joka näkyy ohjelman käyttäjälle. Olio osaa piirtää itsensä ja se osaa vastaanottaa käyttäjän syötettä. iOS järjestelmässä UIKit rajapinta tarjoaa valmiita näkymäluokkia. Kontrolleriolio toimii näkymän ja mallin välissä. iOS ohjelmoinnissa jokainen näkymä vaatii kontrolleriolion. Näkymäolio ilmoittaa käyttäjän syötteestä kontrollerille ja kontrolleri päivittää mallia ja näkymää tarpeen mukaan. Malli voi myös lähettää ilmoituksia kontrollerioliolle. [13]

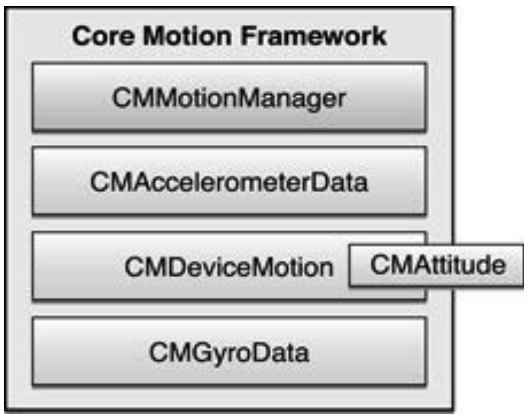


Kuva 10: Model-View-Controller suunnittelumalli. [13]

2.3.2 Inertia-antureiden käyttäminen iOS-järjestelmässä

Jo ensimmäisessä vuonna 2007 julkaistussa iPhone-matkapuhelimessa oli kolmiulotteinen kiihtyvyysanturi. Tästä voidaan päätellä, että inertia-anturit ovat olleet osa iOS-kehitystä ihan alusta asti. Vuonna 2010 julkaistussa iPhone 4 -matkapuhelimessa oli kiihtyvyysanturin lisäksi kolmiulotteinen gyroskooppi.

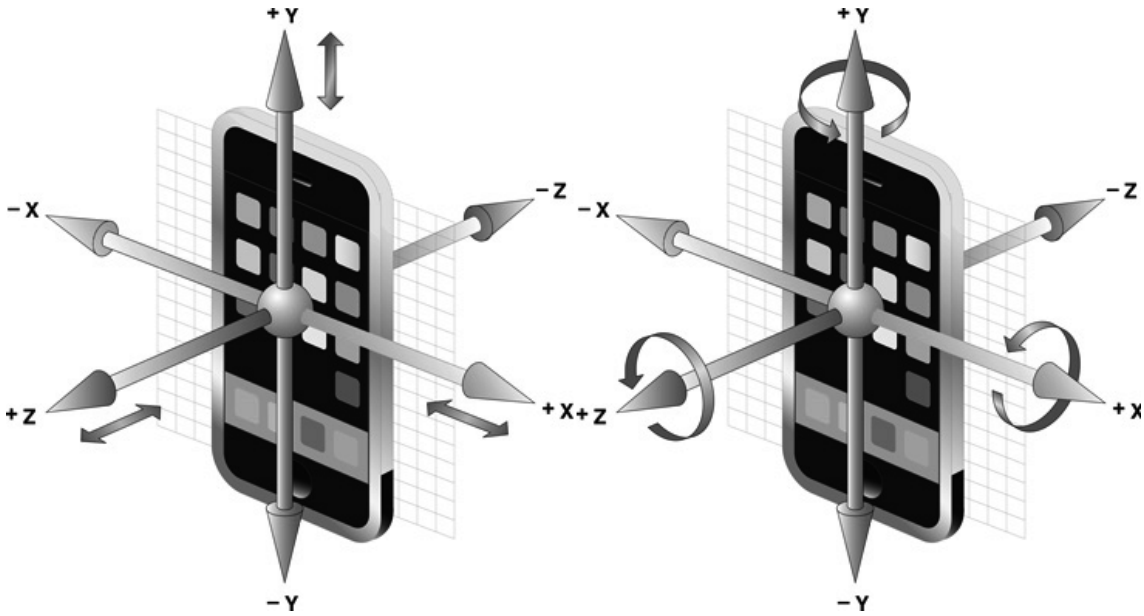
Kehittäjä voi iOS-järjestelmässä lukea antureilta saatavaa mittaustietoa suoraan tai antaa käyttöjärjestelmän tunnistaa tästä mittaustiedosta tiettyjä etukäteen määriteltyjä eleitä, kuten laitteen ravistamisen. Näiden eleiden vastaanotto edellyttää, että kontrolleriolio on periytytetty *UIResponder* -luokasta ja sen tulee toteuttaa joko *motionBegan:withEvent:* tai *motionEnded:withEvent:* -funktio. Käyttöjärjestelmältä on myös mahdollisuus pyytää huomautuksia, kun mobiililaitteen ruutu kääntyy pystysuunnasta vaakatasoon tai toisin päin. [15]



Kuva 11: Core Motion rajapinnan luokat.[15]

Inertia-antureiden lukeminen iOS-järjestelmässä tapahtuu Core Motion -rajapinnan avulla. Tämä kirjasto huolehtii inertia-antureiden mittaustiedon lukemisesta ja tarjoaa sen käytettäväksi järjestelmän muille osille tai kolmannen osapuolen sovellukselle. Kuvassa 11 on esitetty Core Motion -rajapinnan luokat. *CMAccelerometerData* -luokka kuvaa kiihtyvyysantureilta tietyllä ajan hetkellä luettuja näytearvoja ja *CMGyroData* -luokka kuvaa gyroskoopilta luettuja näytearvoja tietyllä ajan hetkellä. CMDeviceMotion -luokka kapseloi sisäänsä laitteen kaikkien akseleiden kulmanopeudet, kiihtyvyydet, painovoimavektorin sekä laitteen asennon. Laitteen asennon määrittämiseen käytetään Applen dokumentaation mukaan sekä kiihtyvyysanturia että gyroskooppia. Luettavissa oleva kiihtyvyys on G-yksikössä ja muuttuja on liukulukutyyppiä double. Kulmanopeus ilmoitetaan radiaaneina sekunnissa ja muuttuja on liukulukutyyppiä double. iOS:n versio 5.0 toi mukanaan CMDeviceMotion -luokkaan mahdollisuuden pyytää siltä magneettikenttään liittyvää

mittaustietoa. Tämän laskemiseen käytetään joistain laitteista löytyvää magnetometriä. Inertia-antureiden akselien suunnat suhteutettuna iOS-laitteeseen löytyvät kuvasta 12. *CMMotionManager* on hallintaluokka, jonka avulla kehittäjä pääsee käsiksi iOS -järjestelmän liikkeentunnistuspalveluihin. [15]

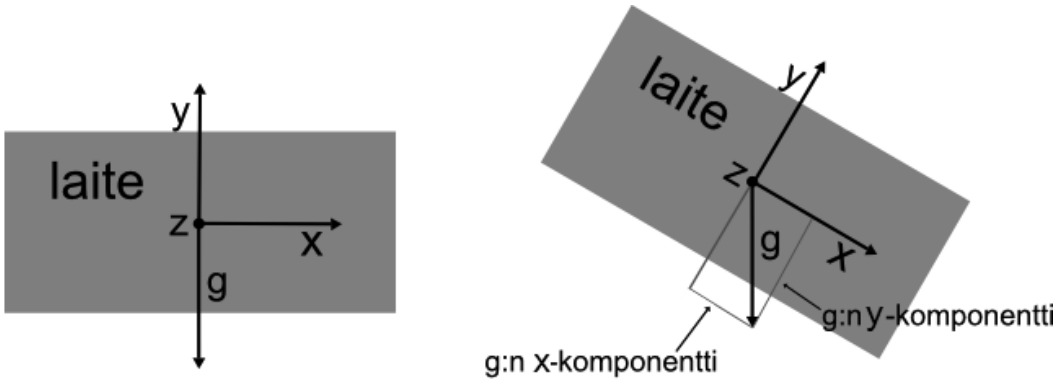


Kuva 12: iOS-laitteen inertia-antureiden akselit. [15]

3 HIIRISOVELLUS

Tehdyn hiirisovelluksen avulla käyttäjä voi käyttää mobiililaitetta tietokoneen hiiren korvaajana. Käyttäjä voi liikuttaa kursoria mobiililaitetta liikuttamalla ja käyttää hiiren painikkeita kosketusnäytön avulla. Käyttäjä voi lisäksi sovelluksesta käsin vaikuttaa hiiren toimintaa muuttaviin asetuksiin. Näihin asetuksiin lukeutuvat kursorin liikkeen herkkyys, kaksoisklikkauksen klikkausten välisen ajan maksimipituus ja toimintatilan valinta.

Hiirisovellus tehtiin mobiililaitteelle, joka sisältää kolmiulotteisen gyro- ja kiihtyvyysanturin. Tämä tarkoittaa sitä että anturilta luettavat näytearvot on käsiteltävä ja tulkittava siten että niiden avulla kursorin liikutus tietokoneen näytöllä on mahdollista. Näiden kahden anturin avulla päätin toteuttaa sovellukseen kaksi erilaista tapaa käyttää antureita hiiren kursorin liikuttamiseen. Näistä tavoista toinen sopii paremmin kiihtyvyysanturille ja toinen gyroskoopille.



Kuva 13: Laitteen kallistukseen perustuvan toimintatilan periaatekuva.

Kiihtyvyysanturille paremmin sopiva tapa mahdollistaa hiiren kursorin liikuttamisen mobiililaitetta kallistamalla. Laitetta kallistaessa maapallon vetovoimakiihtyvyys aiheuttaa kiihtyvyysanturin vaakatasossa oleviin akseleihin kiihtyvyyttä. Hiirisovellusta käytettäessä mobiililaitetta on tarkoitus pitää kädessä näyttö ylöspäin. Mitä suurempaan kulmaan laitetta kallistaa, sitä suuremmaksi anturin mittaama kiihtyvyys kasvaa. Kuva 13 havainnollistaa tätä. Anturilta luettavaa kiihtyvyyden arvoa voidaan käyttää hiiren kursorin liikenopeuden muodostamiseen. Arvo tarvitsee vain skaalata kokeilemalla sopivaksi. Tämä tarkoittaa, että kursorin liikenopeus on verrannollinen anturin

mittaaman kiihtyvyyden suuruuteen. Hiiren kursorin liikuttaminen tarvitsee vertikaalisen ja horisontaalisen nopeuden, joten anturilta pitää lukea kahden eri akselin kiihtyvyysarvot. Tämä onnistuu, sillä käytetystä mobiililaitteestä löytyy kolmiulotteinen kiihtyvyysanturi.

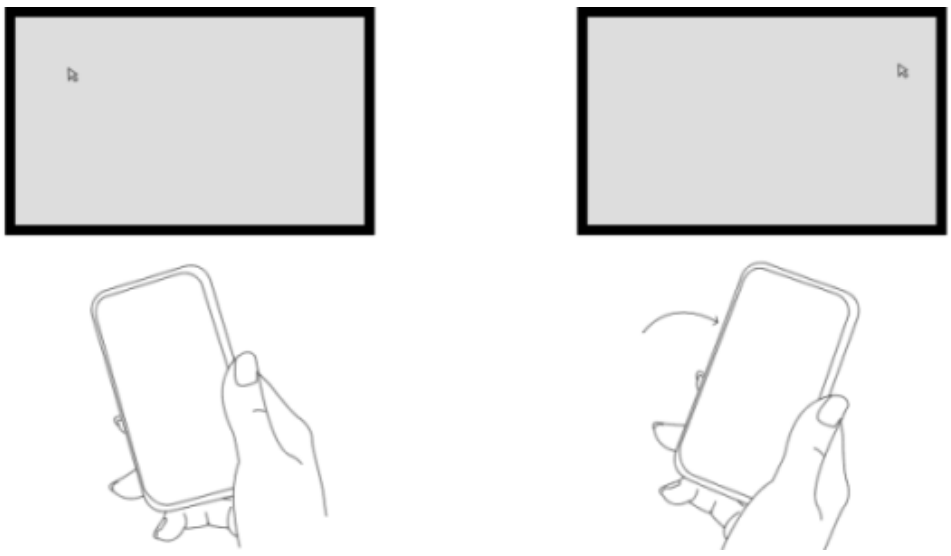
Gyroskoopille paremmin sopiva tapa mahdollistaa hiiren kursorin liikuttamisen mobiililaitetta kääntämällä. Tällä tavalla muodostuu illuusio siitä että hiiren kursoria ikään kuin liikutettaisiin osoittamalla näyttölaitetta. Gyroskooppi mittaa kulmanopeutta, jota laitteen kääntäminen aiheuttaa. Tämä kulmanopeutta voidaan skaalaamalla käyttää hiiren kursorin liikenopeutena. Kuva 14 havainnollistaa tätä. Tässäkin tapauksessa arvot pitää lukea kahdelta eri akselilta, jotta saadaan hiiren kursorille vertikaalinen ja horisontaalinen liikenopeus. Laite voi kuitenkin olla käyttäjän kädessä näyttölaitteeseen nähden kallellaan, jolloin mobiililaitteen akselit asettuvat näyttölaitteeseen nähden vinottain. Tätä ongelmaa pyritään korjaamaan trigonometrisia funktioita hyväksi käyttämällä. Korjattu kulmanopeus akselille x saadaan kaavalla:

$$knopeus_x = nayte_x \cdot \cos(a) + nayte_y \cdot \sin(a) \quad , \qquad (1)$$

missä $nayte_x$ on gyroskoopilta luettu x-akselin näytearvo, $nayte_y$ on gyroskoopilta luettu y-akselin näytearvo ja kulma a on radiaaneina laitteen sivuttainen kallistuskulma. Vastaavasti y-akselille saadaan korjattu kulmanopeus kaavalla:

$$knopeus_y = nayte_y \cdot \cos(a) - nayte_x \cdot \sin(a) \quad , \qquad (2)$$

missä $nayte_x$ on gyroskoopilta luettu x-akselin näytearvo, $nayte_y$ on gyroskoopilta luettu y-akselin näytearvo ja kulma a on radiaaneina laitteen sivuttainen kallistuskulma.



Kuva 14: Laitteen kääntämiseen perustuvan toimintatilan periaatekuva.

Kursorin liikuttamisen lisäksi toteutettu hiirisovellus tukee sivujen vieritystä. Sivujen vieritykseen käytetään aina laitteen kallistuskulmaa, eli käytetään vierityksen nopeuden muodostamiseen kiihtyvyysanturilta luettavia arvoja. Tässäkin tapauksessa tulee arvot skaalata kokeilemalla sopivaksi. Vieritys toimii siten että käyttäjä painaa vierityspainikkeen alas ja sen ollessa alhaalla kallistaa laitetta.

