Jani Tissari

RUUVITRACKER - AVOIMEN LÄHDEKOODIN GPS-PAIKANNUSJÄRJESTELMÄ

Järjestelmän toiminta ja hyödyntäminen

Opinnäytetyö
Tietotekniikan koulutusohjelma

Maaliskuu 2013

MIKKELIN AMMATTIKORKEAKOULU
Mikkeli University of Applied Sciences

KUVAILULEHTI

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Opinnäytetyön päivämäärä** **7.3.2013**
Tekijä(t) Jani Tissari	**Koulutusohjelma ja suuntautuminen** **Tietotekniikan koulutusohjelma**

Nimeke

RuuviTracker- avoimen lähdekoodin GPS- paikannusjärjestelmä
Järjestelmän toiminta ja hyödyntäminen

Tiivistelmä

RuuviTracker on Lauri Jämsän suunnittelema ja harrastelijapohjalta aloitettu GPS-paikannusjärjestelmä, joka noudattaa avoimen lähdekoodin periaatteita. Järjestelmä koostuu fyysisestä RuuviTracker-laitteesta, sekä sitä varten rakennetusta Linux-pohjaisesta palvelinympäristöstä. Laitteen toiminta perustuu mikrokontrollerilla ohjattuun GPS-paikannukseen ja langattomaan tiedonsiirtoon GPRS-verkon välityksellä. Tällöin RuuviTracker-laitteen sijaintia on mahdollista seurata esim. tavallisen internet-selaimen tai matkapuhelimelle ohjelmoidun sovelluksen välityksellä. Laite sisältää lukuisia erilaisia komponentteja ja antureita, sekä tarjoaa käyttäjille mahdollisuuden kehittää ja laajentaa laitteen ominaisuuksia. Kyseessä on siis eräänlainen elektroniikan kehitysalusta, joka tarjoaa perinteisenä versiona valmiit ominaisuudet langatonta GPS-paikannusta varten ja tukee järjestelmän laajentamista myös omia sovelluksia varten.

Tämän opinnäytetyön tavoitteena on RuuviTracker- järjestelmän toimintaan perehtyminen ja sen hyödyntäminen. Olen seurannut ja kerännyt tietoa RuuviTracker- järjestelmän kehittymisestä ja suunnitteluvaiheesta tuotantolinjalle ja sitä kautta käytäntöön sovellettavaksi. Työni varsinainen tarkoitus on esitellä, kuinka RuuviTracker-järjestelmä toimii kokonaisuudessaan, sekä rakentaa Windows Phone-käyttöjärjestelmälle sovellus, joka kykenee kommunikoimaan RuuviTracker-laitteen kanssa.

Henkilökohtainen työpanokseni pitää sisällään paljon aiheeseen liittyvää tutkimustyötä, jonka tarkoituksena on koota RuuviTracker-järjestelmän tietoutta yhteen pakettiin. Työni käsittelee paljon teoriapohjaista tietoa järjestelmän vaatimista ominaisuuksista, kuten GPS-, GSM, GPRS-verkkojen toimintaa, sekä yleistä tietoa mikrokontrollereiden tekniikoista ja Windows Phone-käyttöjärjestelmästä. Tarkoituksenani on laatia mahdollisimman kattava tietolähde kyseisen projektin ymmärtämiseksi, sekä ohjelmointityön avulla demonstroida RuuviTracker- laitteen hyödyntämismahdollisuuksia käytännössä. Työtä varten rakennettu ohjelma esittelee RuuviTracker- järjestelmän keskeisimpiä toimintoja, kuten sijaintitietojen piirtämistä kartalle ja RuuviTracker- palvelimen tilannetietoja. Ohjelma hyödyntää myös Windows Phone- laitteen omaa sijaintidataa.

Asiasanat (avainsanat)

Vapaa lähdekoodi, RuuviTracker, GPS, GSM, GPRS, Mikrokontrolleri, Windows Phone, .NET, C#

Sivumäärä	**Kieli**	**URN**
73+1+7	Suomi	

Huomautus (huomautukset liitteistä)

Ohjaavan opettajan nimi	**Opinnäytetyön toimeksiantaja**
Reijo Vuohelainen	

DESCRIPTION

MIKKELIN AMMATTIKORKEAKOULU Mikkeli University of Applied Sciences	**Date of the bachelor's thesis** **7.3.2013**
Author(s) Jani Tissari	**Degreeprogramme and option** Informationtechnology

Name of the bachelor's thesis

RuuviTracker – open source GPS- positioning system and it's features

Abstract

RuuviTracker is a GPS tracking system designed and developed by Mr.LauriJämsä. The system consists of a physical device called RuuviTracker and for the purpose-built Linux-based server environment. The whole system is designed to follow the principles of open source ideology. The functionality of the device is based on the microcontroller-managed GPS positioning service and wireless communication via GPRS network. This makes it possible to follow RuuviTracker device's location with the standard internet web browser or a mobile phone. The device includes the number of different components and sensors as well as provides users the ability to develop and expand the properties of the device. That makes RuuviTracker a kind of electronics development platform that provides a wirelessly controllable GPS positioning system through its original form, but it still supports developing and expanding the system for further applications.

The main goal of this thesis was to study the functionality of the RuuviTracker system as thoroughly as possible and learn how it could be used in practice. I followed and collected information about RuuviTracker's different development stages, from the idea to the production line and in to deployment. The actual purpose of my work was to introduce how the RuuviTracker system worked as a whole and also to program a mobile application that would be able to communicate with the RuuviTracker device. As a programming platform I used the Windows Phone operating system, supported by many new smartphones, such as Nokia, HTC and Samsung.

My personal contribution included a lot of research work on the subject in order to draw up an information package for the RuuviTracker system. My work dealt with theory-based information about the features required by the system, such as GPS, GSM and GPRS networks, microcontroller techniques and the Windows Phone operating system. The study resulted in a comprehensive information package on RuuviTracker and its system requirements. The programming work demonstrated RuuviTracker's potential in practice.

Subjectheadings, (keywords)

Open source, RuuviTracker, Satellite Navigation, GSM, GPRS, Microcontroller, Windows Phone, .NET, C#

Pages	Language	URN
73+1+7	Finnish	

Remarks, notes on appendices

Tutor	Bachelor's thesis assigned by
Reijo Vuohelainen	

SISÄLTÖ

LYHENTEET JA TERMIT

3G	Third Generation
	Matkapuhelinteknologian sukupolvi
4G	Fourth Generation
	Matkapuhelinteknologian sukupolvi
A/D	Analog-to-Digital Converter
	Signaalin muuntaja
ADC	Analog-to-Digital Converter
	Signaalin muuntaja
A-GPS	Assisted GPS
	GPS- paikannustekniikka
AMPS	Advanced Mobile Phone System
	Matkapuhelinteknologia
API	Application Programming Interface
	Ohjelmointirajapinta
ARM	Advanced RISC Machines
	Mikrokontrollerivalmistaja
AT	Attention
	Modeemilaitteiden käskykanta
BCH	Broadcast Channel
	GSM- verkon radiosolu
BGA	Ball Grid Array
	Mikropiirien juotostekniikka
BIOS	Basic Input-Output System
	Tietokoneohjelma
BSC	Base Station Controller
	GSM-arkkitehtuurin toimielin
BSDL	Boundary Scan Description Language
	JTAG- ympäristössä käytettävä merkintäkieli
BSS	Base Station Subsystem
	GSM-arkkitehtuurin toimielin
BTS	Base Transceiver Station
	GSM-arkkitehtuurin toimielin
C/A	Coarse Acquisition Code

	GPS- satelliittien lähettämä signaali
CAN	Controller Area Network
	Mikrokontrollereiden ja ajoneuvojen hyödyntämä käyttöliittymä
CDMA	Code Division Multiple Access
	Radiosignaalin kanavanvaraustekniikka
CLR	Common Language Runtime
	.NET- ohjelmointiympäristön osa
CPU	Central Processing Unit
	Laskentayksikkö
D/A	Digital-to-Analog Converter
	Signaalin muuntaja
DOD	Department of Defense
	Amerikan puolustusministeriö
DVD	Digital Versatile Disc
	Tallennusmedia
ECU	Engine Control Unit
	Ajoneuvojen tietokoneyksikkö
EDGE	Enhanced Data rates for GSM Evolution
	Matkapuhelinteknologia
EEPROM	Electronically Erasable Programmable Read-Only Memory
	Muistityyppi
ESD	Electrostatic Discharge
	Suojausmerkintä staattiselta sähköltä
FDMA	Frequency Division Multiple Access
	Radiosignaalin kanavanvaraustekniikka
FM	Frequency Modulation
	Taajuusmodulaatio
GPRS	General Packet Radio Service
	Matkapuhelinteknologia
GPS	Global Positioning System
	Maailmanlaajuinen paikannusjärjestelmä
GSM	Global System for Mobile Communications
	Matkapuhelinteknologia
HC-HSDPA	Dual-Carrier High-Speed Downlink Packet Access
	Matkapuhelinteknologia

HSDPA	High-Speed Downlink Packet Access
	Matkapuhelinteknologia
HTML	Hypertext Markup Language
	Kuvauskieli
HTTP	Hypertext Transfer Protocol
	Siirtoprotokolla
I/O	Input / Output
	Sisääntulo- jaulostuloväylät
I²C	Inter-Integrated Circuit
	Mikrokontrollereiden hyödyntämä käyttöliittymä
IDE	Integrated Development Environment
	Kehitysympäristö
IMEI	International Mobile Equipment Identity
	Puhelinlaitteidein tunnistusmerkintä
JSON	JavaScript Object Notation
	Tiedonsiirtotekniikka
JTAG	Joint Test Action Group
	Mikrokontrollereiden hyödyntämä käyttöliittymä
LCD	Liquid Crystal Display
	Näyttölaitteiden tekniikka
LDO	Low-dropout regulator
	Regulaattori-komponentti
LED	Light-Emitting Diode
	Puolijohdekomponentti
LiFePo4	Lithium iron phosphate
	Akkutekniikka
Li-Ion	Lithium-ion
	Akkutekniikka
LiPo	Lithium Polymer
	Akkutekniikka
MCU	Microcontroller Unit
	Mikrokontrolleriyksikkö
MEMS	Micro Electro Mechanical Systems
	Pieni elektro-mekaaninen järjestelmä
MS	Mobile System

	GSM-arkkitehtuurin toimielin
MSC	Mobile Switching Center
	GSM-arkkitehtuurin toimielin
MSIL	Microsoft Intermediate Language
	Ohjelmointikieli
NMEA	National Marine Electronics Association
	Tiedostomuoto
NMT	Nordisk Mobiltelefon
	Matkapuhelinteknologia
NSS	Network and Switching Subsystem
	GSM-arkkitehtuurin toimielin
OSS	Operation and Support Subsystem
	GSM-arkkitehtuurin toimielin
P	Precision Code
	GPS- satelliittien lähettämä signaali
PCU	Packet Control Unit
	GPRS-arkkitehtuurin toimielin
PRN	Pseudo Random Number
	GPS- satelliittien sijainnin ilmaiseva koodi
RAM	Random Access Memory
	Muistityyppi
REST	Representational State Transfer
	Ohjelmistorakenne
RISC	Reduced Instruction Set Computer
	Tietokoneiden suoritinarkkitehtuurien suunnittelufilosofia
ROM	Read Only Memory
	Muistityyppi
Rx	Receive
	Vastaanottolinja
SATA	Serial ATA
	Tietokoneiden väylätekniikka
SD	Secure Digital
	Muistikortti
SDK	Software Development Kit
	Kehitystyökalut

SGSN	Serving GPRS Support Node
	GPRS-arkkitehtuurin toimielin
SIM	Subscriber Identity Module
	Matkapuhelinten operaattoritunnus
SMS	Short Message Service
	Tekstiviestipalvelu
SPI	Serial Peripheral Interface
	Mikrokontrollereiden käyttöliittymä
SQL	Structured Query Language
	Ohjelmointikieli
TCP/IP	Transmission Control Protocol / Internet Protocol
	Internet protokolla
TDMA	Time Division Multiple Access
	Radiosignaalin kanavanvaraustekniikka
Tx	Transmit
	Lähetyslinja
UART	Universal Asynchronous Receiver Transmitter
	Mikrokontrollereiden ja muiden laitteiden käyttöliittymä
UMTS	Universal Mobile Telecommunications System
	Matkapuhelinteknologia
URL	Uniform Resource Locator
	Internet sivujen osoittaja
USART	Universal Synchronous/Asynchronous Receiver-Transmitter
	Mikrokontrollereiden ja muiden laitteiden käyttöliittymä
USB	Universal Serial Bus
	Liitäntätyyppi
USB-OTG	Universal Serial Bus On-The-Go
	Liitäntätyypin ominaisuus
VGA	Video Graphics Array
	Näyttöstandardi
WCDMA	Wideband Code Division Multiple Access
	Radiorajapinta
WLAN	Wireless Local Area Network
	Langaton internet
XAML	Extensible Application Markup Language

Ohjelmointikieli

1 JOHDANTO

Nykyään GPS-paikannusjärjestelmät ovat yleistyneet matkapuhelimiin ja lukuisiin muihin elektronisiin laitteisiin. Paikannuslaitteita valmistetaan mitä erikoisempiin tarkoituksiin aina koirien kaulapantajäljittimistä auton varashälyttimiin. Markkinoilla olevat paikannuslaitteet ovat miltei aina kaupallisia tuotteita, jotka maksavat paljon. Lisämaksua kertyy useimmiten myös käyttölisenssien ylläpitämisestä ja laitteiden päivittämisestä. Tästä kaikesta huolimatta laitteiden laajentaminen tai hyödyntäminen muuhun käyttöön on liki mahdotonta, koska paikantimet ovat suljettuja järjestelmiä.

Olin pohdiskellut päättötyöni aiheeksi jotain GPS-paikannukseen liittyvää ja löysinkin idean selaillessani ruuvipenkki.fi-nimistä internet-sivustoa, jonka käyttäjäkunta koostuu tietotekniikan ja elektroniikan harrastelijoista, sekä ammattilaisista. Törmäsin sivustolla harrastelijavoimin aloitettuun projektiin, jonka tarkoituksena oli rakentaa halpa ja langattomasti luettava GPS-paikannuslaite, jonka tekniikka perustuu avoimeen lähdekoodiin. Tutustuin kyseiseen RuuviTracker-nimiseen projektiin tarkemmin ja myöhemmin totesin, että kyseisestä projektista on aihetta myös päättötyöhön.

RuuviTracker on Lauri Jämsän suunnittelema GPS-paikannusjärjestelmä, joka noudattaa avoimen lähdekoodin ajattelutapaa. Järjestelmän kaikki osat; fyysinen laite, internet palvelin, sekä ohjelmistot ovat toteutettu harrastuspohjalta. Vapaan lähdekoodin ansiosta ne ovat myös kenen tahansa käytettävissä tai muokattavissa omiin tarkoituksiinsa. RuuviTracker- laitteesta tulee myöhemmin myyntiin valmiiksi koottu malli, jolla on tarkoitus myös ansaita kaupallista voittoa.

RuuviTracker-laite on eräänlainen elektroniikan kehitysalusta, jolla voidaan hyödyntää GPS-,GSM- ja GPRS-verkkojen tarjoamia ominaisuuksia mikrokontrolleritekniikan ohjauksella. Järjestelmään kuuluu myös vapaa Linux-pohjainen palvelinohjelmisto, jonka avulla RuuviTrackerin lähettämää tietoa voidaan ohjata GPRS-verkon välityksellä muille alustoille, kuten tavallisille internet-selaimille tai matkapuhelimille.

Järjestelmän tiedot ovat vapaasti kaikkien saatavilla ja laitetta pystyy helposti muokkaamaan omiin käyttötarpeisiin. Laitteen ensisijainen tarkoitus on GPS-sijaintitiedon siirtäminen internet palvelimelle ja siitä muille laitteille, mutta lisä-antureiden ja ohjelmoinnin avulla myös muiden tietojen siirtäminen on mahdollista.

Seurasin RuuviTracker-projektia sen alusta alkaen ja kirjoittelinkin mm. ensimmäiselle laiteprototyypille käsinkolvausohjeen. Tarvitsin kuitenkin päättötyötä varten paljon konkreettisempaa tekemistä kuin pelkkää tutkimustyötä. Päätin myöhemmin ostettuani uuden Windows-puhelimen, että päättötyöni voisi yhdistää kaksi asiaa ja puhelimelle voisi rakentaa ohjelman, jonka avulla RuuviTracker-laitetta olisi mahdollista seurata.

Työni tarkoitus on tutustua mahdollisimman tarkasti RuuviTracker järjestelmän toimintaan sekä fyysisellä, että ohjelmallisella tasolla. Työni teoriaosuus käy läpi tarvittavat asiat mikrokontrollereiden perustoiminnasta aina GPS-, GSM-, GPRS-verkkojen rakenteeseen, sekä Windows Phone-käyttöjärjestelmän periaatteisiin. Teoriaosuuden jälkeen tutustutaan RuuviTracker-laitteeseen ja sen toimintaan aina suunnitteluvaiheesta tuotantolinjalle asti. Pääsin seuraamaan ja dokumentoimaan laitteen tuotantoa Savonlinnan 3K-tehtaalle, joten päätin hyödyntää työssäni tietoa myös itse tuotantoprosessista.

Varsinainen työskentelyosuus koostuu ohjelmointityöstä, jonka teen Windows Phone-käyttöjärjestelmälle. Ohjelman ensisijaisena tarkoituksena on pystyä suorittamaan yksinkertaista kommunikointia RuuviTracker-laitteen kanssa langattomasti internetin välityksellä. Kun ohjelmaa saadaan kehitettyä tarpeeksi, on se tarkoitus julkaista myös julkiseen käyttöön.

2 MIKROKONTROLLERI

Mikrokontrollerilla tarkoitetaan yksittäistä mikropiiriä, jonka sisään on integroitu suuri joukko erilaisia toimielimiä. Mikrokontrollerin avulla pystytään ohjamaan ulkoisia laitteita halutuilla tavoilla, sillä piirin suorittama ohjelmarakenne on itse suunniteltavissa. Toiminnoiltaan mikrokontrolleria voidaan kuvitella tietokoneena, joka on rakennettu yhden mikropiirin sisälle. [1 & 2.]

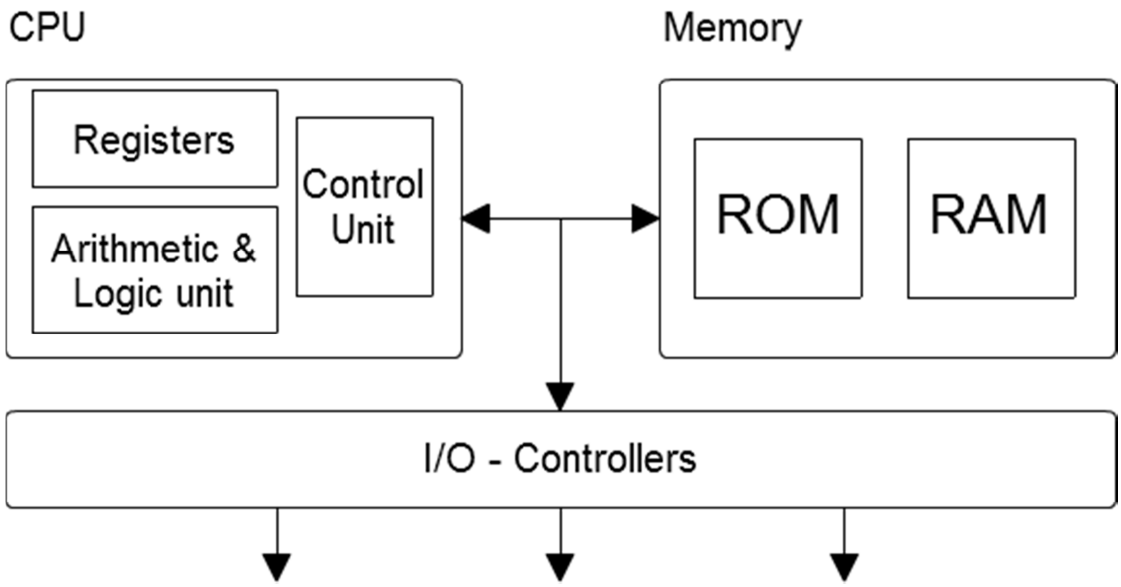
Nykypäivänä mikrokontrollereita käytetään melkein kaikissa mahdollisissa elektronisissa sovelluksissa. Kaukosäätimet, pesukoneet ja lähes kaikki digitaaliset laitteet hyödyntävät mikrokontrolleritekniikkaa. Yleisyyteen vaikuttavat syyt ovat melko yksiselitteisiä, sillä piirit ovat halpoja, vähävirtaisia, vievät hyvin vähän tilaa ja niiden suunnittelutyö erilaisiin käyttökohteisiin on suhteellisen helppoa. [1 & 2.]

Mikrokontrolleri tarvitsee yleensä sähkön lisäksi vain ulkoisen kiteen ja pari kondensaattoria, joiden avulla piirille luodaan sen tarvitsemat kellopulssit. Mikrokontrolleri on siis toiminnaltaan hyvin autonominen. [1 & 2.]

2.1 Tietokone- ja mikrokontrolleriarkkitehtuuri

Mikrokontrollerin fyysistä rakennetta voidaan tarkastella erittelemällä piirin sisäisiä toimielimiä ja tarkastelemalla niiden toimintaa. Hyvänä vertailukohteena voidaan käyttää useimmille tutumman tietokonearkkitehtuurin rakennetta, sillä molemmat mallit ovat keskenään melko samanlaisia.

Yksinkertaisimmillaan tietokonearkkitehtuuri voidaan paloitella kolmeen eri osaan: suoritinyksikköön, muisteihin ja I/O-väyliin. Kyseinen perusjako ilmenee lähes jokaisessa tietokonepohjaisessa arkkitehtuurimallissa, vaikkakin tekniikka kehittyy jatkuvasti ja arkkitehtuurit vaihtelevat aina eri laitevalmistajien kesken. Tietokonearkkitehtuuria voidaan esittää mm. kuvan 1 mukaisella arkkitehtuuri kuvalla. [38].



KUVA 1. Tietokonearkkitehtuuri [38].

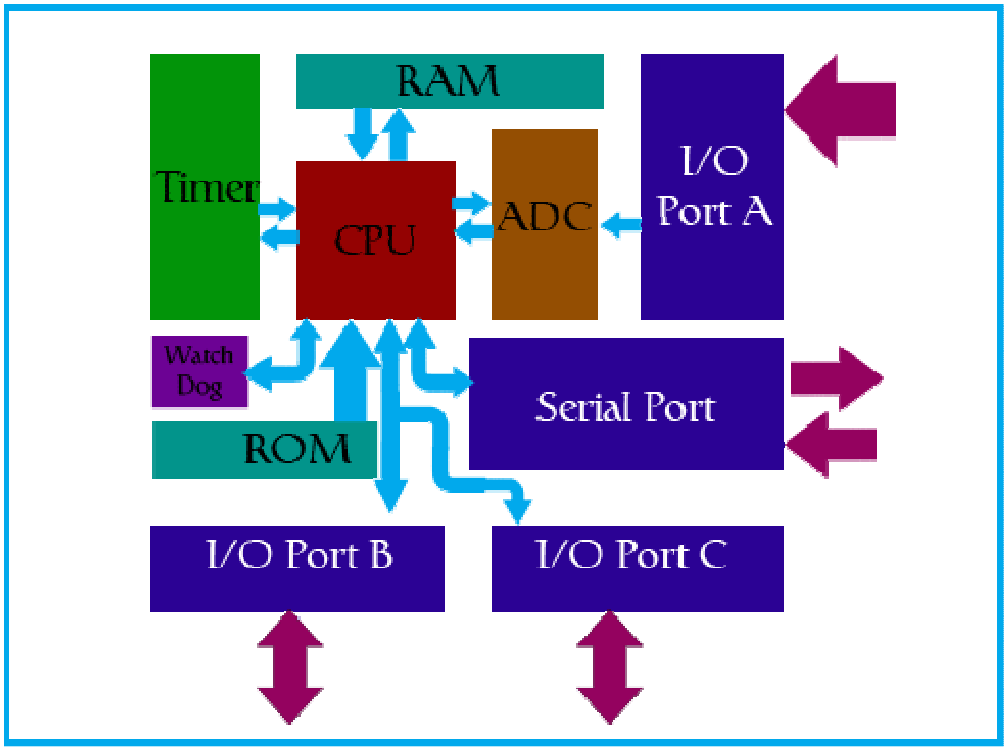
CPU, eli tietokoneympäristössä tutummin tunnettu prosessori on yksikkö, joka suorittaa sen ympäristöön liitettyjen komponenttien vaatimat laskusuoritukset. Prosessorin ytimeen on integroitu erilaisia toimintoyksiköitä, joiden perusteella voidaan suorittaa matemaattisia laskuja. Ytimen hallintayksikön avulla prosessori kykenee ottamaan laskutoimituksia vastaan ja kääntää ne prosessorille ymmärrettävään muotoon, eli prosessorin sisäisten rekisterien mukaisiksi käskyiksi. Tämän jälkeen laskutoimitukset lasketaan matematiikka- ja logiikkayksiköiden avulla ja vastaukset palautetaan hallintayksikön kautta muille komponenteille. [38].

Järjestelmän muisti muodostuu ROM- ja RAM- muisteista. ROM-muisti on haihtumatonta lukumuistia, joka säilyttää tietonsa, vaikka laitteesta katkaistaisiin virrat. Tietokoneissa ROM-muistia hyödynnetään esim. BIOS-tietojen säilytyspaikkana, jonka avulla tietokoneen laiteasetukset pysyvät aina samanlaisina. Ilman ROM-muistia tietokoneen laiteasetukset jouduttaisiin aina käynnistyksen yhteydessä määrittämään uudestaan. RAM-muisti on ns. työmuistia, jota hyödynnetään erilaisten prosessien ylläpitämisessä. Jos halutaan esim. laskea kaksi muuttujaa yhteen, niin muuttujat talletetaan RAM-muistiin niille osoitettuun osoitteeseen, josta prosessori osaa sitten noutaa ne laskusuoritusta varten. [38].