Työ rakentuu kahdesta osasta, joista toinen on mobiililaitteella ajettava iOS-sovellus ja toinen on tietokoneella ajettava sovellus, joka hoitaa nappien painallukset ja kursorin liikutuksen tietokoneella. Mobiililaitteella ajettava sovellus lukee antureilta saatavia näytearvoja ja muuttaa ne kursorin ja sivun vierityksien liikenopeuksiksi. Tämän lisäksi sovellus sisältää kosketusnäytöltä käytettäviä painikkeita, jotka toimivat tavallisen hiiren painikkeiden tavoin. Sovellus lähettää antureilta ja painikkeilta saamansa datan eteenpäin tietokoneella ajettavalle sovellukselle. Tietokoneella ajettavan sovellukses suorittaa hiiritoiminnot mobiililaitteella ajettavalta sovellukselta saamansa datan perusteella. Tietokonesovellus käyttää hiiritoimintojen suorittamiseen tietokoneella käytössä olevan käyttöjärjestelmän tarjoamia ohjelmointirajapintoja.

Mobiililaitesovelluksen ja tietokonesovelluksen välinen yhteydenpito hoidetaan käyttämällä UDP-verkkoprotokollaa. UDP -verkkoprotokollan käyttö edellyttää että tietokoneen ja mobiililaitteen välille muodostetaan IP-verkkoyhteys. Tämä voidaan tehdä liittämällä puhelin ja tietokone samaan WiFi-verkkoon tai käyttämällä puhelimen Personal Hotspot ominaisuutta. Personal Hotspot mahdollistaa IP-verkkoyhteyden muodostamisen USB-väylän, Bluetoothin tai WiFi-yhteyden ylitse[15]. Viestintä hoidetaan käyttämällä yksinkertaista kommunikointiprotokollaa, joka on rakennettu UDP -protokollan päälle.

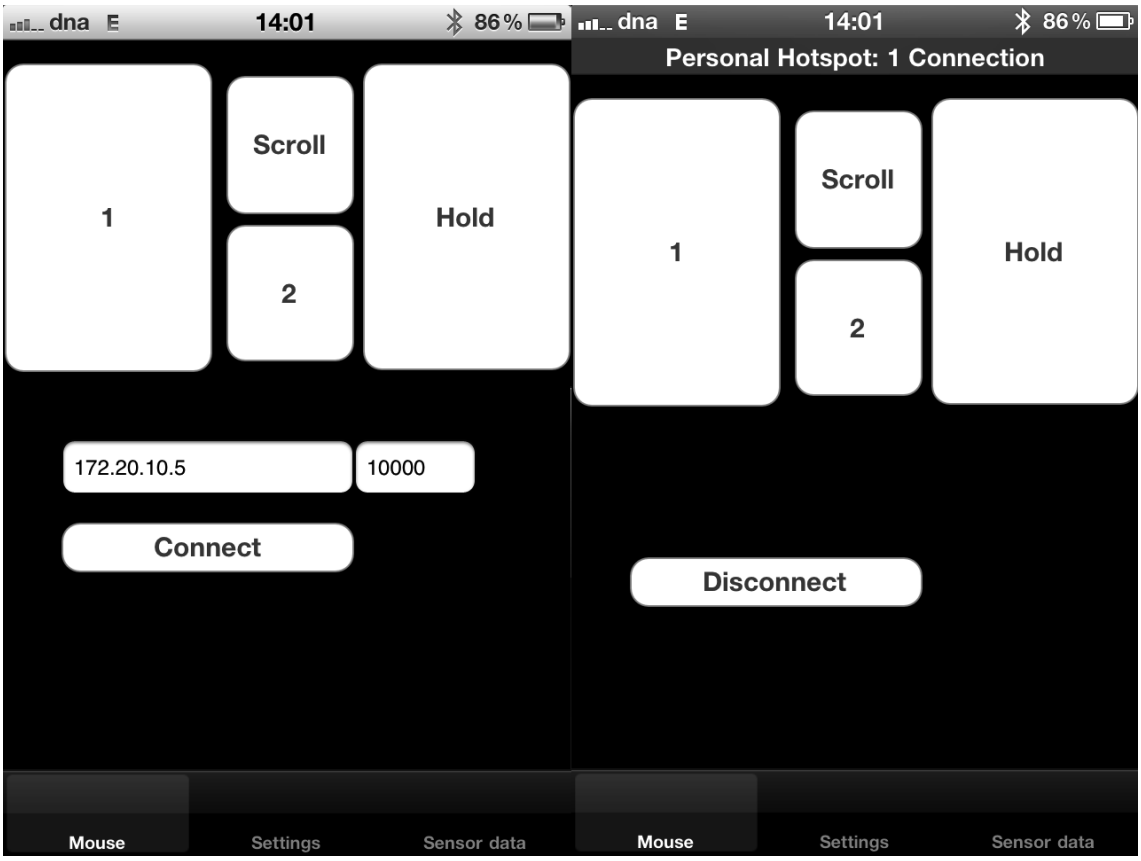
Hiirisovellus toteutettiin iPhone 4 -mobiililaitteelle. Sovelluksen tarkoitus on lukea käytössä olevia inertia-antureita ja painonappeja ja tulkita tämä data hiiren kursorin liikenopeudeksi, sivun vieritysnopeudeksi ja hiiren nappien painalluksiksi. Tämän lisäksi sovellus sisältää muutettavia asetuksia, joilla käyttäjä voi vaikuttaa hiiren toimintaan. Sovellus tehtiin alunperin iPhone 4 puhelimelle, jossa oli iOS 4.3.5 -käyttöjärjestelmä, mutta sovellus toimi muutoksitta myös uudemmassa iPhone 4S puhelimessa, jonka ohjelmistoversio oli 5.1.1. Sovelluksen tekemiseen käytettiin Xcode 4.3.2 kehitysympäristöä. Xcode on iOS -kehitykseen ainoa Applen tukema vaihtoehto.

3.1 Käyttöliittymä

Työssä käytössä olevassa iPhone 4 -puhelimessa on 3,5 tuuman kapasitiivinen kosketusnäyttö. Näytön resoluutio on 960 x 640. iOS-ohjelmointiin tarkoitettu XCode-kehitympäristö tarjoaa mahdollisuuden asetella ohjelman käyttöliittymän myös sivuttain, mutta toteutettuun sovellukseen pystysuunta tuntui luontevammalta.

Käytössä olevassa mobiililaitteessa ei ole juurikaan fyysisiä näppäimiä ja se on tarkoitettu käytettäväksi pääasiassa kosketusnäytön avulla. Tämän takia tehtyä iOS -sovellustakin käytetään kosketusnäytön avulla.

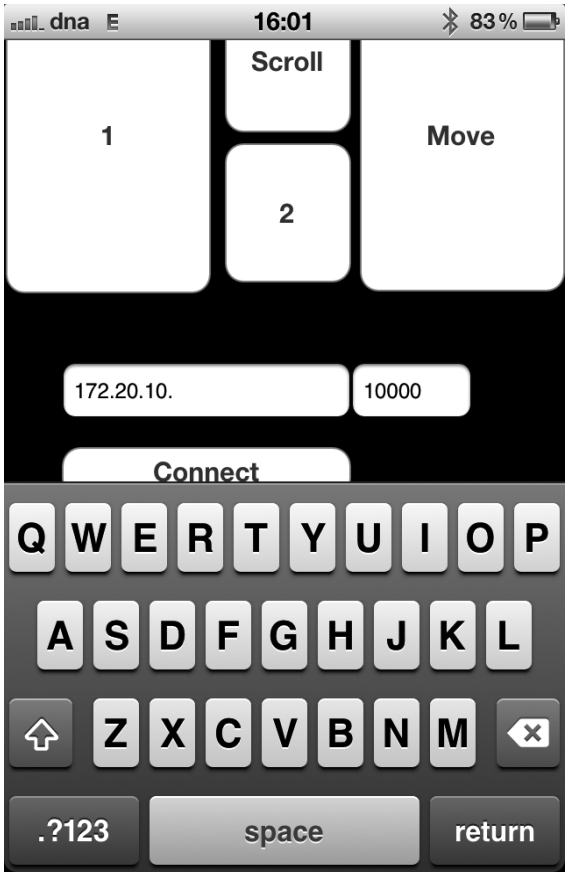
Ohjelman käyttöliittymä sisältää kaksi näkymää. Ensimmäinen näkymä on itse hiiren käyttöliittymä, joka on esitetty kuvassa 15. Toinen näkymä on asetusnäkymä, jonka avulla käyttäjä voi muuttaa hiiren toimintaan vaikuttavia asetuksia. Tämä näkymä on esitetty kuvassa 17. Käyttäjä voi vaihtaa näkymiä molemmissa kuvissa alhaalla näkyvän palkin avulla.



Kuva 15: iOS -hiirisovelluksen hiirikäyttöliittymänäkymä ennen ja jälkeen yhdistämisen.

Hiiren käyttöliittymänäkymä sisältää painikkeet, joita tietokonehiiren peruskäyttö edellyttää. Painikkeet 1 ja 2 ovat tietokonehiiren vasen ja oikea painike. Scroll -näppäin mahdollistaa sivun rullaamisen pysty- tai vaakasuunnassa. Move -painikkeen avulla liikuttaa tai pysäyttää kursori riippuen valituista asetuksista. Gyroskooppia hyödyntävän tilan ollessa päällä Move -napin tarkoitusta voidaan muuttaa siten, että sillä liikuttamisen sijaan pidetäänkin kursoria paikallaan. Kuvassa 15 on esitetty näkymä ennen yhteyden avaamista tietokonesovellukseen ja tämän jälkeen. Ennen yhteyden avaamista näkymässä on tekstikentät tietokoneen IP-osoitteelle ja tietokonesovelluksen

käyttämälle porttinumerolle. Kun nämä arvot on syötetty, niin yhteys voidaan avata Connect -painikkeella. Kun yhteys on onnistuneesti avattu näkymässä on yhteyden katkaisua varten Disconnect -nappi. Personal Hotspot ominaisuus tulee ennen hiiren käyttöä kytkeä päälle tietokoneen ja puhelimen välille tai laitteet tulee kytkeä samaan WiFi-verkkoon esimerkiksi saman tukiaseman kautta. Tietokoneen IP-osoite tulee selvittää ennen käyttöä esimerkiksi tietokoneen yhteysasetuksista, oli käytössä sitten Personal Hotspot tai langaton lähiverkkoyhteys erillisen tukiaseman kautta. Jos yhteys katkeaa kesken käytön sovellus ilmoittaa siitä punaisella ”Disconnected” -tekstillä. IP-osoitteen syöttäminen tekstikenttään ei ole kovin käyttäjäystävällinen tapa ja ennen kuin sovellus oli valmis esimerkiksi App Storeen, tämän tilalle kannattaisi kehittää jokin parempi ratkaisu. Jo pelkästään IP-osoitteen selvittäminen vie hetken ja sen tekeminen saattaa olla osaamattomalle käyttäjälle hankalaa.

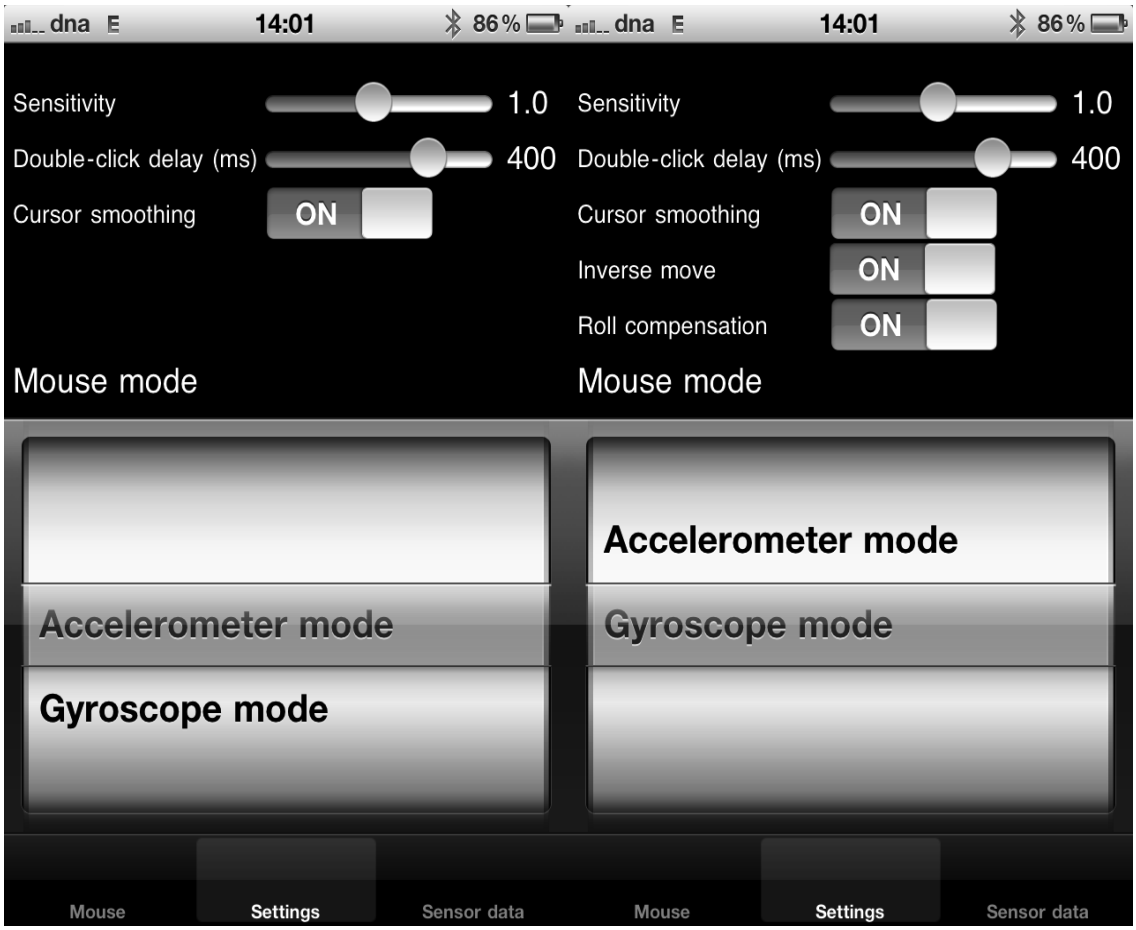


Kuva 16: Tekstikenttään kirjoittaminen nostaa kuvaa, jotta tekstikenttä on näkyvissä kirjoituksen ajan.