I/O-väylät muodostuvat erilaisista liitännäisyksiköistä, joiden avulla tietokoneeseen voidaan liittää erillisiä laitteita, kuten näytönohjaimia, USB-laitteita, DVD-asemia, kiintolevyjä, ym. Liitännäisyksiköt ja niiden käyttämät väylät ovat suunniteltu erilais-

ten standardien mukaisesti, jotta erilliset laitteet pystyvät kommunikoimaan järjestelmän kanssa oikein. Tämän seurauksena eri protokollat käyttävät erilaisia liittimiä, kuten esim. USB-laitteet käyttävät niille suunniteltua porttia ja kiintolevyt liitetään SATA-väylään. [38].

Kun siirrytään tarkastelemaan mikrokontrollereiden arkkitehtuuria, niin kuvassa 2 huomataan, että siinä esiintyy sama perusrakenne, kuin tietokonearkkitehtuurissa. Suurimpana erona arkkitehtuurien välillä on, että yleisempi tietokonearkkitehtuuri vastaa fyysisesti suurempaa järjestelmää ja siinä lähes aina hyödynnetään myös mikrokontrollereita. Kuvan 2 esittämä arkkitehtuurirakenne noudattaa yleisimpiä mikrokontrollereiden ominaisuuksia, jotka kuitenkin vaihtelevat eri piirien välillä.



KUVA 2. Mikrokontrollerin arkkitehtuuri [37].

Kuten huomataan, niin arkkitehtuurin perusrakenne säilyy samana, lukuunottamatta muutamaa lisäelementtiä. Myös muistien koot ja prosessorin kellotaajuudet eroavat arkkitehtuurien välillä, sillä tietokoneet suorittavat huomattavasti enemmän laskutoimituksia. Mikrokontrollerien muistikoot vaihtelevat malleista riippuen yleensä kymmenistä kilotavuista muutamiin megatavuihin ja kellotaajuudet muutamista megahertseistä satoihin. Mikrokontrollerit suunnitellaan pääasiassa käytettäväksi sulautetuissa

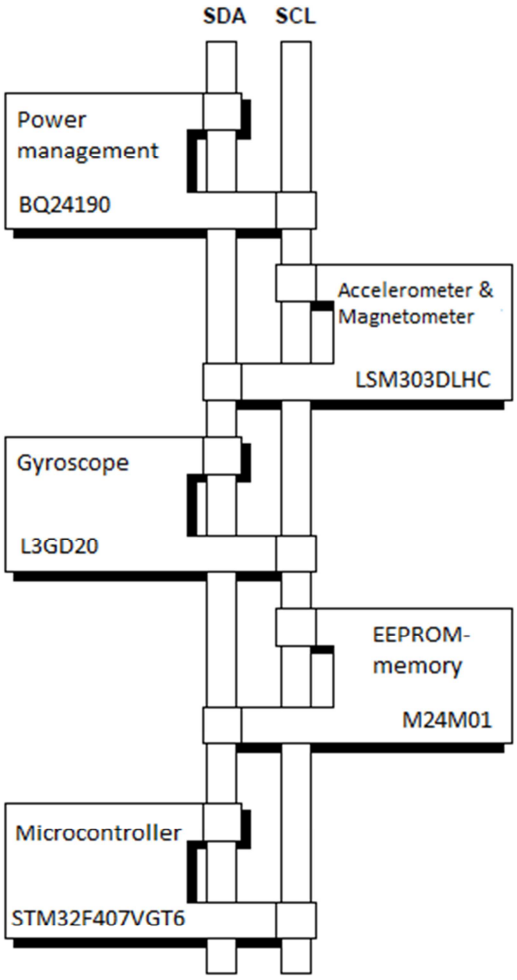
järjestelmissä, eli suorittamaan vain niille ohjelmoituja prosesseja. Tämän takia tehontarve on paljon pienempi, kuin tietokoneissa. [37].

Mikrokontrollerin arkkitehtuuriin kuuluvat useimmiten kuvassakin esiintyvät ajastin-, ADC-, sarjaportti- ja vahtikoirayksiköt. Ne ovat yleisimpiä mikrokontrollereiden tarvitsemia ominaisuuksia, joten ne useimmiten löytyvät piireistä integroituina. Ajastinyksiköiden avulla mikrokontrolleri saadaan tarvittaessa mittaamaan prosessien aikaviiveitä, eli ns. ”ymmärtämään kellonaikaa”. Sarjaporttiliitäntä on yksi mikrokontrollereiden yleisimmistä kommunikointiliitännöistä, jota voidaan käyttää lukuisiin tiedonsiirtotapoihin, kuten esim. mikrokontrollereiden ohjelmoimiseen. ADC-yksiköiden avulla saadaan muunnettua analogista signaalia digitaaliseksi ja myös mahdollisesti päinvastoin. Vahtikoiralla viitataan piirin sisäiseen järjestelmään, joka tarkkailee mikrokontrollerin toimintaa ja kriittisten virheiden ilmetessä käynnistää piirin uudelleen. [37].

Mikrokontrollerit tukevat useita erilaisia käyttöliittymiä, joita tarvitaan tiedon käsittelemisessä ja -siirtämisessä esimerkiksi eri komponenttien tai laitteiden välillä. Käyttöliittymien lukumäärät ja tyypit vaihtelevat laajasti eri piirinvalmistajien ja piirimallien välillä. Mikrokontrollereiden valinnoissa tuleekin aina ottaa huomioon rakennettavan laitteen käyttötarkoitukset, joiden avulla pystytään määrittämään oikeanlainen piiri haluttuun käyttötarkoitukseen. [37].

2.2 I²C

I²C on Philipsin kehittämä sarjamuotoinen tiedonsiirtokäyttöliittymä, joka on suunniteltu ensisijaisesti mikrokontrollereita varten piirilevyn sisäiseen tiedonsiirtoon. Kyseessä on kaksisuuntainen tiedonsiirtoväylä, joka muodostuu data- ja kellolinjasta. Väylän toimintaperiaate toimii master/slave-periaatteella, jossa master-komponentti toimii väylän isäntänä. Isäntä mm. generoi väylän käyttämän kellotaajuuden ja päättää tiedonsiirron aloittamis- ja lopettamisajankohdista. Slave, eli väylän ns. orjakomponentit lähettävät ja vastaanottavat tietoa, kun isäntä komponentti niin käskyttää. [3 & 4, s. 4-6.]

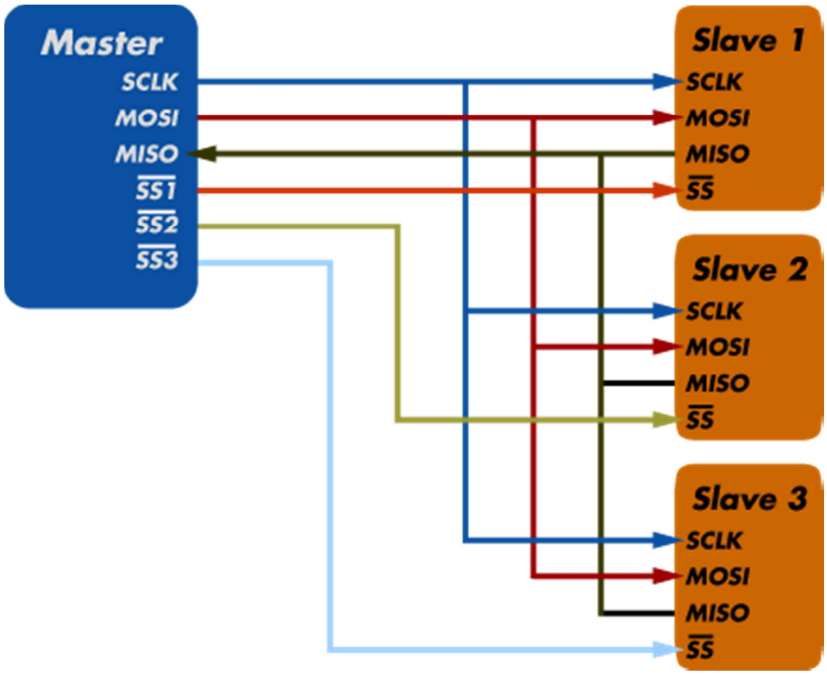


KUVA 3. I²C-väylän kytkentäperiaate [4, s. 4-6]

Kuvassa 3 mallinnetaan I²C-väylän kytkentäperiaatetta. Kuvan 3 alin komponentti, eli mikrokontrolleri toimii väylän isäntänä ja muut komponentit orjalaitteina. Jokaiseen väylään liitetyllä komponentilla on oma uniikki osoite, joiden avulla yhteydenpito isännän ja orjan välillä mahdollistuu. Jos mikrokontrolleri haluaa noutaa esim. gyroskoopin tuottamaa dataa, niin pyyntökäsky muodostetaan väyläosoitteen avulla. Tällöin käskypyyntö ei vaikuta muiden komponenttien toimintaan häiritsevästi ja vastaus saadaan takaisin oikealta komponentilta. [3 & 4, s. 4-6.]

2.3 SPI

SPI on Motorolan kehittämä sarjamuotoinen tiedonsiirtokäyttöliittymä, joka myös toimii I²C:n tavoin master/slave-periaatteella. Käyttöliittymä on toiminnoiltaan ja rakenteeltaan melko samanlainen, kuin I²C. Näkyvin ero ilmenee kuitenkin SPI:n käyttämästä neljästä kytkentälinjasta I²C:n kahden sijaan, joka huomataan myös kuvasta 4. [5].



KUVA 4. SPI-väylän kytkentäperiaate

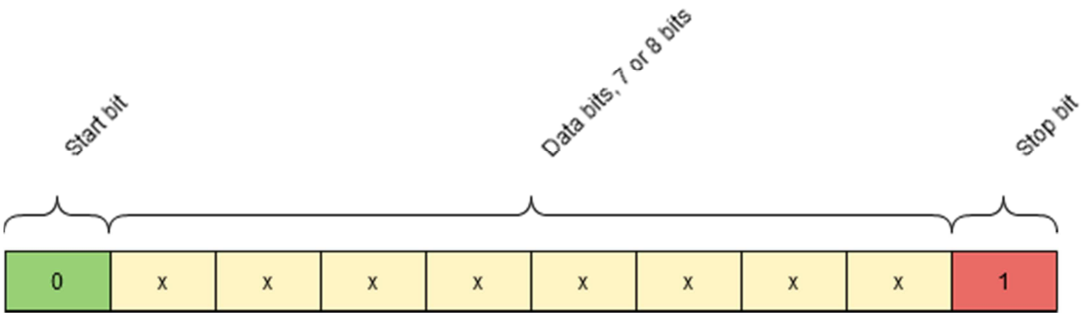
SPI-liittymä muodostuu kuvan 4 mukaisesti kellosignaalista (SCLK), sisääntulo- ja ulostulosignaaleista (MISO, MOSI) ja orjakomponentin valintasignaalista (SS). Orjalaitteiden tunnistus tapahtuu nimien perusteella ja toimintaperiaate on melko samanlainen kuin I²C-kytkennässä. [5].

SPI- ja I²C-käyttöliittymien suurin ero ilmenee tiedonsiirtonopeuksista, jonka takia monet mikrokontrollerit tukevat molempia käyttöliittymiä samanaikaisesti takaamalla mahdollisimman monipuolisen suunnitteluympäristön. SPI-väylän avulla päästään suuriin tiedonsiirtonopeuksiin, koska väylän käyttämällä kellotaajuudella ei ole juurikaan rajoitteita. Tämän takia väylä soveltuu hyvin sovelluksiin, jossa komponenttien välinen tiedonsiirto halutaan mahdollisimman nopeaksi ja vakaaksi. Huonoina puolina mainittakoon suurempi tilantarve, koska jokaista orjalaitetta varten joudutaan varaamaan useampi mikrokontrollerin pinni. [5].

2.4 UART ja USART

UART ja USART ovat sarjadatan siirtoon tarkoitettuja yksiköitä, joille on mikrokontrollerilla integroitu tuki. Tietoa voidaan siirtää asynkronisesti (UART), tai synkronisesti/asynkronisesti (USART) eri laitteiden välillä, riippuen käytettävistä laitteista ja käyttöliittymien tuesta. Synkroninen sarjaliikenne käyttää yhtä signaalia kuhunkin suuntaan, kun taas asynkronisessa liikenteessä käytetään erillistä tahdistus signaalia. [9].

UART- ja USART-liitännät ovat yleisimpiä sarjamuotoisia tiedonsiirtometodeja, kun halutaan esim. yhdistää mikrokontrolleri tietokoneeseen kommunikointia varten. Kyseisillä liitännöillä voidaan mm. suorittaa mikrokontrollerin ohjelmointia tietokoneelta käsin tai halutessaan voidaan tulostaa mikrokontrollerin lähettämiä tietoja tietokoneen terminaaliohjelmilla. [9].

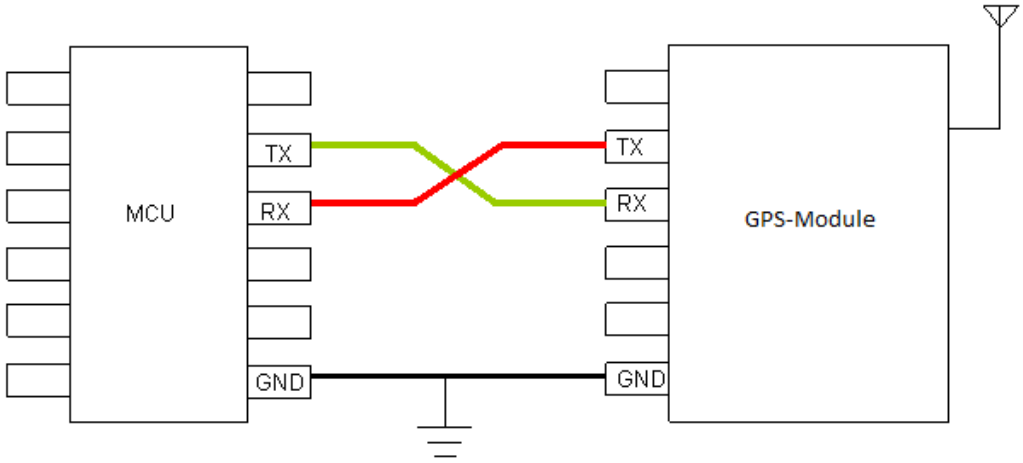


KUVA 5. Sarjamuotoinen datakehys [8, s. 2-4].

Sarjamuotoinen data siirretään UART- ja USART-väylillä kuvan 5 mukaisessa kehysmuodossa. Siirrettävä tieto pakataan 7- tai 8-bitin levyisiin paketteihin, jotka liitetään kuvan mukaisiin lähetyskehyksiin. Lähetyskehys muodostuu aloitus- ja lopetusbitistä, joiden mukaan sarjaliikennettä osataan hallita. Kun kehys on saatu muodostettua, se siirretään bitti kerrallaan vastaanottavalle laitteelle, joka osaa purkaa sen oikeaoppisesti, noudattamalla samaa kehysperiaatetta. [8, s. 2-4].

Yksi tärkeimmistä UART ja USART- sarjaporttiliitäntöjen ominaisuuksista on vuonhallinta. Kyseisen ominaisuuden avulla pystytään vaikuttamaan tiedonsiirron turvallisuuteen varmistamalla, että juuri haluttu datamäärä siirtyy pisteestä A pisteeseen B. Käytännössä järjestelmä on suunniteltu siten, että datan siirtämistä tarkkaillaan jatkuvasti ja mahdollisten virheiden ilmetessä tiedonsiirto voidaan pysäyttää hetkeksi ja

jatkaa myöhemmin, ilman että tietoa katoaisi mihinkään. Esimerkiksi, jos dataa lähettävän sarjaportin tiedonsiirtonopeudeksi on asetettu suurempi arvo, kuin mihin tiedon vastaanottaja on oikeasti kykenevä reagoimaan, niin tietoa joudutaan tallettamaan puskuriin, eli erilliseen muistiin. Tämä tarkoittaisi sitä, että puskuri täyttyisi nopeasti ja sen yli menevä data häviäisi matkalle. Vuonhallinta-ominaisuuden avulla tiedonsiirto voidaan turvata siten, että dataa vastaanottava laite lähettää keskeytyspyynnön lähettävälle laitteelle ennen kuin puskuri täyttyy. Tällöin tiedonsiirto keskeytyy hetkeksi ja jatkuu automaattisesti jälleen, kun puskurimuisti on saatu tyhjennettyä. Kyseisellä tavalla on mahdollista siirtää tietoa laitteiden välillä, joiden tiedonsiirtokapasiteetit eivät normaalisti olisi keskenään yhteensopivia. Tällä tavoin dataa ei häviä missään vaiheessa ja tiedonsiirto on paljon turvallisempaa. [6].

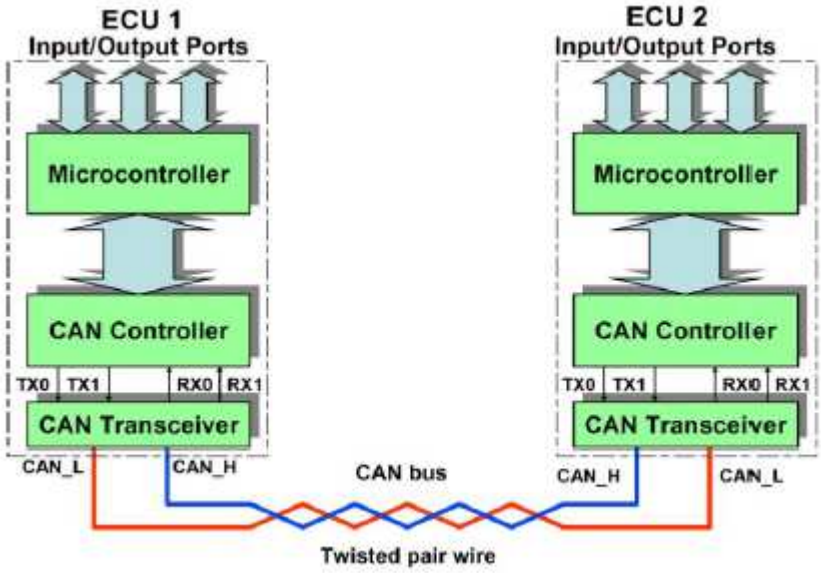


KUVA 6. UART-liitäntä GPS-moduulin ja mikrokontrollerin (MCU) välillä [39].

Mikrokontrollerinpiirilevyllä UART-liitäntä voidaan toteuttaa esim. kuvan 6 mukaisella tavalla. Tiedonsiirrossa tarvitaan vain lähetin- ja vastaanotinlinja, (Tx ja Rx), jos komponenteilla on omat virtakytkennät. Jos halutaan suorittaa UART-kytkentä kaapelin välityksellä, niin mukaan tarvitaan useimmiten virtaliittimet, sekä vuonhallintaliittimet. Vuonhallinnalla toteutetaan ns. kättelyperiaate, jonka avulla keskenään liitetyt laitteet ovat enemmän tietoisia toisistaan ja tiedonsiirto on luotettavampaa. [39].

2.5 CAN

CAN-väylä on Boschin suunnittelema kommunikointiprotokolla, joka toimii viestipohjaisesti. CAN-väylien avulla voidaan liittää esim. useita mikrokontrollereita toisiinsa, jolloin aikaansaadaan väylän nimenkin mukainen laiteverkkoliikenne. CAN-väylän kytkentäperiaate useimmissa tapauksissa on samanlainen kuin UART-väylällä, eli yksinkertainen parikaapelikytkentä, kuten kuvasta 7 ilmenee. [11, s. 5-7].



KUVA 7. CAN-väylän kytkentä kahden ajoneuvotietokoneen (ECU) välillä [10, s. 1-3].

CAN-väylää käytetään ensisijaisesti ajoneuvoissa, sekä monissa muissa teollisuuden laitteissa. Väylä on toiminnoiltaan suhteellisen yksinkertainen ja sen ansiosta myös hyvin vikasietoinen. Väylä mahdollistaa myös pitempien välikaapeleiden käytön kommunikoivien laitteiden välille, kuten jopa 1km. [10, s. 1-3].

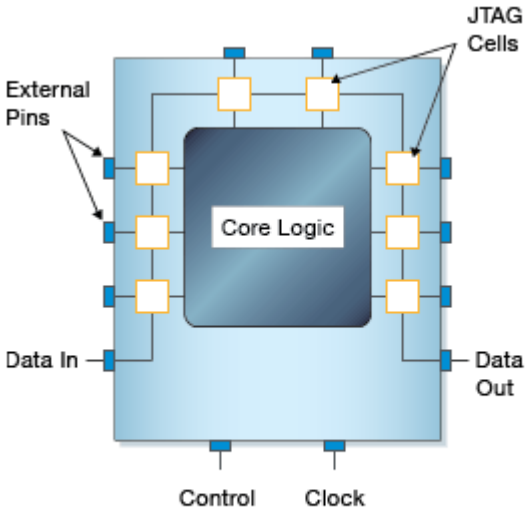
Kuvan 7 esittämässä mallissa havainnollistetaan, kuinka kaksi ajoneuvon tietokoneyksikköä on kytketty toisiinsa. Käytännössä tämä tarkoittaa sitä, että useampia mikrokontrollereita voidaan kytkeä yhtenäiseen ”verkkoon”, jolloin mikrokontrollerit pystyvät käskyttämään tarvittaessa toistensa toimintoja, eli I/O-portteihin kytkettyjä komponentteja. [10, s. 1-3].

Nykyään uudemmat autot sisältävät ajotietokoneita, joka tarkoittaa, että autoista löytyy valmiina CAN-liitäntä. Erillisillä elektronisilla lukulaitteilla pystytään kytkeytymään ajoneuvojen ajotietokoneisiin CAN-liitännän kautta ja mm. lukemaan tai nollaamaan erilaisia vikakoodeja. [10, s. 1-3].

2.6 JTAG

JTAG-ympäristö on aikaisemmin mainittuja liitäntöjä uudempi ja tarkoitusperiltäänkin erilainen. JTAG-liitäntää käytetään lähestulkoon kaikissa uudemmissa mikropiireissä, joiden pinnien lukumäärä on suuri tai kyseessä on altajuotettava BGA-piiri. JTAG-liitäntä kehitettiin elektroniikan suunnittelun ja testaamisen apuvälineeksi, eli ns. debuggausliittymäksi. Debuggaamisella viitataan virheiden paikantamiseen ja korjaukseen elektroniikan sisäisestä toiminnasta. [12 & 13.]

JTAG-kytkennälle on yleensä varattu mikrokontrollerilta tietyt pinnit, joiden avulla piirien debuggaus onnistuu. Jos yhdellä piirilevyllä on useampi JTAG-kytkentäinen komponentti, niin useita komponentteja on mahdollista kytkeä ketjuun, jolloin kaikkien komponenttien seuraaminen onnistuu yhden JTAG-liitännän välityksellä. Komponenttien JTAG-liitännät ovat kytköksissä piirin sisäisiin JTAG-soluihin, kuten kuvassa 8 esitetään. [12 & 13.]



KUVA 8. JTAG- kytkettävä laite ja sen liitännät, sekä solut [12 & 13.]

Jos komponentin toimintaa halutaan testata JTAG-liitännän kautta, niin tarvitaan erillinen JTAG-laite, joka on esim. kytköksissä tietokoneeseen. Tämän jälkeen tarvitaan tutkittavan komponentin BSDL-tiedosto, joka saadaan komponentin valmistajalta. Tiedostoon on määritelty tarkkailtavan, eli ”debugattavan” komponentin mittaustiedot, jonka perusteella osataan suorittaa komponentille juuri oikeanlaiset testit. Myös erilaisten laitteiden ohjelmointi on mahdollista JTAG-litännän välityksellä. [12 & 13.]

3 GPS-TEKNOLOGIA

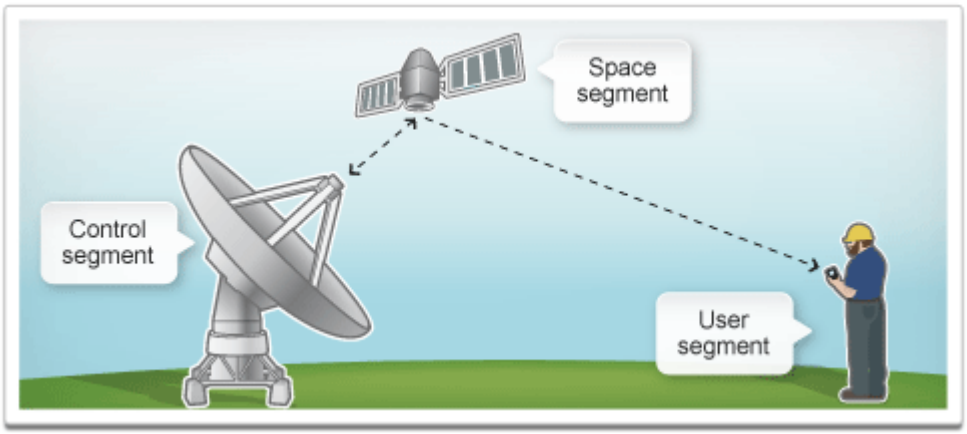
GPS on reaaliaikainen ja maailmanlaajuinen paikannusjärjestelmä, jonka toiminta perustuu maata kiertävien satelliittien lähettämiin paikannustietoihin. Järjestelmän avulla pystytään määrittämään paikannettavien kohteiden leveys- ja pituuskoordinaatit, joiden avulla pystytään selvittämään sijainti kartalla. [14, s. 1-2].

GPS-järjestelmä kehitettiin Amerikan puolustusministeriön DOD:n toimeksiannosta 1970-luvulla. Se suunniteltiin pääasiallisesti sotilaskäyttöön, mutta järjestelmä sallittiin myöhemmin hyödynnettäväksi myös siviilikäytössä. [14, s. 1-2].