Kirjoittaminen tekstikenttiin tapahtuu käyttämällä iOS:n ohjelmistorajapinnan tarjoamaa virtuaalista näppäimistöä, joka näkyvissä ollessaan vie melkein puolet näytöstä. Tämän takia ruudulla näkyvää kuvaa nostetaan, jotta tekstikentät näkyvät kirjoittamisen aikana. Virtuaalinen näppäimistö sekä kuvan nostaminen ovat esitettyinä

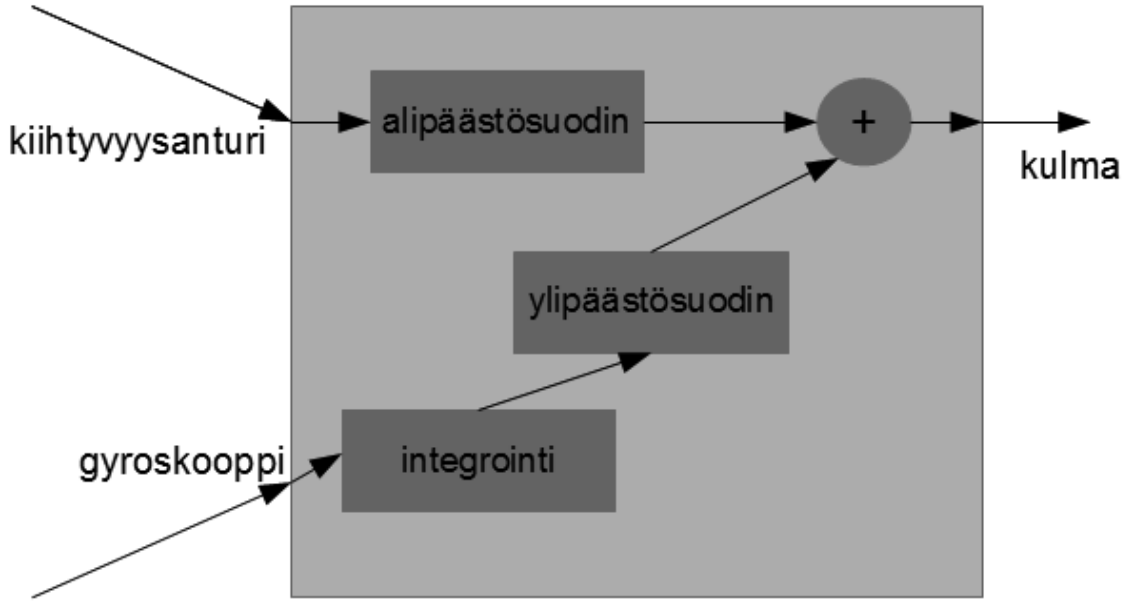
kuvassa 16.

Asetusnäkymä mahdollistaa hiiren toimintaan vaikuttavien asetuksien muuttamisen. Merkittävin näistä asetuksista on hiiren toimintatilan valinta. Toimintatilan voi valita ”Mouse mode” -tekstin alla näkyvästä valikosta. Asetusnäkymä on hieman erilainen näille kahdelle tilalle ja tämä ero näkyy kuvassa 17. ”Accelerometer mode” -tila tarkoittaa että kursorin liike muodostetaan pääosin kiihtyvyysanturin avulla, kun taas ”Gyroscope mode” -tila tarkoittaa että kursorin liike muodostetaan gyroskoopin avulla. Hiiren kursorin liikkeen herkkyyttä voidaan säätää ”Sensitivity” -liukusäätimellä. Jos käytössä on OS X -versio tietokonesovelluksesta, niin ”Double-click delay” -liukusäätimellä voidaan säätää tuplaklikkauksen klikkausten välillä olevaa maksimiviivettä. ”Cursor smoothing” kytkimellä voidaan kytkeä päälle tila, jonka päällä ollessa tietokonesovellus pyrkii pitämään kursorin liikkeen sulavana huonollakin yhteydellä. Jos gyroskooppia käyttävä tila on kytketty päälle, näkyy ruudulla lisäksi kytkimet ”Inverse move” ja ”Roll compensation”. Näistä ensimmäinen muuttaa ”Move” -napin ”Hold” -napiksi ja muuttaa hiiren toimintaa siten, että kursori liikkuu jatkuvasti ja pysähtyy kun ”Hold” -painike painetaan pohjaan. ”Roll compensation” -kytkin kytkee päälle tilan, jossa puhelimen sivuttaissuunnan kallistuskulma ei vaikuta kursorin liikkeeseen.



Kuva 17: iOS -sovelluksen asetusvälilehti molemmilla tiloilla.

3.2 Gyroskoopin ja kiihtyvyysanturin näytearvojen yhdistäminen



Kuva 18: Laitteen kallistuskulman suodatuksen periaatekuva. [16]

Työssä kokeillaan myös tapaa yhdistää kiihtyvyysanturin ja gyroskoopin näytearvot yhdeksi tarkemmaksi hiiren kallistuskulman näytearvoksi. Idea perustuu Shane Coltonin esittelemään suodatustapaan. Tässä yksinkertaisessa tavassa kiihtyvyysanturilta luettu näytearvo laitetaan alipäästösuotimen läpi ja gyroskoopilta integroitu arvo ylipäästösuotimen läpi ja näistä muodostetaan kallistuskulma. Alipäästösuotimen tarkoitus on suodattaa kiihtyvyysanturin arvoista pois lyhytkestoiset heilahdukset ja ylipäästösuotimen tarkoitus on suodattaa gyroskoopille ominaista vääristymää, joka kumuloituu merkittäväksi ongelmaksi erityisesti integroitaessa. Suodattimen idea on siis että kiihtyvyysanturin ja gyroskoopin näytearvot täydentävät toisiaan. Kuva 18 havainnollistaa tämän suodatuksen toimintaa. Näytearvo $angle_n$ ajan hetkellä n saadaan muodostettua seuraavasti:

$$angle_n = A \cdot (angle_{n-1} + gyroValue \cdot dt) + (1.0 - A) \cdot accValue \quad , \qquad (3)$$

missä gyroValue on gyroskoopilta luettava näytearvo, accValue on kiihtyvyysanturilta luettava näytearvo ja dt on peräkkäisten näytearvojen välinen aika. Kerroin A voidaan laskea seuraavasti:

$$A = \frac{\tau}{\tau + dt} \quad , \qquad (4)$$

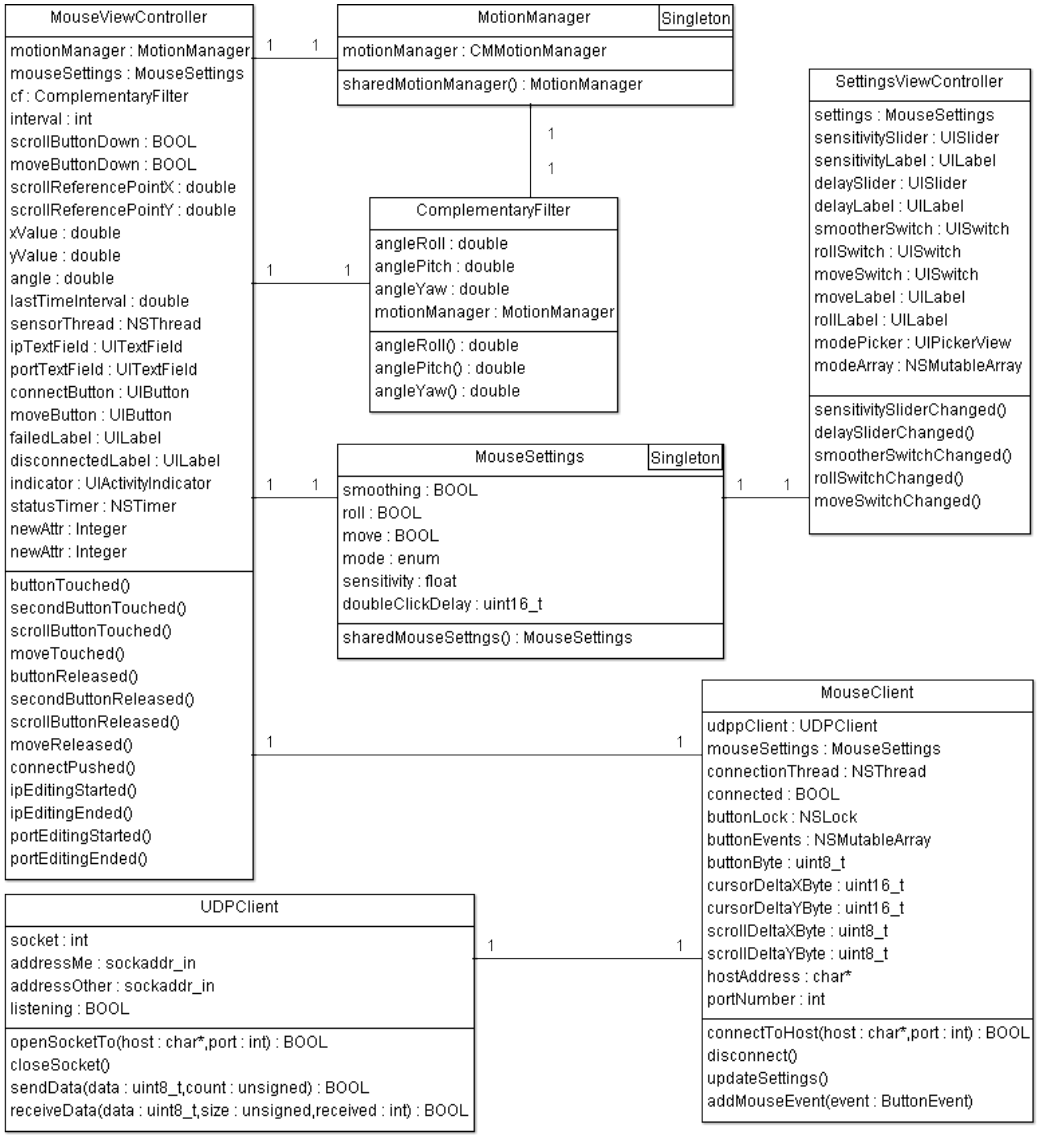
jos tiedetään aikavakio τ sekunteina ja näytteenottotaajuus. Aikavakio τ määrittää rajan kiihtyvyysanturin ja gyroskoopin välille. Kun on kyse rajaa pienemmistä ajanjaksoista luotetaan gyroskooppiin ja kun on kyse rajaa isommista ajanjaksoista, luotetaan kiihtyvyysanturiin.[16]

3.3 Sovelluksen rakenne

Apple on tarkoittanut, että iOS -sovellukset ohjelmoidaan MVC-ohjelmistoarkkitehtuurin mukaisesti ja tätä arkkitehtuuria pyrittiin käyttämään osittain myös tehdyssä mobiilisovelluksessa. Mobiilisovellus sisältää kaksi välilehteä, joista tuli MVC-arkkitehtuurin mukaiset näkymät. Nämä näkymät toteutettiin XCoden sisältämällä käyttöliittymätyökalulla. Nämä molemmat näkymät tarvitsivat käsittelijän. Nämä käsittelijät toteutettiin luokkina, jotka periytettiin *UIViewController* -luokasta. Näiden käsittelijöiden tarkoitus on ohjata näkymiä. Nämä kontrollerioliot ovat vastuussa laitteen näytöllä tapahtuvista muutoksista. Nämä käsittelijät sisältävät suurimman osan toteutetun mobiilisovelluksen toimintalogiikasta. Käsittelijöiden lisäksi sovellus sisältää joitakin suodatukseen, asetusten kuvaamiseen ja verkkoliikenteen toteuttamiseen liittyviä luokkia. Mobiilisovelluksen sisältänät luokat löytyvät kuvassa 19 esitetystä luokkakaaviosta.

Hiiren käyttöliittymänäkymää ohjaava käsittelijä on nimeltään *MouseViewController*. Tämä käsittelijä sisältää ohjelmalogiikan yhteyden avaamiseen ja hiiren tilan päivittämiseen. Kun näkymästä painetaan painiketta, näkymä kutsuu käsittelijään määriteltyä funktiota. Yhteyttä muodostaessa käyttäjä on ensin kirjoittanut IP-osoitteen ja portin käyttöliittymän tekstikenttiin ja tämän jälkeen hän painaa Connect -nappia. Tässä vaiheessa suoritetaan käsittelijäfunktio. Funktio lukee käyttöliittymän tekstikentistä IP -osoitteen ja portin ja tämän jälkeen pyytää verkkoliikenteestä vastuussa olevaa oliota avaamaan verkkoyhteyden. Käsittelijäfunktio päivittää käyttöliittymään tiedon onnistuiko tämä yhteyden avaus. Jos yhteyden avaus onnistui käsittelijäfunktio käynnistää säikeen, jonka tehtävänä on lukea inertia-anturien arvoja sekä painonappeja ja näiden perusteella lähettää hiiren tilapäivityksiä verkkoyhteyden ylitse. Kun hiiren käyttöliittymänäkymästä poistutaan tämä säie pysäytetään.

Asetusvälilehden näkymää ohjaava käsittelijä on nimeltään *SettingsViewController*. Se sisältää ohjelmalogiikan sovelluksen asetusten muuttamiseen. Sovelluksen asetuksia kuvaa luokka nimeltään *MouseSettings*. Käsittelijä muuttaa tämän luokan instanssin muuttujia. Kun asetusvälilehden näkymän asetuksia muutetaan, kutsutaan käsittelijään kirjoitettuja käsittelyfunktiota. Nämä funktiota lukevat uuden asetusarvon ja päivittävät sen *MouseSettings* -luokan instanssiin.



Kuva 19: iOS -sovelluksen luokkakaavio.

Sovelluksen verkkoliikenteestä huolehtivat *MouseClient* ja *UDPClient* nimiset luokat. Näistä *MouseClient* toimii korkeammalla tasolla ja se huolehtii että käytettyä kommunikointiprotokollaa noudatetaan. Luokka sisältää jäsenfunktiot esimerkiksi yhteyden avaukseen, asetuksien lähettämiseen ja napin painalluksen asettamiseen. Tämän lisäksi luokassa on jäsenmuuttujat kursorin paikan muutokselle, johon pääsee luokan ulkopuolelta käsiksi. Yhteyden avaus käynnistää säikeen, joka kuittauspaketin saatuaan lähettää verkkoyhteyden yli hiiren tilan päivityspaketin. *UDPClient* -luokka kapseloi sisäänsä socketeihin liittyvät toimenpiteet, kuten socketin avaamisen, sulkemisen, datan vastaanottamisen ja lähettämisen. *MouseClient* -luokka käyttää sen jäsenfunktioita hyväkseen verkkoyhteyden käyttöön.

Sovelluksen sisältämä *ComplementaryFilter* -luokka sisältää jäsenfunktiot, joilta voidaan pyytää laitteen kallistuskulma. Tämän kallistuskulman muodostamiseen on käytetty sekä kiihtyvyysanturin että gyroskoopin näytearvoja.