Järjestelmä toiminta on passiivista, eli yksisuuntaista tietoliikennettä satelliiteilta paikannuslaitteille. Käytännössä toiminta on melko samanlaista kuin radiosignaalin vastaanotto ja kuuntelu. Tietoa siis voidaan ottaa vastaan, mutta ei lähettää takaisin.
Nykypäivinä paikannuksen tarkkuus teoriassa karttakuvalla on noin 8,5 m. Käytännössä tarkkuutta pystyy parantamaan paljon tarkemmaksikin, riippuen paikannuslaitteen laskentametodeista ja nopeudesta. Suurimmat mittausvirheet johtuvat maapallon ilmakehän aiheuttamista häiriöistä, kuten ionosfäärin tuottamista viiveistä satelliittien radiosignaaleissa. [14, s. 1-2].

3.1 GPS-arkkitehtuuri

Kun halutaan kuvata näinkin monimutkaisen järjestelmän toimintaa, on helpompaa hajottaa eri toiminta-alueet pienempiin, helpommin tulkittaviin osiin. Kun järjestelmä on jaettu pienempiin osiin, ymmärretään helpommin eri segmenttien toimintaa, sekä niiden keskenäistä vuorovaikutusta. GPS-järjestelmän arkkitehtuuria voidaan helpommin kuvata kolmella eri segmentillä, joita ovat kuvassa 9 näkyvät avaruussegmentti, hallintasegmentti, sekä käyttäjäsegmentti. [14, s. 1-3]

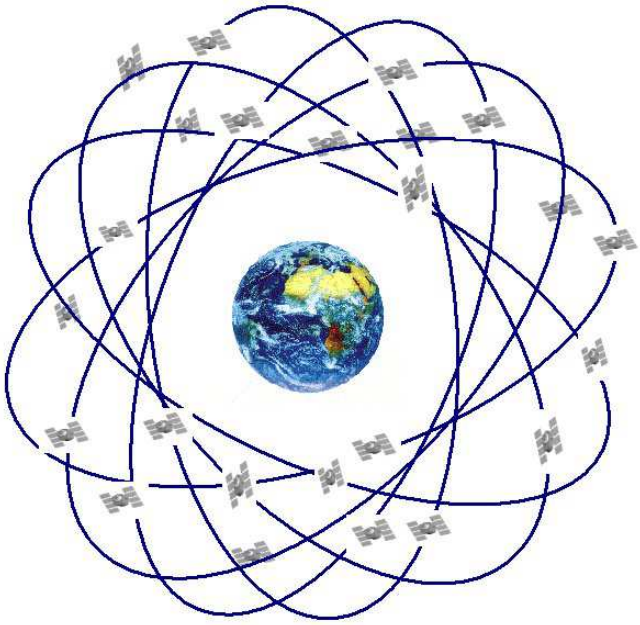


KUVA 9. GPS-arkkitehtuurin segmentit [14, s. 1-3]

GPS-järjestelmän tärkein toimielin koostuu satelliiteista, eli avaruussegmentistä. Paikannukseen käytettäviä satelliitteja on maata kiertävillä radoilla yhteensä 24 kappaletta. Maata kiertäviä ratoja on 6 ja jokaiselle radalle on sijoitettu 4 satelliittia. Tällä tavoin aikaansaadaan symmetrisesti toimiva satelliittiliikenne, joka mahdollistaa riittävän ja jatkuvan peittoalueen kaikkialle maapallolla. [14, s. 1-3]

Satelliitit toimivat noin 20200 km korkeudessa maanpinnasta katsoen. Korkeus vaihtelee hieman riippuen satelliittien sijainnista kiertoradoillaan. Kiertoradan kiertäminen kestää satelliitilta noin 12 tuntia. Kuva 10 mallintaa satelliittien määrää ja sijaintia kiertoradoillaan. [14, s. 1-3]

Satelliitit osaavat määrittää tarkan sijaintinsa kiertoradallaan sisäisen atomikellon ja hallintasegmentin avustuksella. Sijainnista muodostetaan PRN-koodia, joka lähetetään käyttäjäsegmentille. PRN- koodista mainitaan tarkemmin myöhemmässä pseudoetäisyys kappaleessa. [14, s. 1-3]



KUVA 10. GPS-satelliitit ja niiden kiertoradat

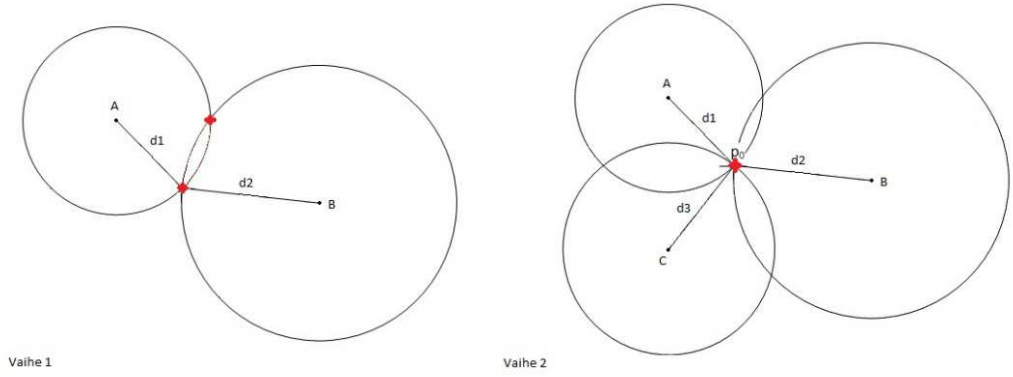
Hallintasegmentti koostuu maanpäällisestä hallintoverkosta, jonka keskusvalvonta-asema sijaitsee Amerikassa Colorado Springsissä. Hallintoverkkoon kuuluu lisäksi 4 miehittämätöntä valvonta-asemaa, jotka sijaitsevat ympäri maailmaa mahdollisimman lähellä päiväntasaajaa. Näiden asemien avulla seurataan ja laskelmoidaan satelliittien sijainteja kiertoradoillaan, niin että käyttäjä segmentin osapuolille lähetettävä signaali on validia. [14, s. 6-8].

Hallintaverkoston avulla voidaan myös suorittaa tarvittavia huoltotoimenpiteitä satelliiteille, jos niiden toiminnoissa ilmenee virheitä. Näin ollen keskusvalvonta-asema on ainoa yksikkö, joka pystyy kommunikoimaan satelliittien kanssa kaksisuuntaisesti. Tämä ilmenee myös kuvasta 9. [14, s. 6-8].

Käyttäjäsegmentti koostuu nimensä mukaisestikin GPS-järjestelmän käyttäjistä, kuten eri liikennevälineiden navigointilaitteet, matkapuhelimet, sekä muut elektroniset laitteet, jotka hyödyntävät paikannusominaisuuksia. Fyysiseltä olemukseltaan GPS-vastaanotin on elektroninen laite, joka on suunniteltu vastaanottamaan GPS-satelliittien lähettämää radiosignaalia. Kyseiseen toimenpiteeseen kuuluu useimmiten myös signaalin tulkitseminen käyttäjälle helpommin ymmärrettävään muotoon, kuten pisteeksi kartalle. [14, s. 6-8].

3.2 Trilateraatio

Trilateraatio on geometrinen laskentamenetelmä, jonka avulla pystytään paikantamaan haluttuja kohteita. Samaa menetelmää käytetään myös GPS-paikannuksessa. Menetelmä perustuu siihen, kun tunnetaan kaksi tai useampi kiintopistettä ja niiden etäisyydet paikannettavasta kohteesta, voidaan laskea paikannettavan kohteen sijainti. Menetelmän hyödyntäminen käytännössä toimii aivan samalla tavalla, mutta esimerkiksi GPS-paikannuksessa kiintopisteet ovat satelliitteja, koska niiden tarkka sijainti tiedetään koko ajan liikkeestä huolimatta. Satelliittien etäisyydet paikannettavasta laitteesta saadaan laskettua signaalien aikaviiveiden avulla. [16].

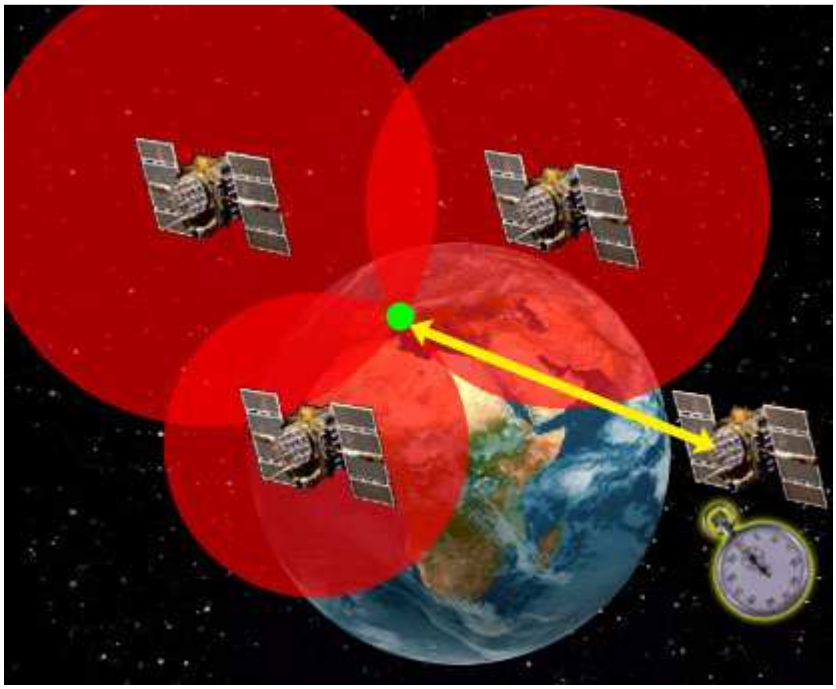


KUVA 11. 2-ulotteinen trilateraatio [16].

Menetelmää voidaan tarkastella 2-ulotteisesta perspektiivistä kuvan 11 mukaisella tavalla. Ensimmäisessä vaiheessa paikannettavalla kohteella on kaksi kiintopistettä A ja B, joiden etäisyydet d1 ja d2 tiedetään. Koska etäisyydet d1 ja d2 pitävät paikkansa jää paikannettavalle kohteelle kaksi mahdollista sijaintia, jotka määrittyvät kuvan ensimmäisen vaiheen ympyröiden leikkauspisteiksi. [16].

Seuraavassa vaiheessa paikannukseen saadaan avuksi kolmas kiintopiste, jonka avulla paikannettavan kohteen sijaintia saadaan tarkennettua yhteen mahdolliseen leikkauspisteeseen. Tämä tarkoittaa myös sitä, että mitä enemmän kiintopisteitä on mahdollista hyödyntää, sitä tarkempaan mittaustulokseen päädytään. [16].

GPS-paikannuksen toimintaperiaate on muuten aivan samanlainen, mutta se tapahtuu 3-ulotteisessa avaruudessa. Kuvassa 12. näytetään, että paikannukseen tarvitaan vähintään kolme satelliittia ja neljännen satelliitin avulla eliminoidaan aikaviiveiden aiheuttamat paikannusvirheet. Näin ollen GPS-paikannustiedon määrittämiseksi tarvitaan vähintään neljä satelliittia, mutta tarkkuus paranee, jos käyttöön saadaan useampia satelliitteja. [16].



KUVA 12. GPS-trilateraatio [17].

3.3 Pseudoetäisyys

Kuten aiemmin mainittu, GPS-järjestelmän perusidea koostuu sijaintitietoa lähettävistä satelliiteista, niitä hallitsevista hallintoyksiköistä ja itse käyttäjistä. Yksinkertaistettuna käyttäjän GPS- vastaanotin kerää satelliiteilta saaman signaalin ja kääntää sen käyttäjän ymmärtämään muotoon, eli pituus- ja leveyskoordinaateiksi.

Matemaattisesti tätä toimenpidettä kutsutaan pseudoetäisyyden määrittämiseksi, joka tapahtuu kaavan (1) mukaisella tavalla. Kyseisen kaavan avulla pystytään laskemaan paikannettavan kohteen tuntemattomat paikkamuuttujat hyödyntäen kiintopisteitä, kuten aiemmassa trilateraatio- esimerkissä. [17].

$$p = \sqrt{(x-X)^2(y-Y)^2(z-Z)^2} + c\,(\Delta t - \Delta T) \qquad (1)$$

Jossa

p = pseudoetäisyys, eli satelliitin etäisyys vastaanottimesta

x,y,z = satelliitin sijainti 3-ulotteisessa avaruudessa

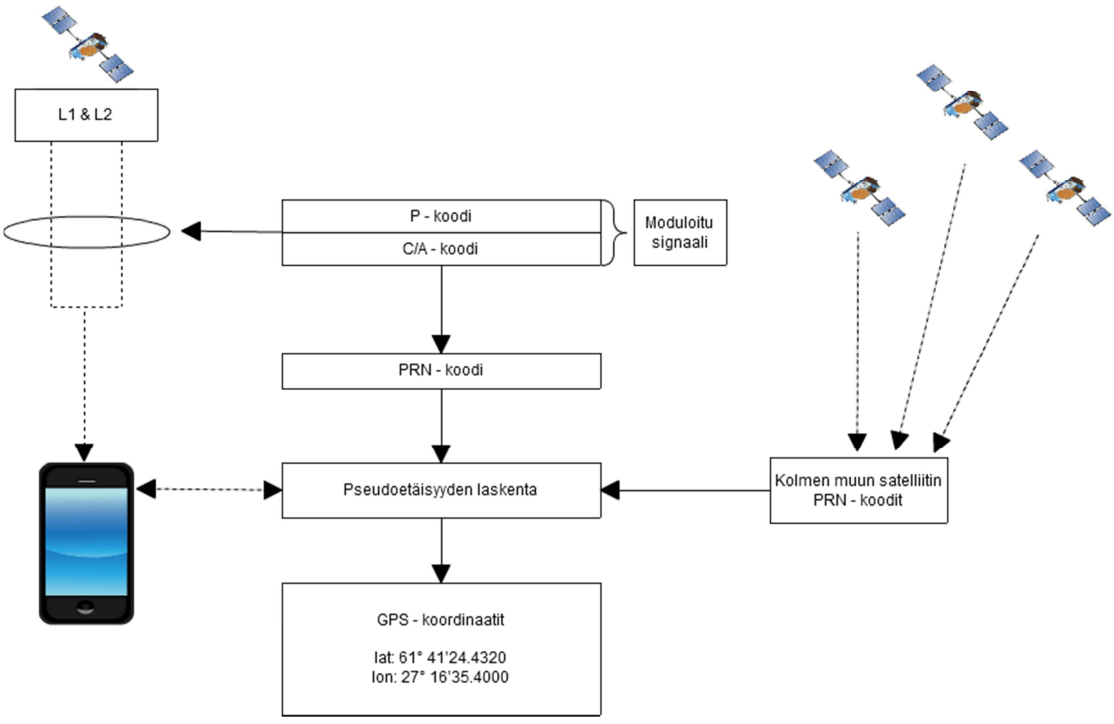
X,Y,Z = vastaanottimen sijainti 3-ulotteisessa avaruudessa

c = valonnopeus

Δt = satelliitin kellovirhe

ΔT = vastaanottimen kellovirhe

Vastaanottimen sijainti saadaan määritettyä muuttujilla X,Y,Z ja ΔT, jotka ovat aluksi tuntemattomia. Niiden ratkaisu vaatii vähintään neljän eri satelliitin lähettämää pseudoetäisyys- dataa, eli PRN- koodia, jotta niiden yhtälöt saadaan laskettua samanaikaisesti. [17].



KUVA 13. GPS-paikannus käytännössä

Oheinen kuva 13. ilmentää satelliittien lähettämän signaalin osatekijöitä ja tulkintavaiheita todenmukaisemmin, kun otetaan huomioon useampia tekijöitä. Jokaisella satelliitilla on kaksi radiolähetintä L1 ja L2, jotka käyttävät radiosignaalin lähettämiseen eri taajuusalueita. L1- lähetin tuottaa 1,575.42MHz- ja L2 1,227.60MHz- signaalia. Hyödyntämällä kahta eri taajuutta saadaan korjattua suurinta GPS-mittauksessa

ilmenevää virhettä, eli ionosfääristä viivettä, joka aiheuttaa radiosignaalin hajautumista ja muuttaa sen nopeutta. [14, s. 8, 13-16].

Itse radiosignaali koostuu kahdesta keskenään moduloidusta koodista, joita ovat P-koodi ja C/A-koodi. Näistä jälkimmäinen on salaamatonta koodia, jota hyödynnetään suurimmassa osassa GPS-laitteista. P-koodi toimii noin 10 kertaa nopeammin kuin C/A koodi, mutta se on salattua ja tulkittavissa vain erikoislaitteilla ja sovelluksilla, kuten esimerkiksi sotilaskäytössä. C/A-koodi muodostuu 1023 bittiä pitkästä lukujonosta, joka toistuu kerran millisekunnissa, eli sen teoreettinen nopeus on tällöin 1,023Mbps. P-koodin nopeus on tällöin noin 10,23Mbps. [14, s. 8, 13-16].

P-koodi ja C/A-koodi muodostavat yhdessä PRN-koodin, jolla tarkoitetaan pseudosatunnaislukujonoa. PRN-koodi on jatkuvasti vaihtuva lukujono, joka vaihtuu satelliitin sijainnin mukaan ja on johdettu tietynlaisilla matemaattisilla algoritmeilla. Näin PRN-koodin avulla pystytään tarkasti tulkitsemaan satelliitin oikea sijainti ja hyödyntämään satelliittia kiinteänä pisteenä paikannuslaitteen sijainnin laskemiseen. GPS-arkkitehtuurin mukaan maanpäällinen hallinto segmentti pitää huolen siitä, että PRN-koodit pitävät paikkansa ja ovat mahdollisimman tarkkoja. [14, s. 8, 13-16].

Kun kaikkien neljän satelliitin lähettämät PRN-koodit on vastaanotettu paikantimeen, pystyy laite laskemaan pseudoetäisyyden kaavaa hyödyntäen sijaintinsa. Sijainnin tarkkuus paranee, jos laskemiseen saadaan mukaan useamman satelliitin lähettämää PRN-koodia. [14, s. 8, 13-16].

4 GSM- JA GPRS-VERKOT

GSM on matkapuhelinjärjestelmä, joka hyödyntää langatonta tiedonsiirtotekniikkaa. Järjestelmä käyttää apunaan digitaalista radioteknologiaa, jonka ensisijaisen tarkoituksena on yhdistää puheluita, sekä muita palveluita muihin kommunikointijärjestelmiin ja päinvastoin. Ensisijaisina asiakaslaitteina käytetään matkapuhelimia. GPRS-järjestelmä on päivitetty GSM-verkko, jonka tarkoituksena on tarjota uusia palveluita GSM-verkon käyttäjille. [17].

4.1 Historia

GSM-verkoilla ja niiden kehityksellä on tärkeä osa yhteiskunnan hyödyntämien tiedonsiirtotekniikoiden keskuudessa. GSM-verkkoja hyödynnettiin aluksi pelkästään langattomien matkapuhelinten yhdistämisessä toisiinsa, sekä kiinteään lankapuhelinverkkoon, mutta kehityksen edistyessä verkkoja on jouduttu laajentamaan entisestään vastaamaan nykypäivän tiedonsiirtovaatimuksia ja erilaisia palveluita. [17].

Kun käydään läpi langattoman tiedonsiirron kehitystä, on hyvä kerrata hieman eri tekniikoita. Matkapuhelinverkkojen kehittymistä ilmaistaan ns. uusilla sukupolvilla, kuten taulukossa 1 esitetään. Verkkojen ominaisuudet, suojaukset ja nopeudet ovat kehittyneet jatkuvasti uusien sukupolvien myötä paremmiksi. Sukupolvien kehitys tapahtuu päivittämällä verkkojen arkkitehtuureja ja lisäämällä uusia fyysisiä tai loogisia palveluita verkkojen rakenteeseen. [17].

TAULUKKO 1. Matkapuhelinverkkojen sukupolvet. [18, s. 1-7 & 20.]

	1G	2G	2.5G	3G	4G
Tekniikat	NMT, AMPS	GSM, CDMA	GPRS, EDGE	3G, UMTS, WCDMA	4G, Mobile IP
Ominaisuudet	Analoginen	Digitaalinen	Päivitetty 2G	Internet pohjainen	Internet pohjainen
Hyödyntäminen	Vain ääni	Ääni ja data	Ääni, data internet	Ääni, data, video, internet	Ääni, data, video, internet
Nopeus	Alhainen	10-14Kbps	50-144Kbps	144Kbps-2Mbps+	100 Mbps

Taulukossa 1 ilmoitettavat nopeudet ovat vain perustekniikoiden perusominaisuuksia ja maksiminopeuksia. 2G verkoilla pystyy hyödyntämään internet-palveluita, koska verkkojen ominaisuuksia ja rakenteita on paranneltu. 3G verkko voi tukea mm. myös DC-HSDPA verkkoteknologiaa, jonka maksiminopeus yltää jopa 42 Mbps suuruiseksi. Nopeudet riippuvat myös maanpäällisestä sijainnista, koska vastaanottimien kantoalueet ovat rajallisia, etenkin suuremmilla datanopeuksilla. Tavallisella GSM-verkolla on melkein koko Suomen kattava peittoalue, mutta nopeampia verkkoteknologioita ei

pysty hyödyntämään joka paikassa, koska kaikkia tukiasemia ei kannata päivittää taloudellisesta syistä johtuen. [18, s. 1-7 & 20.]

Suomessa otettiin käyttöön ensimmäiseksi valtakunnalliseksi radiopuhelinverkoksi NMT, eli pohjoismainen radiopuhelinverkko vuonna 1982. NMT-verkko toimi 450MHz taajuusalueella ja kapasiteetin loppuessa otettiin käyttöön 900MHz NMT verkko vuonna 1987. [19].

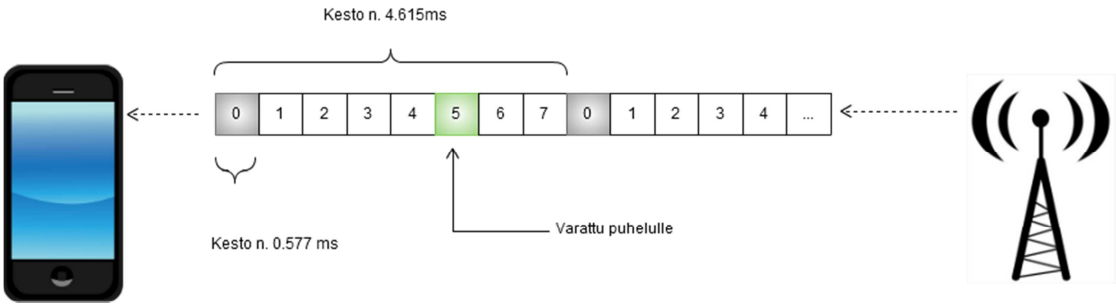
NMT-verkko oli suomessa ensimmäinen laatuaan oleva langaton radiopuhelinverkko, jonka avulla pystyi soittamaan ja vastaanottamaan puheluita kannettavien NMT-puhelinten ja kiinteiden lankaverkkojen välillä. Kun verrataan nykyisiin tekniikoihin, NMT-verkko oli toiminnaltaan analoginen, eli lähetettävä ja vastaanotettava signaali säilyi koko ajan samanmuotoisena. Nykyiset puhelinverkkotekniikat ovat digitaalisia, joka mahdollistaa mm. signaalien korjaamisen ja salaamisen. NMT-verkot lakkautettiin vuonna 2002 GSM:n yleistyessä. [19].

NMT-teknologian seuraaja oli digitaalinen puhelinverkko GSM, joka toimii pääasiallisesti 900/1800 MHz taajuusalueilla. GSM-verkko otettiin käyttöön Suomessa vuonna 1991 ja verkko onkin tähän mennessä levinnyt miltei maailmanlaajuisesti käytettäväksi. GSM-järjestelmä on rakennettu täysin automaattiseksi, joka tarkoittaa sitä, että matkustettaessa toiseen maahan, jossa oma palveluntarjoaja ei toimi, puhelin osaa rekisteröityä automaattisesti toisen palveluntarjoajan verkkoon ja hyödyntää sitä. [19].

4.2 Radioliikenne

GSM-verkon radioyhteys toteutetaan TDMA- ja FDMA- tekniikoilla, jotka mahdollistavat tukiasemien lähetys- ja vastaanottotaajuuksien pilkkomisen useampaan osaan. Tällä tavoin mahdollistetaan useamman eri käyttäjän verkkoliikenne samalla taajuusalueella samanaikaisesti. [21, s. 13-18 & 22. s. 4-18.]

GSM-verkon tapauksessa käytettävä taajuusalue on jaettu 200kHz kokoisiin kanaviin ja jokainen kanava muodostuu kahdeksasta aikavälistä. Tavallisen GSM-puhelun alussa otetaan käyttöön yksi aikaväli, jonka teoreettinen tiedonsiirtonopeus on 9.6 kbps. Yhden aikavälin avulla pystytään ylläpitämään jatkuvaa puheyhteyttä toiseen matkapuhelimeen. [21, s. 13-18 & 22. s. 4-18.]



KUVA 14. TDMA-kehys. [21, s. 13-18 & 22. s. 4-18.]