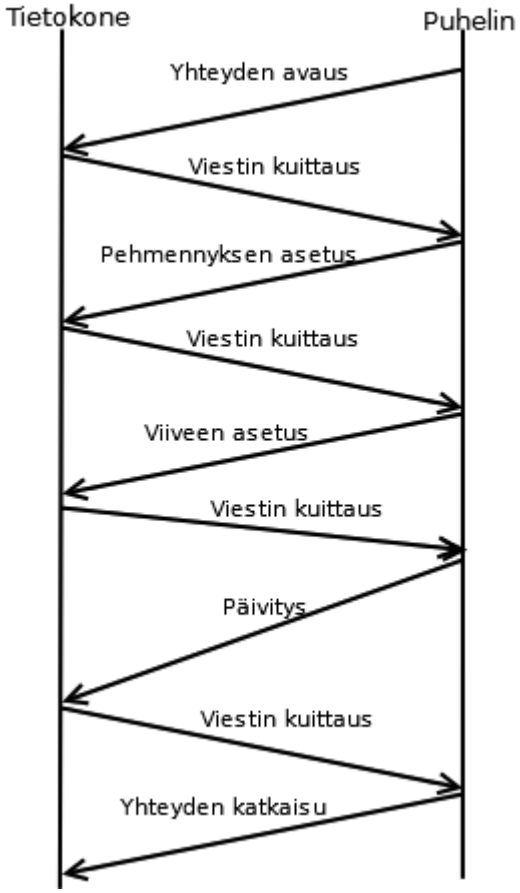
3.4 Kommunikointiprotokolla

Tehdyssä työssä mobiililaitteella ajettavan sovelluksen tulee lähettää jatkuvasti dataa tietokoneella ajettavalle sovellukselle. Laitteiden välille muodostetaan verkkoyhteys käyttämällä iPhonen Personal Hotspot -ominaisutta. Tämän avulla verkkoyhteys voidaan muodostaa USB:n, WiFi:n tai Bluetoothin ylitse. Käytettävästä mediasta riippumatta protokolla, jota molemmat laitteet noudattavat, on tehty IP/UDP -kuljetuskerroksen päälle. UDP on yhteydetön protokolla, jossa lähetettävien pakettien perillemenoa tai niiden oikeellisuutta ei taata millään tavalla. [21] Ensin esitellään työssä käytetty protokolla ja sen jälkeen on pohdintaa miten sitä voisi parantaa.

Työssä käytetty protokolla muodostuu kuudesta erilaisesta viestistä. Näiden viestien pituudet vaihtelevat yhdestä tavusta kahdeksaan tavuun. Protokollan kaikki viestit löytyvät kuvasta 22. Työssä mobiililaite on dataa lähettävä osapuoli ja tietokoneella ajettava sovellus vain kuittaa vastaanotetut viestit. Viestien lähetys on jatkuvaa ja, jos tietokoneella pyörivä sovellus ei vastaanota viestiä asetetun viiveen puitteissa, olettaa se yhteyden katkenneen. Jos mobiililaitteella pyörivä sovellus ei vastaanota kuittausta lähetetystä viestistä asetetun viiveen puitteissa, olettaa se yhteyden katkenneen. Jos yhteyden osapuolet havaitsevat virheen protokollan käytössä yhteys katkaistaan. Yhteyden avaamisen jälkeen iOS -sovellus voi lähettää viestejä missä tahansa järjestyksessä. Esimerkki yhteyden toiminnasta on kuvassa 20.

Yhteys avataan laitteiden välille mobiililaitteen lähettämällä kuuden tavun mittaisella viestillä, joka esittelee laitteen käyttämän protokollaversion. Tässä työssä versio on 1. Jos tietokonesovellus tukee kyseistä protokollaversiota, voi se hyväksyä yhteyden lähettämällä kuittausviestin. Kuittausviesti on tavun mittainen ASCII -kirjain 'A'. Kun yhteys on auki, mobiililaite voi katkaista yhteyden lähettämällä tavun mittaisen yhteysenkatkaisuviestin. Tämä viesti on ASCII -kirjain 'D'.

Yhteyden ollessa auki mobiililaite voi lähettää tietokonesovellukselle päivityksen hiiren tilasta, asettaa kaksoisnäpäytyksen viiveen sekä asettaa pehmennyksen päälle tai pois päältä. Kursorin liikettä pehmentävä asetus kytketään päälle lähettämällä kahden tavun mittainen viesti, jonka ensimmäinen tavu on ASCII -kirjain 'Z' ja toinen tavu on heksamuodossa joko 0x01 tai 0x00, joista ensimmäinen kytkee pehmennyksen päälle ja jälkimmäinen kytkee sen pois päältä. Kaksoisnäpäytyksen viive asetetaan lähettämällä kolmen tavun mittainen viesti, jonka ensimmäinen tavu on ASCII -kirjain 'Y'. Kaksi tavua muodostavat 16-bittisen kokonaisluvun, joka on viiveen pituus.



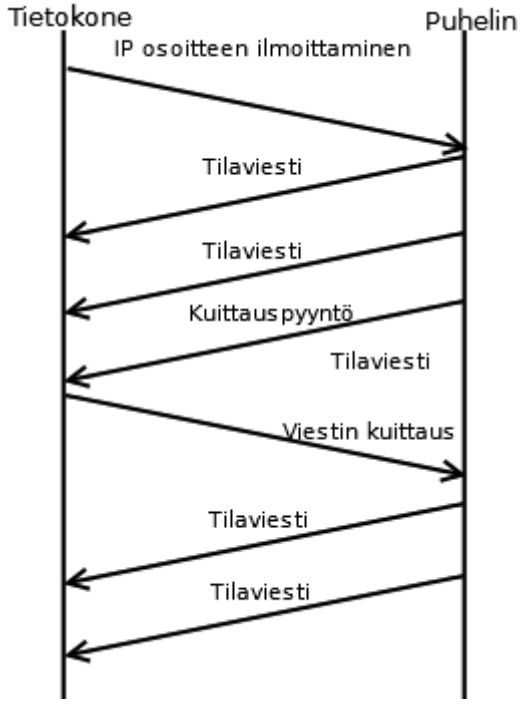
Kuva 20: Esimerkkikaavio protokollan toiminnasta.

Mobiililaite kertoo tietokonesovellukselle hiiren tilan kahdeksan tavun mittaisella viestillä, jonka ensimmäinen tavu on ASCII -kirjain 'A'. Toinen tavu 8-bittinen kenttä, joka sisältää nappien tilat. Toteutetussa työssä tästä kentästä on käytössä vain 2 bittiä. Tavut 3-6 sisältävät kaksi 16-kokonaislukua, jotka ovat hiiren kursorin X- ja Y-akselien liikenopeudet. Tavut 7 ja 8 sisältävät 8-bittiset kokonaisluvut, jotka ovat hiiren rullan liikenopeudet X- ja Y- suunnassa.

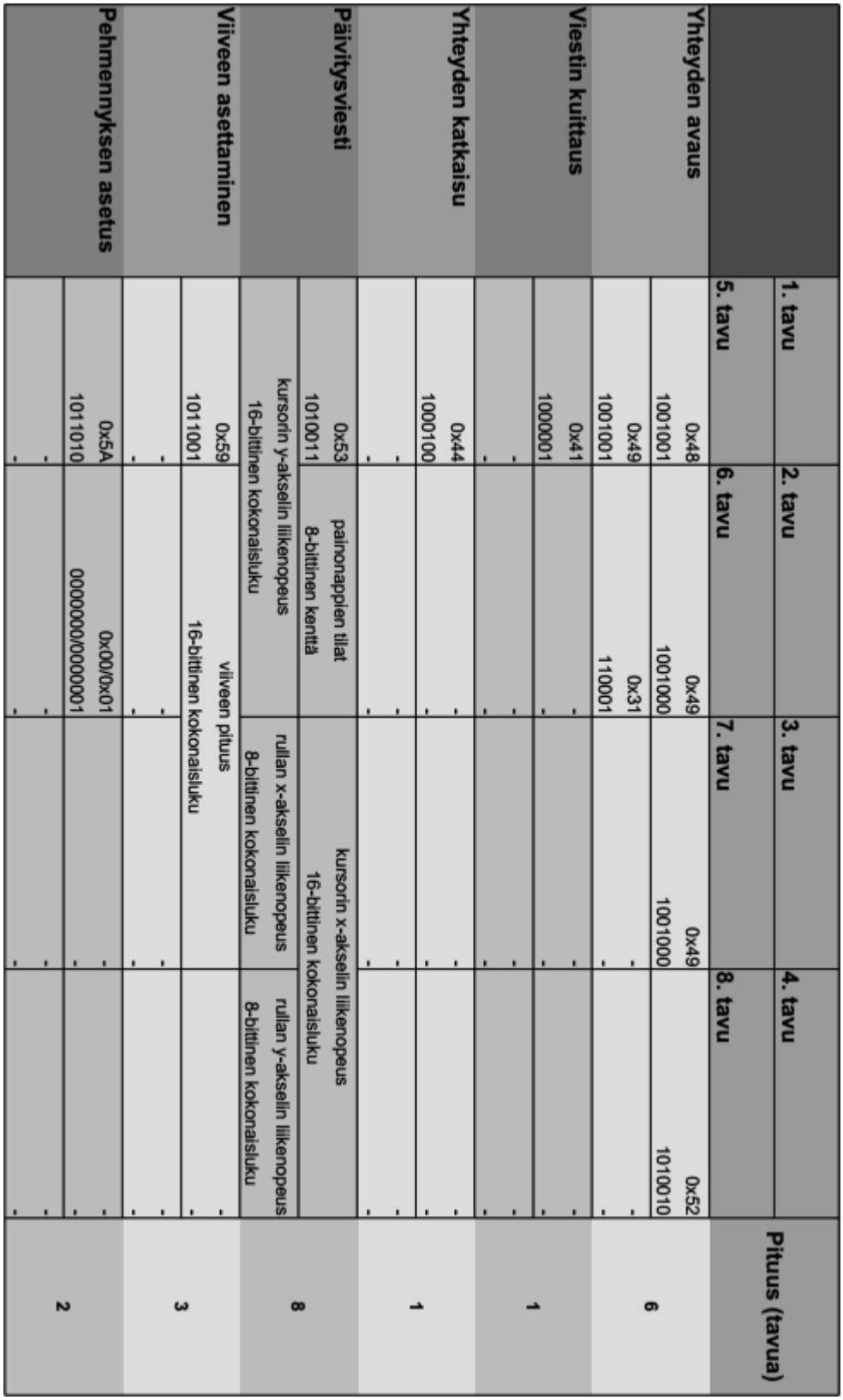
Vaikka edellä esitetty ja työssä käytetty protokolla toimikin testiympäristössä hyvin, ei se ole tarkoitukseen paras mahdollinen. Esimerkiksi jokaisen tilaviestin kuittaus erikseen ei välttämättä ole tarpeellista. Kuittauksen etuna on se, että viestintäkanava ei missään tilanteessa pääse ruuhkautumaan, koska uutta tilaviestiä ei lähetetä ennen kuin kuittaus on vastaanotettu. Tämän lisäksi yhteyden katkeaminen on helppo havaita kun kuittausviestiä ei tietyn ajan sisällä vastaanoteta. Ongelma tässä on se että yhdenkin kuittausviestin katoaminen tarkoittaa yhteyden katkeamista. Pakettien katoaminen on internet-liikenteessä varsin yleistä, mutta testiympäristössä sitä tapahtui harvoin. Tästä huolimatta tälläinen käyttäytyminen tulisi korjata ennen esimerkiksi sovelluksen kaupallistamista. Tässä tapauksessa yhteyden katkeamista voitaisiin tarkkailla

lähettämällä tasaisin väliajoin kuittauspyyntö tietokoneelle, johon sen tulisi vastata tietyn ajan sisällä tai yhteys tulkitaan katkenneeksi. Ruuhkautumisen estämiseksi voidaan tietokoneen ja mobiililaitteen välistä viivettä tarkkailla tämän kuittauspyynnön ja kuittauksen välisestä aikaerosta. Käytännössä tämän ei pitäisi muodostua ongelmaksi, koska taulukon 1 RTT-aikojen perusteella jokainen yhteysmenetelmä kykenee esimerkiksi hiirelle jo riittävään 100 Hz:n päivitystaajuuteen.

Protokollaa voisi myös yksinkertaistaa yhdistämällä viestityyppejä samaan pakettiin tilaviestin kanssa. Yhteyden avauksen voisi periaatteessa pudottaa pois ja protokollan version voisi lisätä tilapäivitysviestiin. Yhteyden katkaisun voisi lisätä tilanpäivitysviestiin yhden bitin lippuna. Nämä muutokset eivät kasvattaisi tilanpäivitysviestin koko juurikaan. Protokollaan olisi tosin kätevä lisätä viesti, jolla tietokonesovellus voisi ilmoittaa ethernet aliverkon broadcast-osoitteeseen lähettämällä oman IP-osoitteensa mobiililaitteelle. Tällä tavoin voitaisiin kiertää ongelma että käyttäjä joutuu kirjoittamaan tietokoneen IP-osoitteen mobiililaitteeseen.



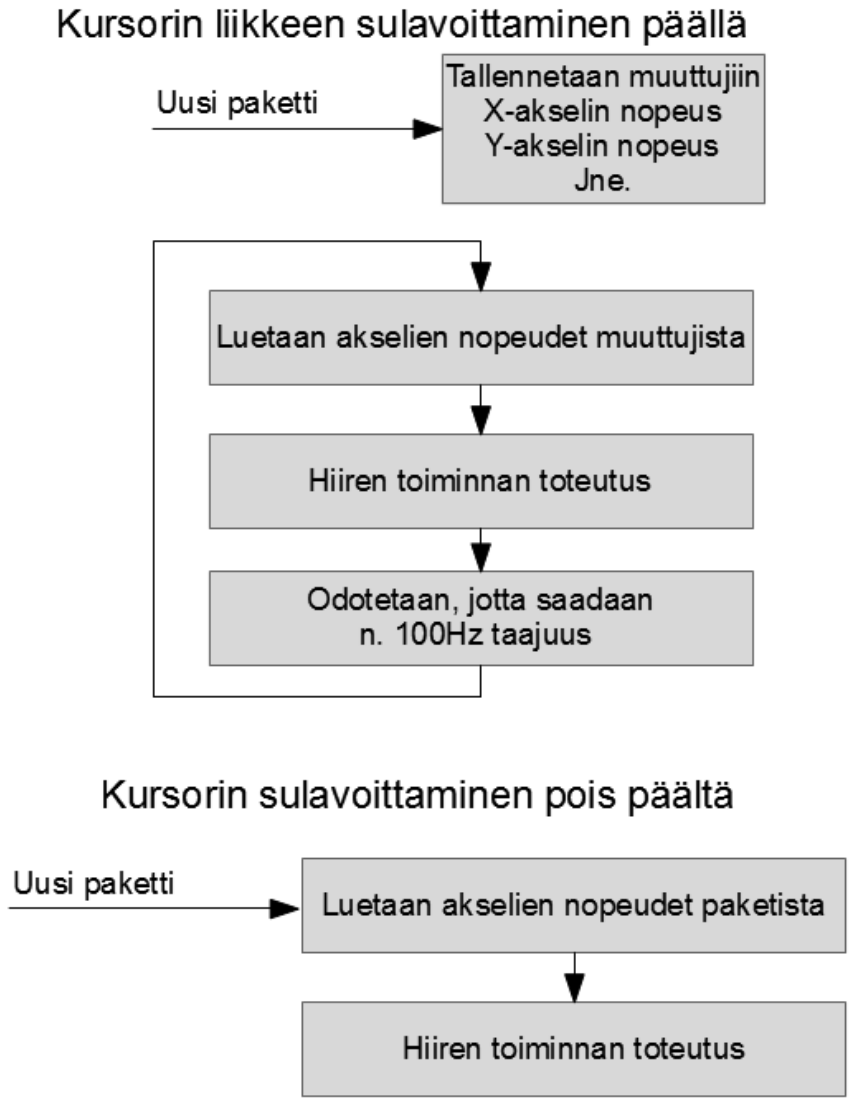
Kuva 21: Esimerkkikaavio siitä kuinka paranneltu kommunikointiprotokolla voisi toimia.