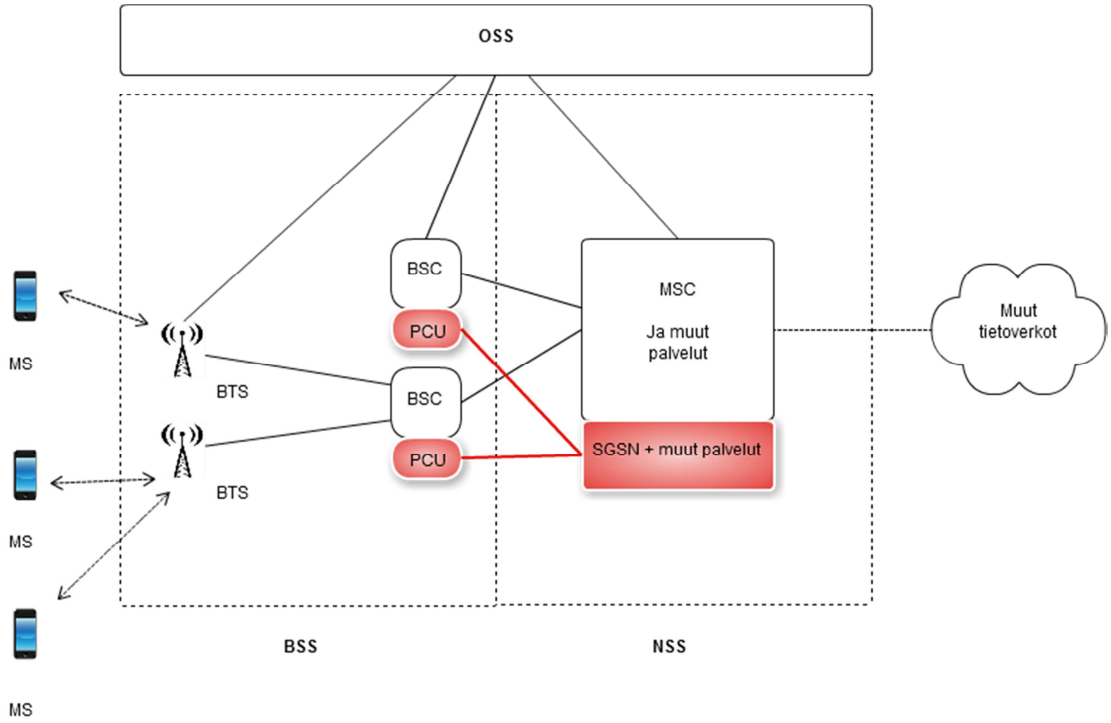
Kuvassa 14 havainnollistetaan GSM-verkon radioliikennettä tukiaseman ja matkapuhelimen välillä. Liikennettä voidaan ajatella jatkuvana ratana, jossa toistuu aikavälit 0-7. Kun aloitetaan esimerkiksi puhelu, niin tukiasema valitsee ja rekisteröi puhelun tarvitseman tiedonsiirron jollekin tietylle aikavälille. Kuvassa esitetty aikaväli 5 on varattu puhelulle. Tästä lähtien jatkuvan radiosignaalin aikavälit 5 kuljettavat pelkästään siihen rekisteröidyn puhelimen puheliikennettä, ellei aikaväliä joudu jostain syystä vaihtamaan toiseen. Tämä jatkuu niin kauan kunnes puhelu lopetetaan ja tukiasema rekisteröi aikavälin taas vapaaksi muuhun käyttöön. Harmaalla värjätyt aikavälit 0 ovat aina varattu ns. BCH-aikaväleiksi, joita käytetään järjestelmätietojen lähettämiseen. BCH-aikavälit auttavat mm. tietojen synkronisoinnissa ja aikaviiveiden korjauksissa. [21, s. 13-18 & 22. s. 4-18.]

4.3 Arkkitehtuuri ja toimintamalli

Yksinkertaistetusti GSM-matkapuhelinverkko toimii seuraavalla tavalla: matkapuhelimesta lähetettävä data, viesti tai puhelu muunnetaan ensin digitaaliseen muotoon ja siirretään langattomasti radiotaajuuksia pitkin tukiasemaan, jonka kautta ne siirtyvät haluttuun määränpäähänsä eri tunnisteiden ja palveluiden avulla.

Kun vertaillaan GSM- ja GPRS-verkon toimintaa, niin yksi merkittävimmistä eroista on se, että GSM-verkko toimii piirikytkentäisesti ja GPRS verkko toimii pakettikytkentäisesti kuten internet. Käytännössä tämä tarkoittaa sitä, että piirikytkentäinen verkko veloittaa käytetyn ajan mukaisesti, kun taas pakettikytkentäinen veloittaa siirretyn datamäärän mukaisesti. GPRS-verkko on rakennettu GSM-arkkitehtuurin päälle, joten puhelut pystyvät siirtymään vain piirikytkentäisesti. GSM verkon perusarkkitehtuuri perustuu kuvan 15 esittämiin neljään eri elementtiin, jotka mahdollistavat päätelaitteiden kommunikoinnin verkon yli toisten laitteiden kanssa. Ne mahdollistavat

myös käyttäjän vapaan liikkuvuuden tukiasemien kantoalueella, ilman että yhteys katkeaisi missään vaiheessa. [24, s. 18-31 & 25.]



KUVA 15. GSM- ja GPRS-verkkoarkkitehtuuri. [24, s. 18-31 & 25.]

MS tunnetaan yleisimmin matkapuhelimina tai muina päätelaitteina. Laitteet koostuvat fyysisestä elektronisesta laitteesta ja siihen määritellystä SIM:istä. Fyysinen laite tarvitsee tunnistusta varten yksilöllisen IMEI-tunnuksen, jonka avulla fyysiset laitteet yksilöllistetään ja tunnistetaan verkossa. Jokaiselle matkapuhelimelle on jo rakennusvaiheessa määritelty yksilöllinen IMEI- tunnus. SIM on yleisimmin matkapuhelimissa käytetty SIM- kortti, joka sisältää käyttäjän valitseman puhelinverkon tiedot. Jos käyttäjä haluaa kytkeytyä esimerkiksi DNA:n puhelinverkkoon, tarvitaan puhelimeen DNA:n valmistama SIM-kortti, joka sisältää tarvittavat verkkotiedot. Näin ollen SIM-kortin avulla osataan käyttäjä tunnistaa ja veloittaa verkon käytön mukaan. Puhelinverkkoon kytkeytyminen ei ole mahdollista ilman kyseisiä SIM- ja IMEI-tunnisteita. [24, s. 18-31 & 25.]

BSS on järjestelmä, jonka avulla toteutetaan matkapuhelinten verkkoliikennöinti langattomasti matkapuhelinten ja verkon tukiasemien välillä. Järjestelmä koostuu pääosin kahdesta eri elementistä, joita ovat BTS ja BSC. BTS kattaa verkon tukiasemien fyysisen laitteiston, kuten lähettimet, vastaanottimet ja niille tarkoitetut antennit. BTS myös valitsee kullekin verkkoyhteydelle omat radiosolunsa. BSC on kontrolliyksikkö,

joka hallinnoi BTS ryhmiä ja säätelee niiden toimintaa. Tässäkin tapauksessa puhutaan fyysisistä laitteista. [24, s. 18-31 & 25.]

NSS on järjestelmä, joka hallitsee useita erilaisia verkon elementtejä, joita tarvitaan mm. käyttäjien autentikoimiseen, datayhteyksien jälleenohjaukseen ja muuhun liikennöintiin. NSS:ää kutsutaan myös ns. verkon ytimeksi. MSC on yksi tärkeimmistä elementeistä, joka ohjaa puhelut puhelinnumeroiden avulla oikeisiin paikkoihinsa. [24, s. 18-31 & 25.]

OSS on elementti, joka on kytketty BSS- ja NSS- järjestelmien yhteyteen. Sen avulla hallinnoidaan ja tarkkaillaan verkon liikennettä ja kuormaa. OSS:n avulla voidaan tarvittaessa suorittaa myös erilaisia huoltotoimenpiteitä puhelinverkolle. [24, s. 18-31 & 25.]

Muut tietoverkot -osio ilmaisee käsitteellistä kokoelmaa erilaisista verkoista, joihin on mahdollista kytkeytyä GSM/GPRS verkon kautta. Tässä tapauksessa hyvä esimerkki on internetpalveluihin pääsy GSM-verkon kautta hyödyntäen GPRS-palvelua. [24, s. 18-31 & 25.]

4.4 GPRS

GPRS on GSM-tekniikan päälle rakennettu palvelu, eli toisin sanoen kyseessä on päivitetty GSM-verkon arkkitehtuurirakenne. GSM-verkon perusarkkitehtuuria on muutettu mm. ohjelmistopäivityksillä ja uusilla fyysisillä laitteilla, jotka mahdollistavat esimerkiksi internetpalveluiden käytön matkapuhelimen välityksellä. [24, s. 18-31 & 26, s. 5-8, 13-19, 24-29.]

Tästä voidaan päätellä, että GSM-verkon tarjoama yksi aikaväli ei nopeudeltaan ole riittävä tai tehokas internetpalveluiden ylläpitämiseen. Näin ollen GPRS-palvelu vaatii rakenteellisia muutoksia GSM-verkkoon ja sen radioliikenteeseen. Tästä johtuen GPRS-palvelu mahdollistaa radiosignaalin useamman aikavälin samanaikaisen käytön. Teoreettisesti GPRS- palvelu pystyykin samanaikaisesti ottamaan 200 kHz:n radiokanavan kaikki kahdeksan aikaväliä käyttöönsä. Tämä tarkoittaisi jopa 115 kbps tiedonsiirtonopeutta, mutta käytännössä nopeus yltää kuitenkin vain noin 50 kbps tasolle, sillä joitakin aikapaikkoja tarvitaan eri toimenpiteisiin. Kuvan 15 punaisella

merkityt palikat ilmaisevat GSM-arkkitehtuurin yhteyteen rakennettuja lisäpalveluita, jotka mahdollistavat GPRS-verkon toiminnan GSM-verkon ohella. [24, s. 18-31 & 26, s. 5-8, 13-19, 24-29.]

PCU on fyysinen laitteisto, jonka avulla radiotaajuuksien kautta pystytään ylläpitämään pakettikytkentäistä liikennettä GPRS-protokollan mukaisesti. Käytännössä tämä vaatii radiosignaalin kehysrakenteen muuttamista GPRS-verkkoa varten sopivammaksi. [24, s. 18-31 & 26, s. 5-8, 13-19, 24-29.]

SGSN on eräänlainen solmurakenne GPRS-arkkitehtuurissa, joka auttaa pakettikytkentäisen dataliikenteen ohjauksessa. Periaatteessa SGSN palvelee samaa asiaa kuin MSC, mutta tiedon ohjaus kohdistuu GPRS-palvelun vaatimalle pakettiliikenteelle. [24, s. 18-31 & 26, s. 5-8, 13-19, 24-29.]

Kuvassa 15 mainitut ”muut palvelut” sisältävät lukuisia erilaisia palveluita ja laitteita, joita vaaditaan verkon ylläpitoon. Kuva esittää hyvin summittaisesti verkon kokonaisrakennetta, sillä esille on tuotu vain tärkeimpiä elementtejä yksinkertaistetussa muodossa, jotta perusrakenteen ymmärtäminen olisi helpompaa. Muut palvelut sisältävät paljon erilaisia lisäpalveluita, joiden avulla suoritetaan mm. data- ja puheliikenteen salausta, käyttäjien rekisteröintiä, autentikointia, virheiden korjausta ym. [24, s. 18-31 & 26, s. 5-8, 13-19, 24-29.]

5 WINDOWS PHONE 7 SDK

Windows Phone on Microsoftin julkistama ja mobiililaitteille suunniteltu käyttöjärjestelmä. Käyttöjärjestelmä on jatkoa vanhemmalle Windows Mobile-järjestelmälle. Vanhemman järjestelmän sovellukset eivät ole yhteensopivia uudemman kanssa, koska käyttöjärjestelmä on pitkälti kirjoitettu uudestaan. Windows Phone-laitteille tapahtuva ohjelmointi tapahtuu .NET-pohjaisilla XNA- ja Silverlight-tekniikoilla. [27, s. 12-16].

Microsoft.NET ympäristö muistuttaa paljon Javaa ja vastaa hyvin nykypäivän asettamia vaatimuksia, etenkin sen suhteellisen helpolla omaksuttavuudella käyttöliittymän ja muiden palveluiden suhteen. .NET-pohjaiset kirjastot ovat suuri etu sovelluskehityksen kannalta, koska ne tarjoavat laajan kehitysympäristön vaativampiinkin sovel-

luksiin ja takaavat paremman yhteensopivuuden muun Windows ympäristön kanssa. Normaalisti pelkästään puhelinsovellusten kehitykseen tarjottavat kirjastot ja ohjelmointikielet ovat melko suppeita ja mallikohtaisia. Sovellusten kehittämiseen tuetut ohjelmointikielet ovat tällä hetkellä yleisimmin käytetty C# ja Visual Basic. [27, s. 12-16].

Microsoft tarjoaa sovelluskehitystä varten maksuttomat perusvälineet: Visual Studio, ExpressionBlend ja XNA-Game Studio sovellukset. Visual Studio on IDE-pohjainen kehitystyökalu, jossa käytettävänä on koodieditori, visuaalinen suunnittelutila käyttöliittymiä varten, sekä virheenkorjaustyökalu. ExpressionBlend on oivallinen laajennus Visual Studion lisäksi, jonka avulla pystytään suunnittelemaan Silverlight- tekniikkaa hyödyntäviä graafisia ulkoasuja sovelluksille. Ohjelmien visuaalinen tärkeys on kasvanut merkittäväksi tekijäksi käytettävyyden kannalta, joten sovelluskehittimet vastaavat hyvin perustarpeisiin. XNA-Game Studio on tarkoitettu ensisijaisesti pelisovellusten suunnittelua varten Windows puhelimiin ja sen avulla pystytään myös hyödyntämään Microsoftin tarjoamia Xbox-live palveluita peleissä. [27, s. 12-16].