	1. tavu	2. tavu	3. tavu	4. tavu	Pituus (tavua)
	5. tavu	6. tavu	7. tavu	8. tavu	
Yhteyden avaus	0x48 1001001	0x49 1001000	0x49 1001000	0x52 1010010	6
	0x49 1001001	0x31 110001	-	-	
Viestin kuittaus	0x41 1000001	-	-	-	1
Yhteyden katkaisu	0x44 1000100	-	-	-	1
Päivitysviesti	0x53 1010011	painonappien tilat 8-bittinen kenttä	kursorin x-akselin liikenopeus 16-bittinen kokonaisluku		8
	kursorin y-akselin liikenopeus 16-bittinen kokonaisluku		rullan x-akselin liikenopeus 8-bittinen kokonaisluku	rullan y-akselin liikenopeus 8-bittinen kokonaisluku	
Viiveen asettaminen	0x59 1011001	viiveen pituus 16-bittinen kokonaisluku		-	3
	-		-	-	
Pehmennyksen asetus	0x5A 1011010	0x00/0x01 00000000/00000001	-	-	2
	-		-	-	

Kuva 22: Kommunikointiprotokollan viestityypit.

4 TIETOKONESOVELLUS



Kuva 23: Kursorin sulavoittamisen periaate.

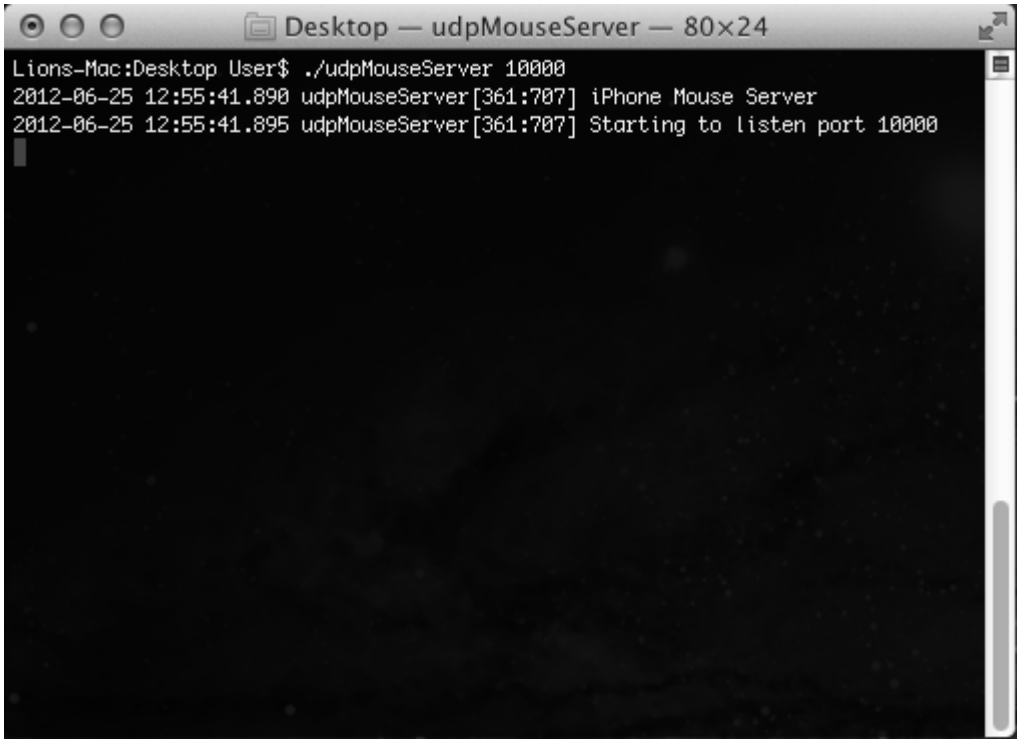
Tietokoneella ajettavan sovelluksen tehtävä on vastaanottaa aika ajoin mobiililaitteelta tulevia hiiren tilaa muuttavia päivityspaketteja. Näiden pakettien rakenne on tarkemmin kuvattu kappaleessa 3.4. Vastaanotettuaa paketin, sovellus suorittaa käyttöjärjestelmän rajapintaa käyttäen päivityspaketin edellyttämät toiminnot. Tälläisiä toimintoja ovat

esimerkiksi hiiren kursorin liikutus, hiiren nappien painaminen ja hiiren rullan liikutus. Ohjelman tehtävä on siis emuloida hiiren toimintaa.

Kuvassa 23 on esitetty periaate, jolla kursorin liike saadaan vaikuttamaan sulavammalta huonolla yhteydellä. Kun tila kytketään päälle hiiren tilan päivitys tehdään tasaisesti 100hz taajuudella, riippumatta siitä saadaanko uusi paketti vai ei.

Tietokonesovellus toimii erillään mobiililaitteella ajettavasta sovelluksesta ja niiden välinen kommunikointi tapahtuu IP-verkkoyhteyden ylitse. Tämä tarkoittaa että tuki tehdylle hiirisovellukselle on helpohko toteuttaa usealle käyttöjärjestelmälle. Riittää että käyttöjärjestelmän ohjelmointirajapinta mahdollistaa ohjelmalliset hiiritoiminnot ja mobiililaitteen ja käyttöjärjestelmän välille saadaan muodostettua IP-verkkoyhteys. Olen toteuttanut komentorivipohjaiset tietokonesovellukset Windows- ja OS X-käyttöjärjestelmille.

4.1 OS X -versio



Kuva 24: Komentorivipohjainen OS X -versio tietokonesovelluksesta.

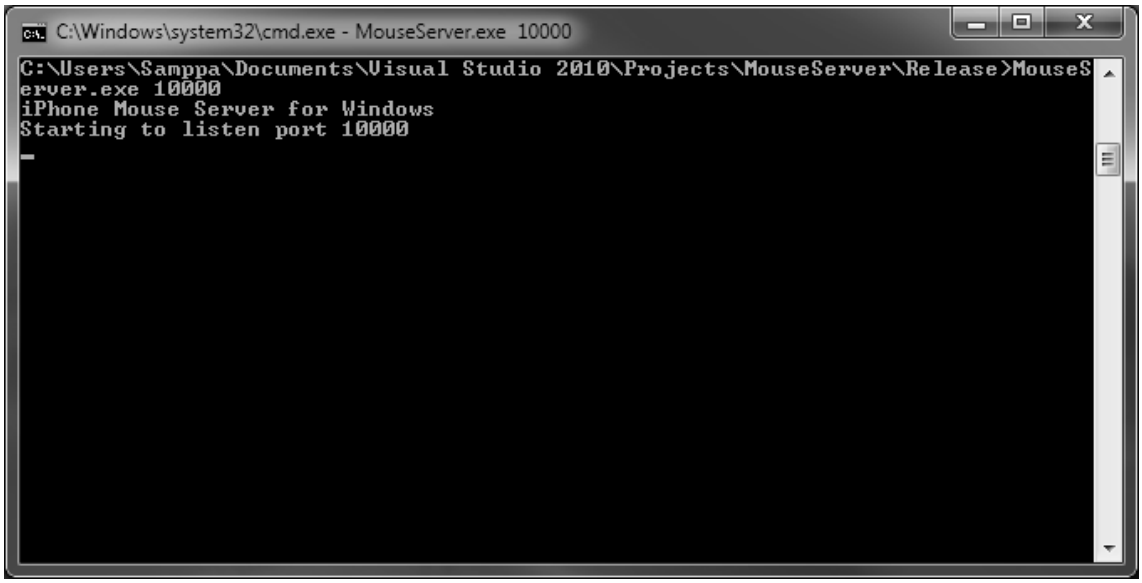
OS X -versio tietokonesovelluksesta on tehty Applen suosimalla Objective-C kielellä käyttäen Applen OS X -kehitykseen tarjoamia rajapintoja ja kirjastoja, jotka sisältyvät oletuksena XCode-kehitystyökaluun. Tässä työssä käytetyn XCode-ohjelman versio oli 4.3.2. OS X -versio sovelluksesta tehtiin ennen Windows -versiota ja ohjelmakoodia kertyi n.1100 riviä.

Ohjelma on käytännössä palvelin ja se käynnistetään antamalla parametrina kuunneltavan portin numero. Kuva 24 esittää käynnistettyä sovellusta. Heti käynnistyksen jälkeen sovellus alkaa kuunnella annettua porttia ja vastaanottamaan UDP-paketteja. Datan vastaanottamiseen käytetään OS X -käyttöjärjestelmän socket-rajapintaa [17]. Ohjelma rakentuu pääohjelmafunktiosta ja kolmesta luokasta. Rakenteeltaan identtisen Windows-version luokkakaavio löytyy kuvasta 27. *DataHandler* -luokka hoitaa protokollan parsimisen ja hiiritoimintojen toteuttamisen. *UDPServer* -luokka kapseloi sisäänsä käyttöjärjestelmän socket-rajapinnan käyttämisen UDP-palvemen toteuttamiseen ja *RingBuffer* -luokka on rengaspuskuri, jota käytetään socketista luetun datan tallentamiseen. Pääohjelmafunktio jää socketin avattuaan odottamaan UDP-pakettia. Vastaanotettuaan paketin ohjelma luovuttaa paketin käsittelijäoliolle, joka tarkistaa paketin oikeellisuuden ja suorittaa sen edellyttämän hiiritoimenpiteet. Jos paketti ei ole protokollan mukainen yhteys katkaistaan. Kuvaus käytetystä protokollasta löytyyy kappaleesta 3.4. Jos paketti on protokollan mukainen vastataan siihen kuittauspaketilla ja suoritetaan paketin edellyttämät toimenpiteet. Toimenpiteitä voivat olla kursorin tilan päivitys tai asetuksen muuttaminen. Ensimmäisen paketin vastaanottamisen jälkeen asetetaan sockettiin aikakatkaisuarvo, jonka avulla huomataan jos paketteja ei enään jostain syystä vastaanoteta. Jos uutta tilapäivityspakettia ei vastaanoteta 2 sekunnin sisällä yhteys tulkitaan katkenneeksi. Vuokaavio ohjelman toiminnasta löytyy kuvasta 26.

Ohjelma tukee kahta erilaista tapaa päivittää hiiren tilaa. Näistä toinen suorittaa päivityksen aina kun uusi paketti vastaanotetaan ja toinen suorittaa päivityksen aina n. 100 Hz taajuudella. Ensimmäisen tavan sulavuus riippuu täysin siitä kuinka tasainen on vastaanotettujen tilapäivityspakettien välinen aikaero ja millä taajuudella uusia tilapäivityspaketteja vastaanotetaan ja toinen tapa vaikuttaa sulavalta vaikka kahden paketin aikaero kasvaisikin hetkellisesti suureksi. Liikkeen tarkkuus tosin saattaa kärsiä, koska hiiren tilan uusi päivitys tehdään aina viimeksi saapuneen paketin tietojen perusteella. Käytössä oleva kommunikointiprotokolla sisältää viestityypin, jonka avulla voidaan valita toinen näistä tiloista käyttöön. Hiiren toiminnot suoritetaan käyttämällä Applen tarjoamaa Quartz Event -rajapintaa. Rajapinta tarjoaa funktion *CGEventCreateMouseEvent*, jolla voidaan luoda hiiritapahtumia ja funktion *CGEventPost*, jolla voidaan tämä tapahtuma suorittaa. [18]

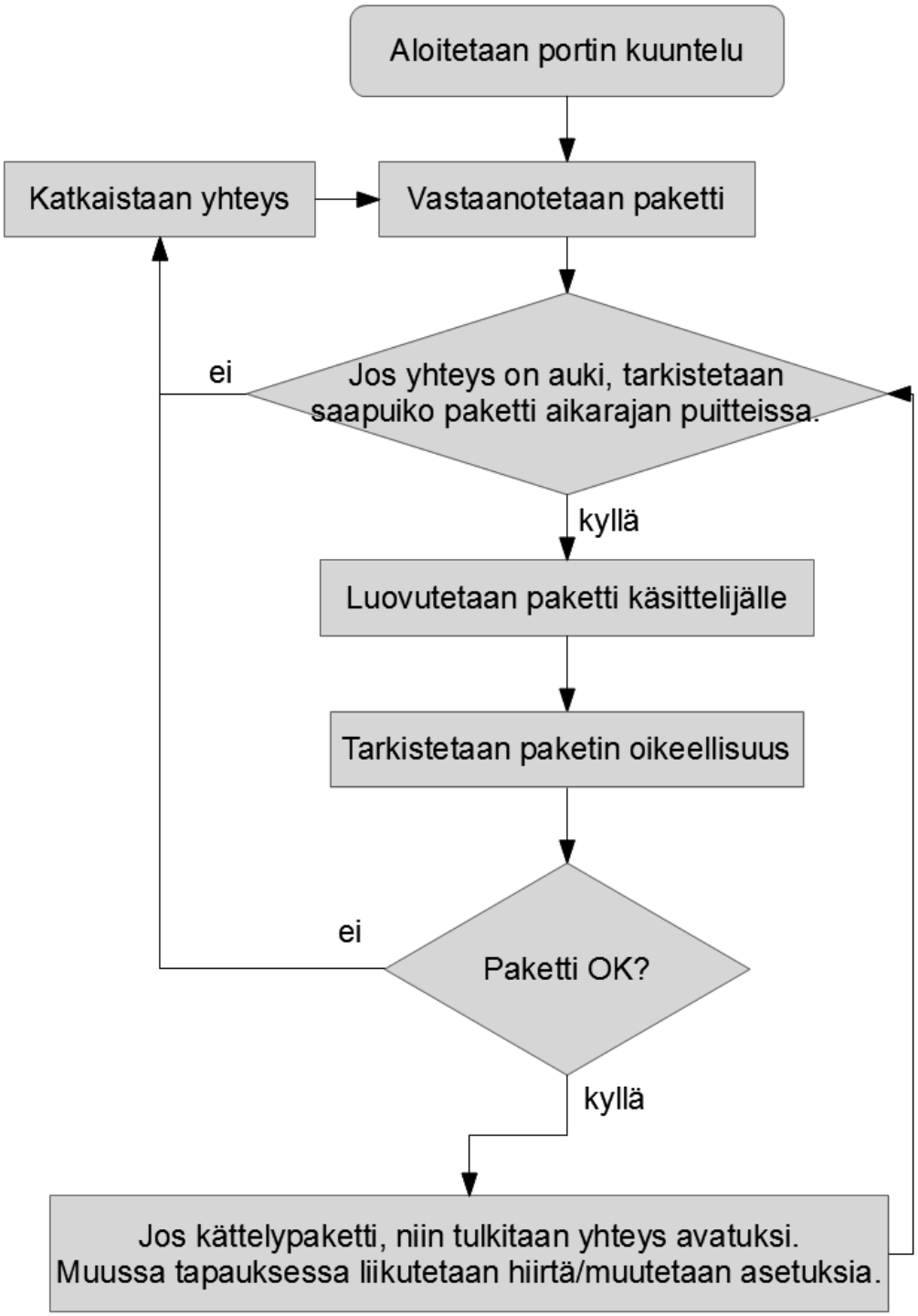
4.2 Windows -versio

Windows -versio tietokonesovelluksesta on tehty C++ -kielellä käyttäen Microsoftin tarjoamia käyttöjärjestelmän rajapintoja. Sovellus tehtiin käyttäen Microsoftin tarjoamaa Visual Studio 2010 Express -kehitystyökalua.

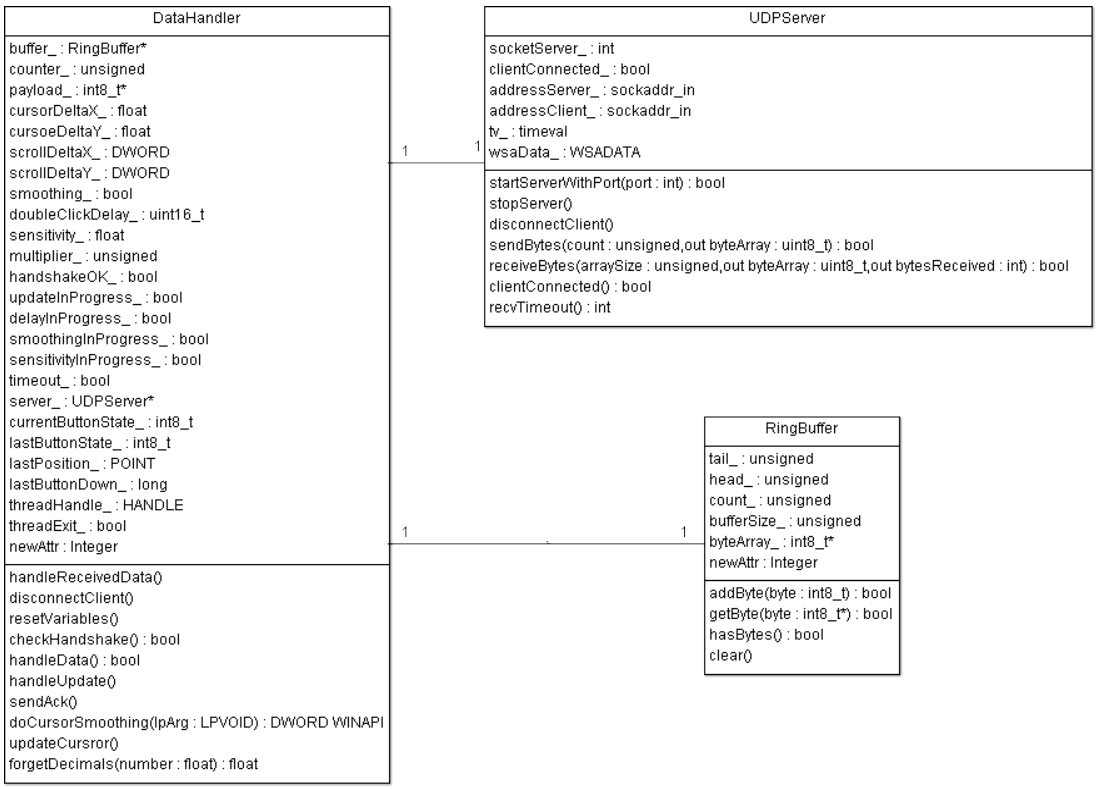


Kuva 25: Komentorivipohjainen Windows -versio tietokonesovelluksesta.

Windows -versio sovelluksesta tehtiin OS X -version jälkeen ja, koska sekä Objective-C että C++ pohjautuvat C-kieleen, oli ohjelman muuttaminen kielestä toiseen aika suoraviivaista. Ohjelmakoodia Windows -versioon kertyi n. 900 riviä. Ohjelman rakenne on identtinen OS X -version kanssa. Tietokonesovelluksen Windows-version luokkakaavio löytyy kuvasta 27. Käyttöjärjestelmäspesifisten asioiden tekemiseen käytettiin Windowsin rajapintoja. Datan vastaanottoon ja hiiren toimintojen suorittamiseen käytettiin Windowsin tarjoamia socket ja mouse_event rajapintoja [19] [20]. Windows-versio ohjelmasta ei toimi sellaisten ohjelmien kanssa, joita käytetään järjestelmävalvojan oikeuksilla.



Kuva 26: Vuokaavio tietokonesovelluksen toiminnasta.



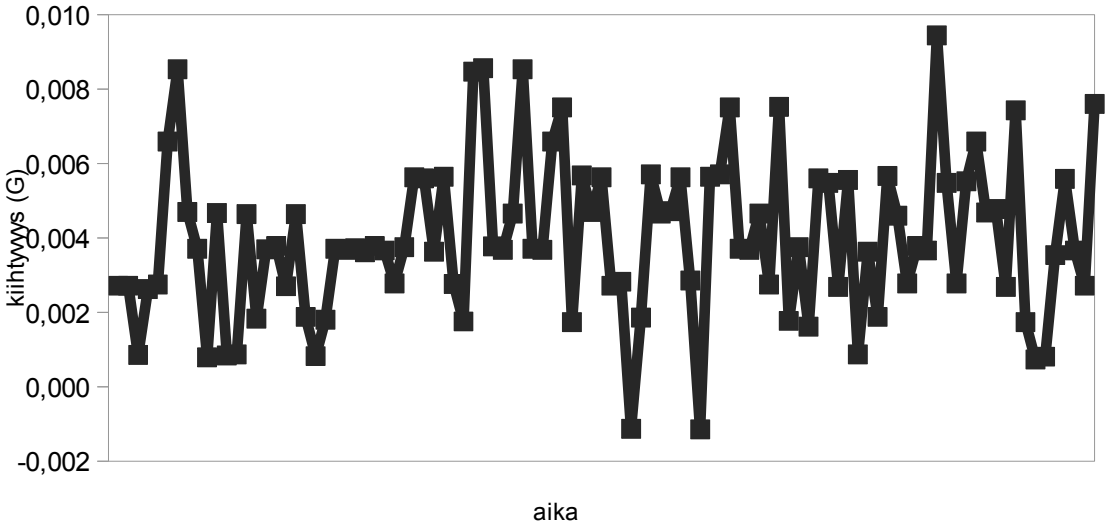
Kuva 27: Windows-sovelluksen luokkakaavio

5 MITTAUKSET JA TULOKSET

Kappaleessa käydään läpi mobiilisovellukselle ja käytössä olleelle mobiililaitteelle tehtyjä mittauksia. Ensin tutkitaan mobiililaitteen antureita ja miten antureilta saatuja näytearvoja käsiteltiin. Tämän jälkeen tutkitaan minkälaisia viiveitä erilaiset yhteysmenetelmät tietokoneen ja mobiililaitteen välillä aiheuttavat testiympäristössä.

5.1 Puhelimen anturit

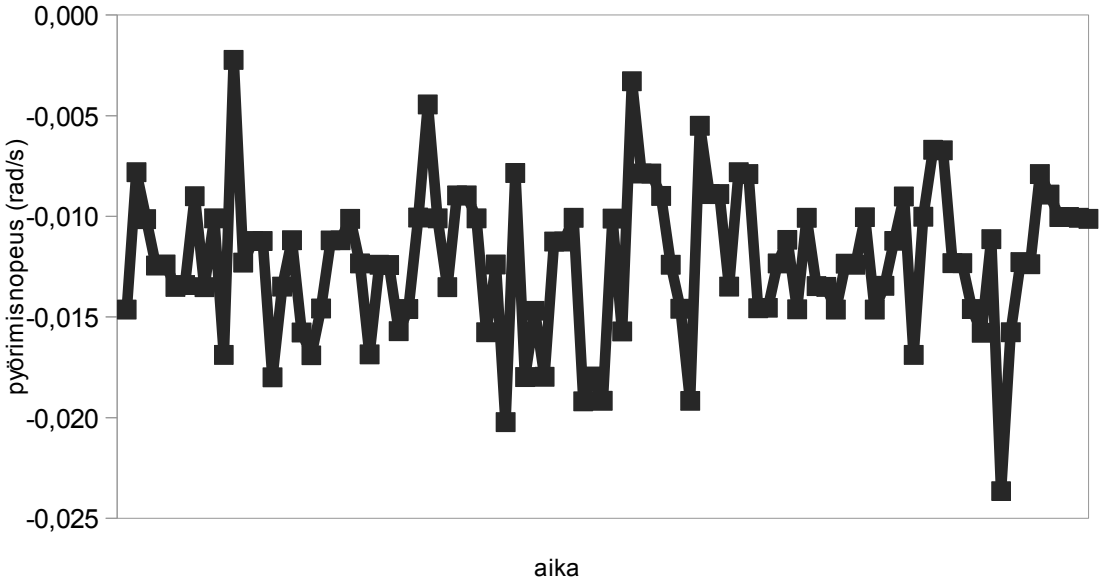
Käytössä olleesta iPhone 4 -mobiililaitteesta löytyy gyroskooppi ja kiihtyvyysanturi, joita molempia käytettiin toteutetussa sovelluksessa. Anturien tuottamien näytteiden laatu on oleellinen seikka niitä käyttävän sovelluksen toiminnalle, minkä takia on mielenkiintoista selvittää minkä laatuista signaalia antureilta saadaan. Tehdyssä sovelluksessa antureilta luetaan uusia arvoja n. 100 Hz taajuudella.



Kuva 28: Kiihtyvyysanturin Y-akselin tuottamia näytteisarvoja 100 hertsin taajuudella. Laite oli mittaushetkellä paikallaan pöydällä.

Kuvissa 28 ja 29 on esitetty antureilta luetut raakaluvut sekä kiihtyvyysanturilta että gyroskoopilta kun mobiililaite oli asetettu paikalleen pöydälle. Ideaalitapauksessa molempien arvojen pitäisi pysyä nollassa koko mittausajan, mutta näin ei kuitenkaan

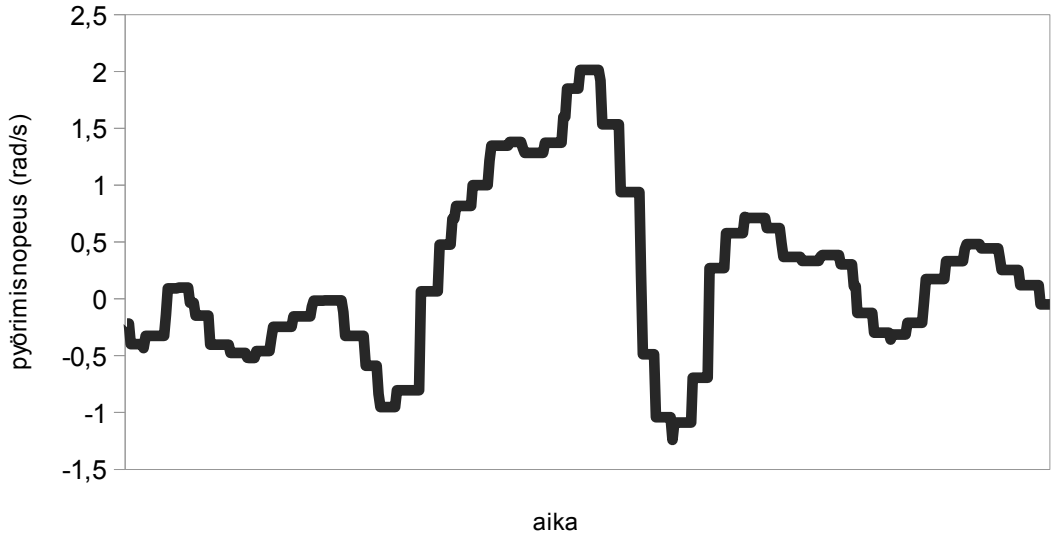
ollute vaan kuvaajissa ilmenee heilahteluja. Kuvaajista havaitaan myös poikkeama, jonka takia nollakohta ei ole aivan nollassa, vaan poikkeaa siitä hieman.



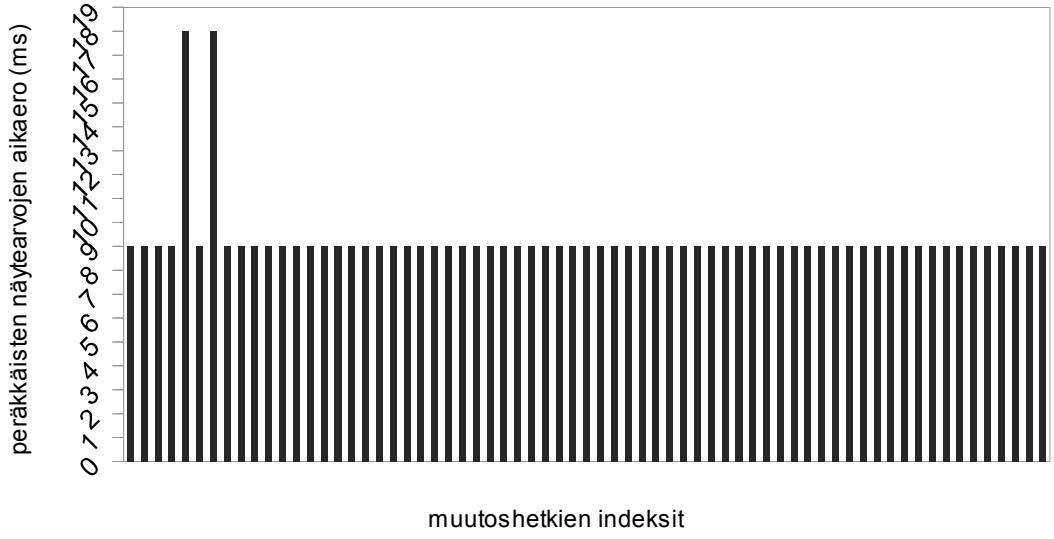
Kuva 29: Gyroskoopin Y-akselin tuottamia näytteisarvoja 100 Hz:n taajuudella. Laite oli mittaushetkellä paikallaan pöydällä.

Näytearvojen laadun lisäksi on mielenkiintoista tarkastella kuinka nopeasti antureilta voidaan lukea uusia arvoja. Näytearvojen lukeminen voidaan tehdä ohjelmakoodissa siten, että luodaan CMMotionManager -olio ja luetaan anturin arvoja sen sisältämistä muuttujista. Toinen tapa on luoda käsittelijäfunktio ja antaa sen viite CMMotionManager -oliolle. Tässä tapauksessa olio kutsuu käsittelijäfunktiota asetetuin väliajoin antamalla anturien arvot funktion parametreina. [14]

Kuvassa 30 on luettu gyroskoopilta näytearvoja 1000 Hz:n taajuudella. Kyseisessä kuvassa on selvästi näkyvissä porrastumista, joka johtuu siitä että luettu arvo pysyy samana usean lukukerran ajan. Tästä voidaan päätellä ollaan ylitetty taajuus, jolla ohjelmistorajapinta suostuu maksimissaan antamaan uusia näytearvoja. Jokaiseen näytearvoon on liitetty aikaleima, jonka avulla voidaan tutkia kuinka kauan näyte pysyy samana. Kuvassa 31 on esitetty peräkkäisten näytearvojen aikaerot. Keskimääräinen ero oli n. 9 ms ja tästä voidaan päätellä että gyroskoopin lukemisen maksimitaajuus on n. 110 Hz. Kuvaajasta näkee lisäksi, että parissa kohtaa jää tuntemattomasta syystä yksi näyte puuttumaan Syy tälläiselle käyttäytymiselle saattaa olla esimerkiksi se, että käyttöjärjestelmä joutuu poistamaan kolmannen osapuolen sovelluksen ajosta hetkeksi jonkin tärkeämmän prosessin vaatiessa enemmän prosessoriaikaa.



Kuva 30: Gyroskoopin X-akselilta luettuja näytearvoja n. 1000 hertsin taajuudella.

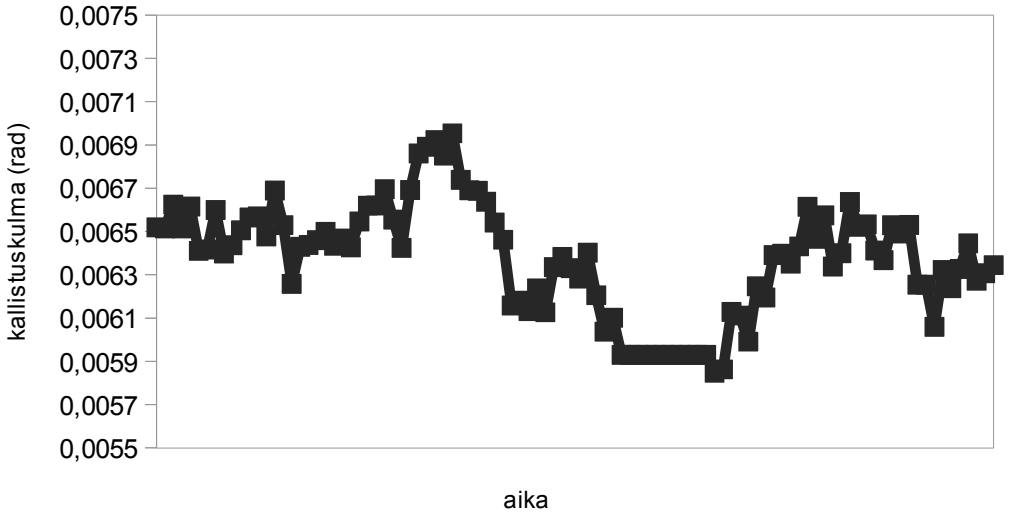


Kuva 31: Peräkkäisten näytearvojen aikaero. Näytearvot on otettu 1000 Hz:n taajuudella ja samat näytteet on poistettu.

5.2 Anturien näytearvojen käsittely

Antureilta luettavissa olevat arvot ovat kiihtyvyysanturin tapauksessa G-voimana ja gyroskoopin tapauksessa radiaaneina sekunnissa. Koska tehdyn sovelluksen on tarkoitus liikuttaa kursoria tietokoneen ruudulla, niin anturien näytearvot tulkitaan X- ja Y-akselien muutoksina pikseleissä. Kun laite on paikallaan pöydällä kursorin ei tietenkään tulisi liikkua ruudulla ja tämän takia näytearvoista tulisi saada häiriöt pois. Lisäksi olisi hyvä saada aikasemmin huomattu nollakohdan vääristyminen pois. Kiihtysvyysanturin tapauksessa vääristymä ei ole niin oleellinen, koska käyttäjä tuskin huomaa pientä kallistumaa nollakohdassa. Gyroskoopin tapauksessa ongelma on merkittävämpi, koska vääristymä aiheuttaa kursorin liikkumisen vaikka laite olisikin paikallaan.

Yksinkertainen ratkaisu ongelmaan on näytearvojen tarkkuuden pienentäminen. Tätä keinoa käytetään työssä gyroskoopin näytearvojen käsittelyssä. Lisäksi käytössä on ohjelmallinen alipäästösuodin. Gyroskoopilta luettu desimaaliluku muutetaan kokonaisluvuksi kertomalla se sopivalla vakiolla. Vakio on valittu siten että anturin ollessa paikallaan pöydällä kokonaisluvut pysyvät nollassa. Kokeilemalla sopivaksi vakioksi löytyi 20.

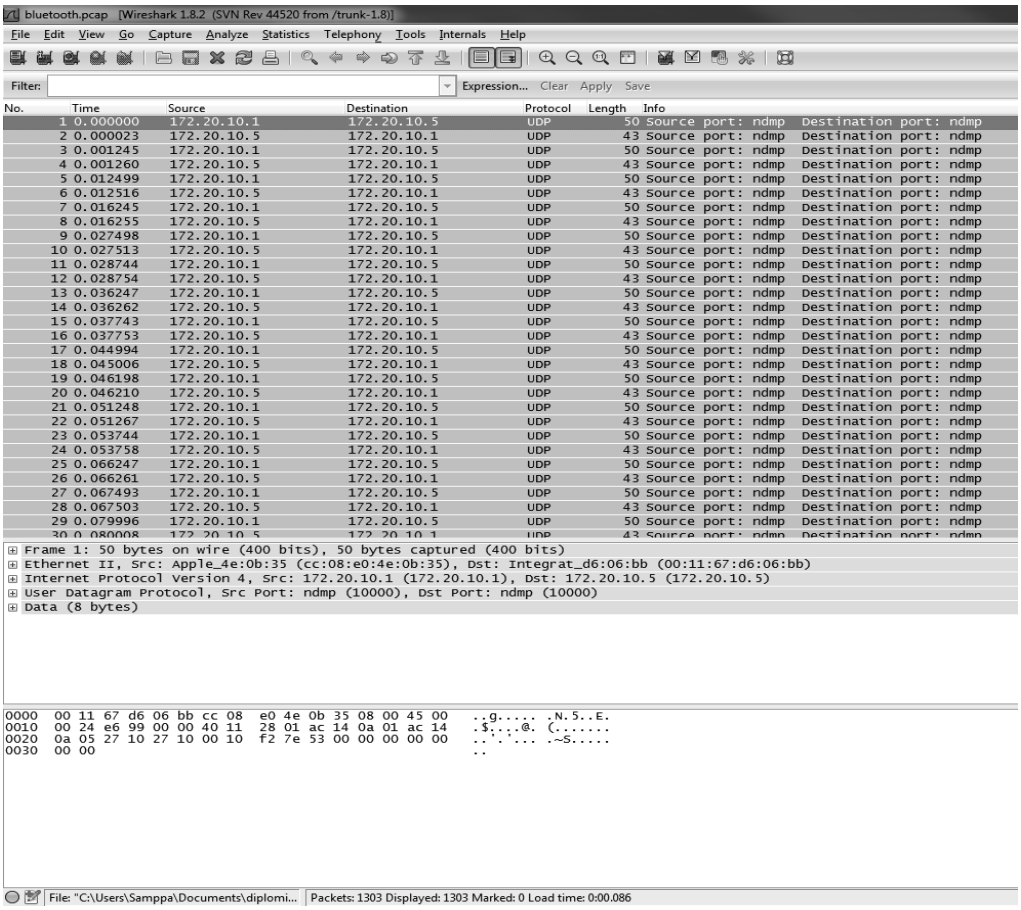


Kuva 32: Kiihtyvyysanturin ja gyroskoopin näytearvot yhdistetty Y-akselin kallistuskulman arvoksi. Mittaushetkellä laite oli paikallaan pöydällä.

Kiihtyvyysanturin tapauksessa kallistuskulman tarkkuuden parantamiseksi käytetään avuksi gyroskoopin tuottamia arvoja. Gyroskoopin tuottama pyörimisnopeus voidaan muuttaa kallistuskulmaksi integroimalla. Ongelmaksi tässä muodostuu vääristymä gyroskoopin nollakohdassa. Tämä vääristymä aiheuttaa kulmaan muutoksen vaikka

anturi olisi paikallaan. Kallistuskulman lopullinen arvo tuotetaan yhdistämällä kiihtyvyysanturin näytearvo ja gyroskoopin näytearvoista integroitu luku kaavan 3 esittämällä tavalla. Ennen summausta kiihtyvyysanturin ulostulo alipäästösuodatetaan ja integroinnin tulos ylipäästösuodatetaan. Tällä tavalla saadaan vähennettyä kiihtyvyysanturin näytearvojen heilahteluja ja gyroskoopin vääristymää.[16] Kuvassa 32 on esitetty tällä tavalla tuotetut kallistuskulman näytearvojen kuvaaja, kun laite oli paikallaan pöydällä. Kaavan 4 τ on tässä tapauksessa 0.1 s. Kuvasta nähdään, kun verrattaan kuvaan 28, että näytearvojen laatu paranee huomattavasti verrattuna suoraan kiihtyvyysantureilta luettuihin näytearvoihin. Tämän ansiosta voidaan käyttää isompaa vakiota kertojana, kun desimaaliluku skaalataan kokonaisluvuksi. Tämä tarkoittaa käytännössä parempaa tarkkuutta. Sopivaksi vakioksi löytyi kokeilemalla 200.

5.3 Yhteyden viiveet



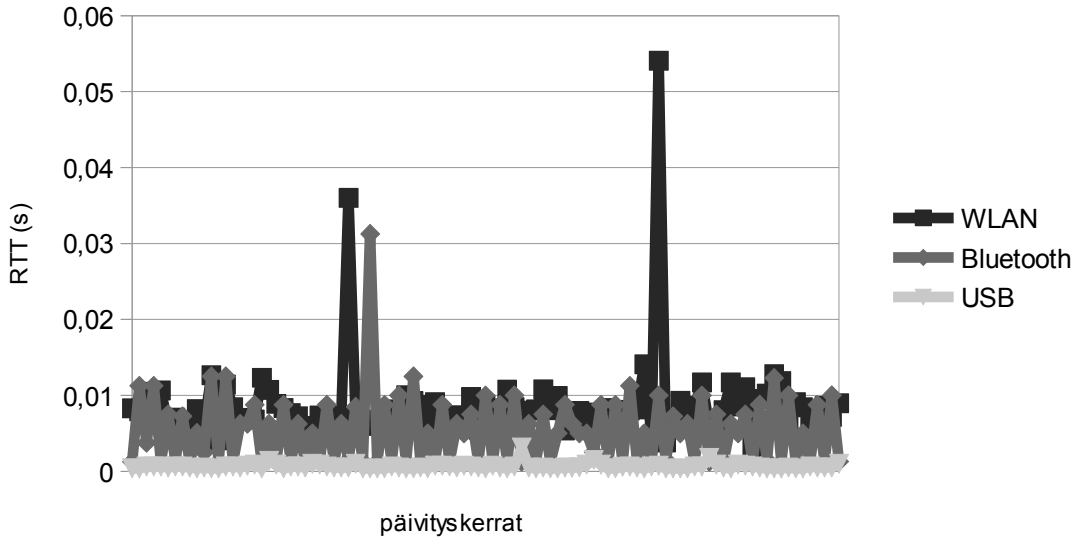
Kuva 33: Kuvankaappaus Wireshark protokolla-analysaattorisovelluksesta.

Mobiilisovellus keskustelee tietokonesovelluksen kanssa UDP-protokollaa käyttäen.

Verkkoyhteys voidaan muodostaa tietokoneen ja iPhone 4 -mobiililaitteen välille usealla tavalla, jolloin kommunikointi laitteiden välillä onnistuu. Erilaisia yhteystapoja on WiFi, Bluetooth ja USB. Näistä kaksi ensimmäistä ovat langattomia ja viimeinen tarvitsee mobiililaitteen ja tietokoneen välille kaapelin. Mobiililaitteen ja tietokoneen välillä ei määrällisesti liiku hirveästi dataa, joten yhteyden kaistanleveys ei ole hiiren suorituskyvyn kannalta kovin oleellinen seikka, mutta yhteyden viiveellä on merkittävä vaikutus hiirisovelluksen käyttökokemuksen sulavuuteen. Mitä kauemmin joudutaan odottamaan kuittausta edellisestä päivityspaketista, sitä pidemmäksi kasvaa kahden päivityspaketin välinen aika. Tämän takia on mielenkiintoista tutkia erilaisten yhteysmahdollisuuksien viiveitä. Viiveiden selvittämisessä käytettiin Wireshark protokolla-analysaattoria. Wireshark on sovellus, jonka avulla voidaan kuunnella ja analysoida tietokoneen verkkosovittimien kautta tapahtuvaa liikennöintiä. Kuva Wiresharkin käyttöliittymästä löytyy kuvasta 33. Taulukossa 1 on esitetty eri yhteysmenetelmien RTT(Round Trip Time) -aikojen keskiarvo. Mittaukset suoritettiin iPhone 4 -laitteella, jonka ohjelmistoversio oli 4.3.5.

	WLAN	Bluetooth	USB
RTT (ms)	8,7	5,3	0,6
Päivitystaajuus (Hz)	114,9	188,7	1666,7

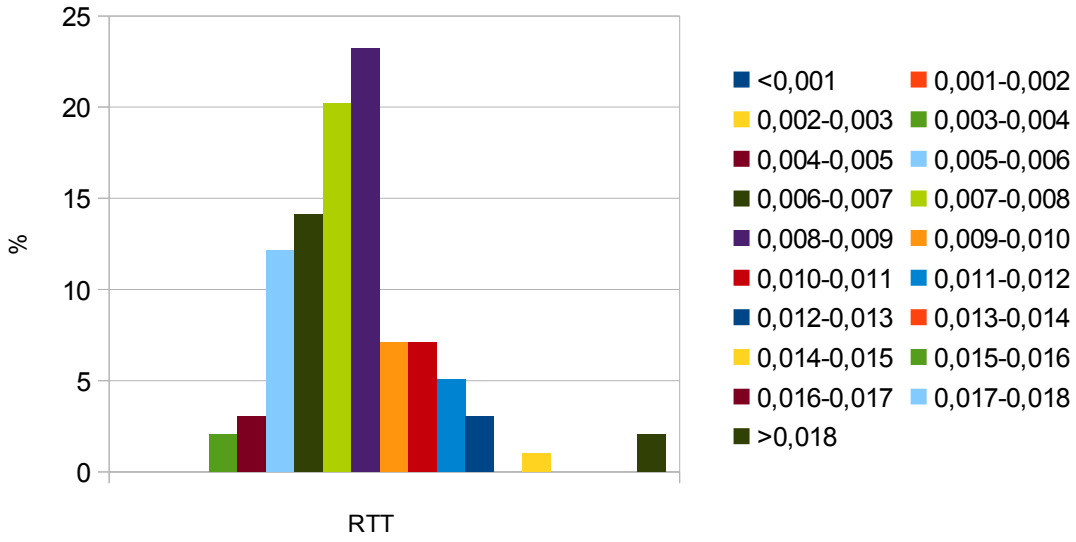
Taulukko 1: Eri yhteysmenetelmien RTT -aikojen keskiarvot ja kursorin keskimääräinen päivitystaajuus.



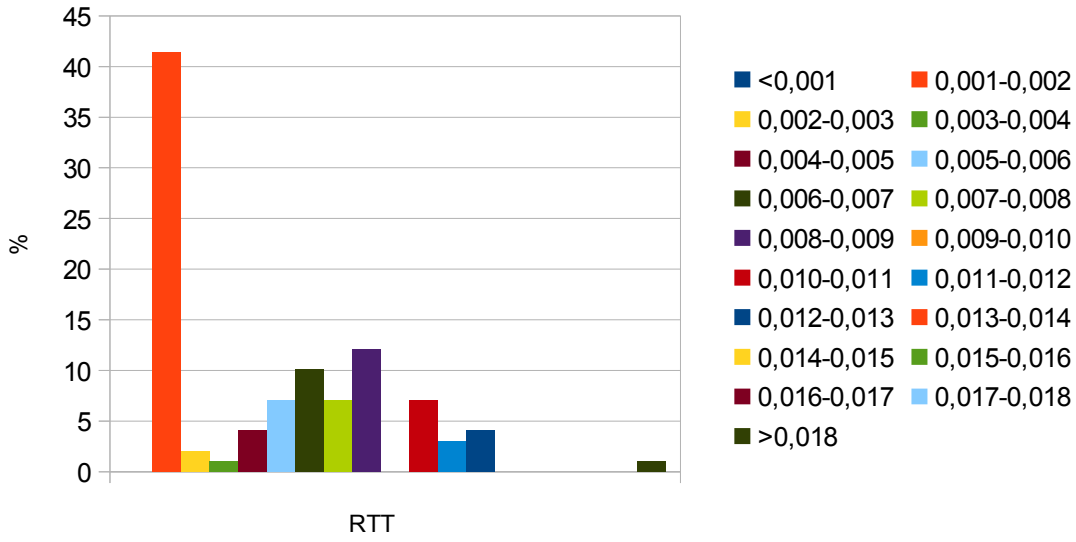
Kuva 34: Kuvaaja eri yhteysmenetelmien RTT -aikojen vaihtelusta.

RTT -ajoista voidaan laskea kyseiselle yhteysmenetelmälle tyypillinen keskiarvo

kursorin päivitystaajuudesta. Jo langattomilla menetelmillä päästiin yli 100 Hz:n, joka on hiirelle riittävä päivitystaajuus.



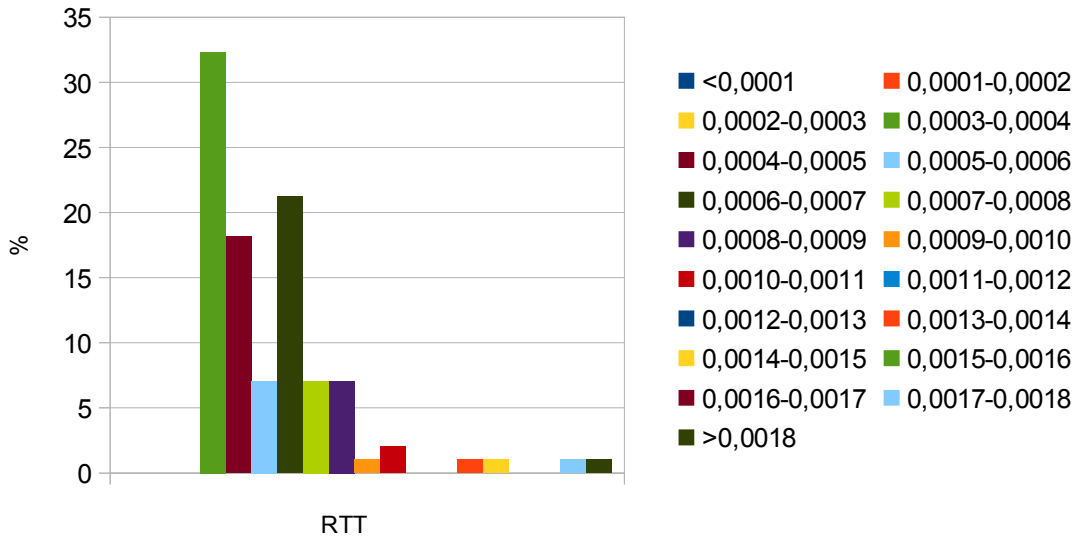
Kuva 35: Histogrammi RTT-ajoista WiFi-yhteydellä.



Kuva 36: Histogrammi RTT-ajoista Bluetooth-yhteydellä.

Keskiarvon lisäksi on mielenkiintoista katsoa kuinka paljon RTT-ajat vaihtelevat keskiarvon ympärillä. Kuvassa 34 on esitetty eri yhteysmenetelmien RTT-aikojen vaihtelu peräkkäisten päivityskertojen välillä ja kuvissa 35, 36 ja 37 on esitetty histogrammit RTT-ajoista eri yhteysmenetelmillä. Kuvaajasta nähdään että

langattomissa yhteysmenetelmissä nähdään muutama piikki, joka saattaa hiirisovellusta käytettäessä näkyä kursorin liikkeen pätkäisynä. Hiirisovelluksen sisältämän kursorin sulavoinnin ollessa päällä, nämä piikit eivät häiritse käyttöä niin paljon.



Kuva 37: Histogrammi RTT-ajoista USB-yhteydellä.

6 YHTEENVETO

Tämän työn tarkoituksena oli toteuttaa langattomasti toimiva hiirisovellus mobiililaitteelle ja siinä onnistuttiin tämän dokumentin kuvaamalla tavalla. Toteutettu sovellus sisälsi mahdollisuudet käyttää kursorin liikuttamiseen sekä mobiililaitteen kiihtyvyysanturia että gyroskooppia. Langaton yhteys toteutettiin käyttämällä UDP/IP-yhteyttä Bluetoothin tai WiFi-avulla, jotta tämä kaikki saatiin toimimaan piti mobiilisovelluksen lisäksi tehdä tietokoneelle sovellus, jonka tarkoitus on toteuttaa hiiritoiminnot tietokoneella. Toteutetun hiirisovelluksen ominaisuuksien yhteenveto löytyy taulukosta 2.

Tarkoitus	Toteuttaa gyro- ja kiihtyvyysanturia käyttävä hiirisovellus iPhone 4 -mobiililaitteelle.
Liitäntätavat	Bluetooth ja WiFi
Kielet	Objective-C ja C++
iOS -sovelluksen rivimäärä	n. 2300 riviä
OS X -sovelluksen rivimäärä	n. 1100 riviä
Windows -sovelluksen rivimäärä	n. 900 riviä

Taulukko 2: Yhteenveto hiirisovellukseen liittyvistä ominaisuuksista.

Mobiilisovelluksen käyttöliittymä tehtiin laitteessa olevan kosketusnäytön ehdoilla. Sovellus rakentuu kahdesta näkymästä, joista toinen näkymä mahdollistaa hiiren käytön ja toinen hiiren toimintaan vaikuttavien asetusten muuttamisen.

Toteutettu hiirisovellus saatiin toimimaan käytetyn iPhone 4 -mobiililaitteen ja tietokoneen välillä siten, että se soveltui hyvin peruskäyttöön, johon voisi lukea esimerkiksi PowerPoint-esityksen pitämisen. Hiiren toimintaa kokeiltiin sekä Mac- että PC-tietokoneella. Hiirisovelluksen toimintaan vaikuttavia seikkoja olivat käytetyn yhteyden viiveet sekä mobiililaitteen antureilta saatava signaali. Hiiri toimi riittävän hyvin peruskäyttöä ajatellen kaikilla sovelluksen tukemilla yhteystavoilla. Hitaimmillaankin päästin yli 100hz:n päivitystaajuuteen, joka on hiirelle riittävä. Myös mobiililaitteen antureilta saadut näytearvot olivat riittävän hyvälaatuisia ja niitä sopivasti skaalaamalla saatiin hiiri toimimaan toivotunlaisesti. Lisäksi yhdistämällä gyro- ja kiihtyvyysanturin näytearvoja saatiin kallistuskulmaan perustuvaan

toimintatilaan huomattavasti lisää tarkkuutta. Mobiililaite suostui antamaan näytearvoja inertia-antureilta maksimissaan n. 110Hz taajuudella, mutta tämä on tähän käyttöön riittävä.

Työn tavoitteena ollut inertia-antureita käyttävä hiirisovellus mobiililaitteelle saatiin toteutettu, vaikkakaan se ei ihan sellaisenaan ole loppukäyttäjälle valmiissa kunnossa esimerkiksi Applen App Storea varten. Tätä ajatellen parannettavaa löytyy käytetyssä kommunikointiprotokollassa ja käyttöliittymässä. Mobiili- ja tietokonesovelluksen välistä kommunikointiprotokollaa voisi yksinkertaistaa ja käyttöliittymästä voisi tehdä loppukäyttäjälle helpomman käyttää. Erityisesti tietokoneen IP-osoitteen syöttäminen käsin mobiilisovellukselle on hankalaa. Tarkkaan työhön perinteinen pöydällä toimiva hiiri on huomattavasti parempi, mutta tilanteissa, joissa tarkkuutta ei hirveästi tarvita ja tasaista pöytäpintaa ei löydy, on toteutettu hiirisovellus käyttökelpoinen.

6 LÄHTEET

[1] Apple. IPhone 4 Tech Specs. [viitattu 15.7.2012]. Saatavilla http://www.apple.com/iphone/iphone-4/specs.html

[2] Logitech. MX Air Rechargeable Cordless Air Mouse Datasheet.[viitattu 15.7.2012]. Saatavilla http://www.pdfdoc.ru/information-technology/computer-peripherals/mice/cordless/logitech-mx-air-cordless-air-mouse-931633-0120-521-560/data-sheet-logitech-mx-air-cordless-air-mouse

[3] RPA Technology. Mobile Mouse Pro Overview. [viitattu 15.7.2012]. Saatavilla http://www.mobilemouse.com/overview.html

[4] David H. Titterton, John L. Weston. Strapdown Inertial Navigation Technology - 2nd Edition. The Institution of Electrical Engineers 2004. ISBN 0 86341 358 7.

[5] Chipworks. Teardown of the Apple iPhone 4 Smart Phone. [viitattu 26.8.2012]. Saatavilla http://www.chipworks.com/blog/recentteardowns/2010/06/23/silicon-teardown-of-the-apple-iphone-4-smart-phone/

[6] ST. LIS331DLH datasheet. [viitattu 26.8.2012]. Saatavilla http://www.st.com/internet/com/TECHNICAL_RESOURCES/TECHNICAL_LITERATURE/DATASHEET/CD00213470.pdf

[7] Kazusuke Maenaka. MEMS Inertial Sensors and Their Applications. Graduate School of Engineering, University of Hyogo Himeji, Japan.

[8] David H. Titterton, John L. Weston. Modern inertial navigation technology and its application. Electronics & Communication Engineering Journal April 2000.

[9] Apple. Language. [viitattu 20.8.2012]. Saatavilla https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/Languages.html

[10] Apple. Tools. [viitattu 20.8.2012]. Saatavilla

https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/Tools.html

[11] Apple. Survey the Major Frameworks. [viitattu 20.8.2012]. Saatavilla http://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/SurveytheMajorFrameworks/SurveytheMajorFrameworks/SurveytheMajorFrameworks.html

[12] Apple. iOS Technology Overview. [viitattu 26.8.2012]. Saatavilla http://developer.apple.com/library/ios/#documentation/Miscellaneous/Conceptual/iPhoneOSTechOverview/Introduction/Introduction.html#//apple_ref/doc/uid/TP40007898-CH1-SW1

[13] Apple. Streamline Your App with Design Patterns. [viitattu 26.8.2012]. Saatavilla https://developer.apple.com/library/ios/#referencelibrary/GettingStarted/RoadMapiOS/chapters/StreamlineYourAppswithDesignPatterns/StreamlineYourApps/StreamlineYourApps.html

[14] Apple. Motion Events. [viitattu 30.9.2012]. Saatavilla http://developer.apple.com/library/ios/#documentation/EventHandling/Conceptual/EventHandlingiPhoneOS/MotionEvents/MotionEvents.html

[15] Apple. IOS: Understanding Personal Hotspot. [viitattu 15.7.2012]. Saatavilla http://support.apple.com/kb/HT4517

[16] Shane Colton. The Balance Filter. [viitattu 15.7.2012].

[17] Apple. socket(2) Mac OS X Developer Tools Manual Page. [viitattu 15.7.2012]. Saatavilla https://developer.apple.com/library/mac/#documentation/darwin/reference/manpages/man2/socket.2.html

[18] Apple. Quartz Event Services Reference. [viitattu 15.7.2012]. Saatavilla https://developer.apple.com/library/mac/#documentation/Carbon/Reference/QuartzEventServicesRef/Reference/reference.html

[19] Microsoft. Socket function. [viitattu 15.7.2012]. Saatavilla http://msdn.microsoft.com/en-us/library/windows/desktop/ms740506%28v=vs.85%29.aspx

[20] Microsoft. Mouse_event function. [viitattu 15.7.2012]. Saatavilla http://msdn.microsoft.com/en-us/library/windows/desktop/ms646260%28v=vs.85%29.aspx

[21] IETF. RFC 768 User Datagram Protocol. [viitattu 15.7.2012]. Saatavilla http://www.ietf.org/rfc/rfc768.txt