Kehitystyökaluihin kuuluu myös Windows Phone-emulaattori, jonka avulla itse rakennettuja ohjelmia pystyy testaamaan ja ”debuggaamaan”, eli paikantamaan virheitä lennosta. Emulaattori kuuluu Windows Phone-kehitystyökaluihin ja sen avulla pystyy testaamaan liki kaikkia puhelinkäyttöjärjestelmän tarjoamia toimintoja, poislukien puheluiden soittamisen, sekä tekstiviestien lähettämisen. [27, s. 12-16].

Sovelluksia ei pysty ainakaan vielä siirtämään suoraan omaan älypuhelimeen, vaan omien ohjelmien käyttämistä varten vaaditaan rekisteröityminen Windows Phone Marketplace-palveluun, jota kautta kaikki muutkin Windows Phone-puhelimille ilmestyvät sovellukset ladataan. Rekisteröityminen maksaa tällä hetkellä noin 80 euroa vuoden ajaksi ja mahdollistaa omien sovellusten kehittämisen, julkaisemisen ja myynnin. Kaikki Marketplacessa julkaistavat sovellukset tarkistetaan ennen niiden julkaisua, sillä sovellusten täytyy täyttää tietynlaiset laatukriteerit. Periaatteessa näin varmistetaan, ettei sovelluksiin asenneta haittaohjelmia ja puhelimien suorituskyky säilyy tietyissä rajoissa. [27, s. 12-16].

5.1 Windows Phone laitteiden minimivaatimukset

Microsoft on asettanut tarkat rajoitteet puhelinvalmistajille laitteiden minimivaatimusten suhteen, jotta Windows Phone-käyttöjärjestelmän käyttämiseen saa luvan. Kyseiset minimivaatimukset esitetään taulukossa 2. Kyseisistä vaatimuksista on myös paljon hyötyä, sillä käyttäjäkokemus saadaan pidettyä mahdollisimman samankaltaisena laitteiden mallista tai valmistajasta riippumatta. Myös sovelluskehitys helpottuu huomattavasti, koska samat sovellukset soveltuvat kaikkiin puhelimiin mallista tai näyttöjen koosta riippumatta. [27, s. 51].

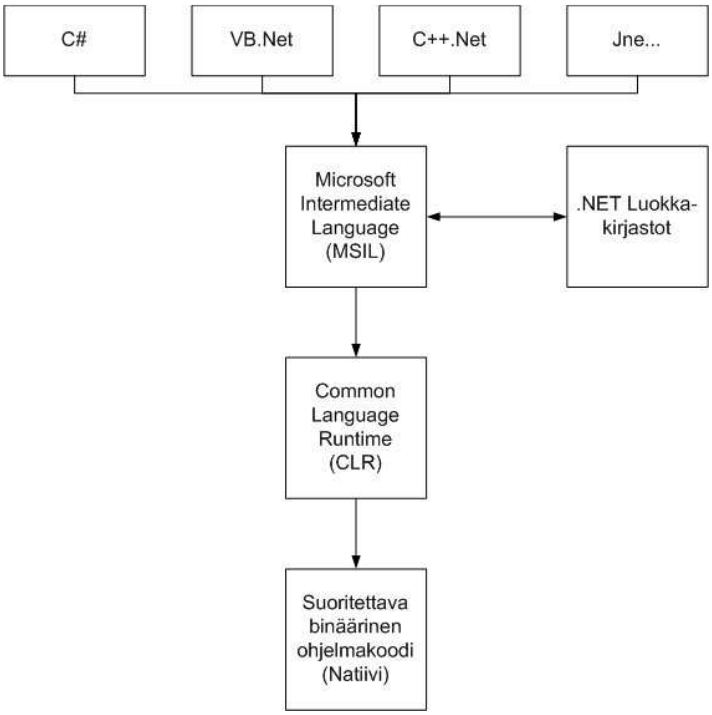
TAULUKKO 2. Windows Phone-laitteiden minimivaatimukset [27, s. 51].

Ominaisuus	**Minimivaatimukset**
Suoritin	ARM-pohjainen v7-suoritin, kellotaajuus vähintään 1GHz.
Muisti	Vähintään 256MB, Flash muistia vähintään 8GB
Grafiikkasuoritin	DirectX 9-laitteistokiihdytystä tukeva.
Näyttö	Wide VGA-tarkkuus (800x480 pikseliä), vähintään neljän pisteen kapasitiivinen kosketustuki.
Verkkoyhteydet	GSM/CDMA markkinoista riippuen, GPRS ja 3G. Lisäksi WiFi eli WLAN.
Sensorit	Kiihtyvyysanturi, kompassi, vallitsevan valon mittari, läheisyystunnistin, sekä A-GPS.
Kamera	Vähintään viisi megapikseliä.
Muuta	Kolme erillistä laitteistonäppäintä: Home, Back ja Search. FM-radio.

5.2 .NET-ympäristö

.NET on Microsoftin kehittämä erilaisissa ohjelmistoympäristöissä käytettävä ohjelmistokomponenttikirjasto, joka tukee n. 20 erilaista ohjelmointikieltä. Sen avulla pystytään toteuttamaan moninaisia ohjelmakokonaisuuksia, jotka toimivat pääasiassa

Windows ympäristössä. Ympäristöllä on laaja tuki erilaisille komponenteille, kuten esim. internet-, tai konsolipohjaisille sovelluksille. [27, s. 17-20].



KUVA 16. .NET-ympäristön toimintamalli

Kuvan 16 mukaisesti, kyseisessä ympäristössä tapahtuva ohjelmointi voidaan suorittaa erilaisilla ohjelmointikielillä, jonka jälkeen ohjelma esikäännetään MSIL-palvelun avulla hyödyntäen .NET-ympäristön luokkakirjastoja. Sen jälkeen eräänlaisena virtuaalikoneen toimiva CLR-palvelu muuntaa esikäännetyn koodin binääriseen muotoon, eli ns. natiivikoodiksi, jonka tietokoneet ja muut laitteet osaavat suorittaa laitteistotasolla. CLR-palvelu sisältää erilaisia lisäpalveluita, jotka vaikuttavat ohjelmien tietoturvaan, sekä laitteiston hallintaan liittyviin elementteihin. [27, s. 17-20].

Windows Phone-ympäristössä tapahtuva ohjelmointi toteutetaan ensisijaisesti C#-kielellä, mutta myös Visual Basic-kieli on tuettuna. Ohjelmien kääntäminen tapahtuu .NET-ympäristön sääntöjen mukaisesti, kuten ylempänäkin kerrottiin. Windows Phone-ympäristö hyödyntää laajemman .NET-luokkakirjaston osajoukkoa, mutta ympäristöä varten on myös luokkia, jotka ovat suunniteltu pelkästään puhelimia varten. [27, s. 17-20].

5.3 Silverlight ja XNA-Game Studio

Microsoftin kehittämä Silverlight-tekniikka on Adobe Flash- tyylinen internet-pohjainen ohjelmointiympäristö, joka suunniteltiin alunpitäen internet-selaimiin liitettäväksi videoiden suoratoisto-sovellukseksi. Ajan myötä tekniikkaan on lisätty erilaisia lisäominaisuuksia, tärkeimpänä mainiten tuen .NET kielille. Käyttöliittymän animaatioita ja vektoreita kuvataan XAML-kielellä, jonka avulla on suhteellisen helppoa määritellä erilaisia visuaalisia objekteja, animaatioita ja niiden toimintaa muiden ohjelmointikielten kanssa. [27, s. 96-104].

Silverlight-tekniikkaa käytetään Windows Phone-käyttöjärjestelmissä, mutta ei aivan sellaisenaan. *Silverlight for Windows Phone* on kokonaisvaltaisesta Silverlight-tekniikasta hieman karsitumpi versio, mutta lisäominaisuuksia kehitetään jatkuvasti ja Windows Phone-alustalle suunnattu tekniikka tarjoaa hallitumman käyttöliittymän kehittäjien kannalta. [27, s. 96-104].

Mobiilipeleillä on suuri jalansija nykyisissä puhelimissa, joten myös niitä varten on suunniteltu parempia kehitystyökaluja. Windows Phone-kehitystyökaluihin kuuluu XNA-Game Studio sovellus, joka on ensisijaisesti suunniteltu toteuttamaan erilaisia pelisovelluksia Windows Phone-alustoille. Silverlight- tekniikalla pystytään toteuttamaan monipuolisia visuaalisia ja interaktiivisia sovelluksia, mutta sellaisenaan se ei sovellu kovin hyvin pelien kehittämiseen. [27, s. 188].

XNA-Game Studio sisältää lukuisia tuttuja pelikoodielementtejä, kuten GameLoop, pelisilmukan, joka kuuluu pelien kehittämisen peruselementteihin. Ohjelmakirjastot ovat myös päivitetty lukuisilla matematiikka- ja fysiikkakirjastoilla, joka parantaa ja nopeuttaa erilaisten 3D-ominaisuuksien hyödyntämistä ja käyttöönottoa. [27, s. 188].

6 RUUVITRACKER-JÄRJESTELMÄ

RuuviTracker on vapaaseen lähdekoodiin perustuva GPS-paikannusjärjestelmä, joka koostuu fyysisestä laitteesta, sille ohjelmoidusta *firmware* ohjelmistosta, sekä laitteiden tiedonsiirtoa tukevasta palvelinympäristöstä. RuuviTracker- palvelimelle on myös suunniteltu oma karttakäyttöliittymä, jota voi katsella tavallisen internet-selaimen

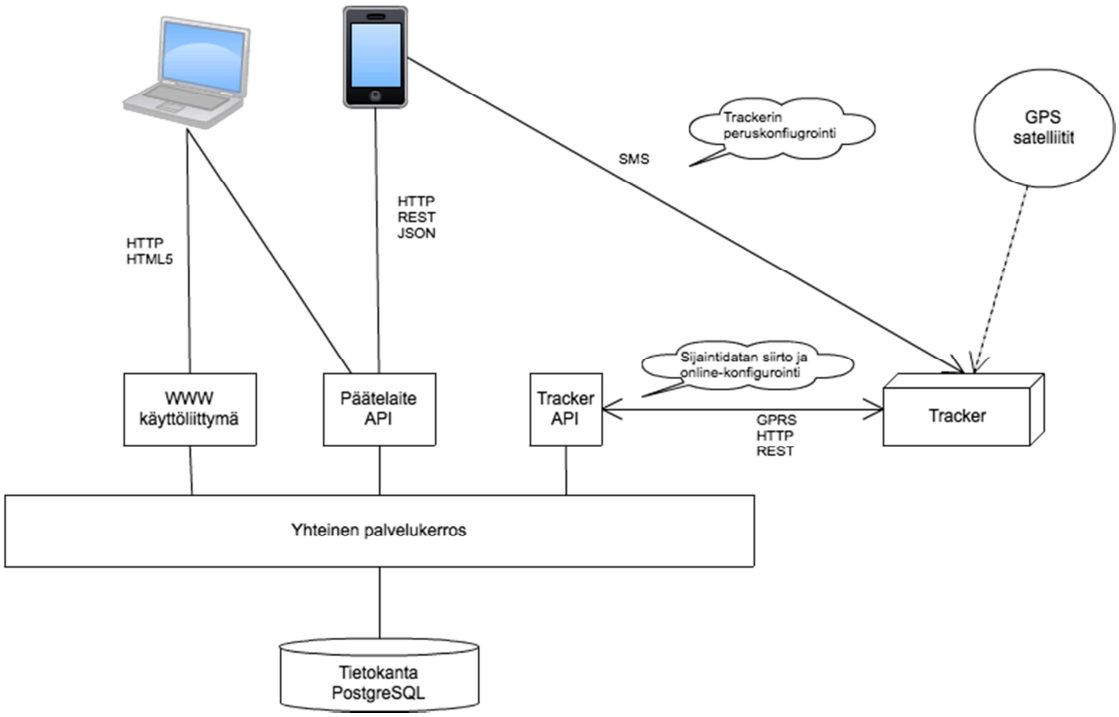
avulla. Tiedonsiirto RuuviTracker- paikantimien ja palvelimen välillä toteutetaan langattomasti hyödyntäen GSM- ja GPRS- verkkojen toimintaa. [40 & 41 & 47.]

Laite on rakenteeltaan hyvin pienikokoinen, vain noin 5cm x 4cm, mutta sisältää mm. GPS/GSM/GPRS moduulin, kiihtyvyysanturin, gyroskoopin, kompassin, SIM/micro SD korttipaikan, sekä ARM Cortex M4- mikrokontrollerin. Komponenttien valinnoissa on otettu huomioon hyvä hinta-laatusuhde, sekä niiden soveltuvuus Suomen sääoloihin. Paikantimelle onkin luvattu toiminnallisuus jopa -40 pakkasasteessa. [40 & 41 & 47.]

Käytännössä RuuviTracker-laite on eräänlainen elektroniikan kehitysalusta, koska kaikki laitteeseen liittyvät kaaviot ja ohjelmakoodit ovat vapaita, sekä kaikkien saatavilla. Laite sisältää myös laajennuspaikkoja, jotka mahdollistavat lisälaitteiden käytön. Laitetta on siis mahdollista muokata haluamallaan tavalla ja liittää esimerkiksi laitteen yhteyteen erilaisista antureista koostuvan laajennuslevyn, jolloin laitetta pystyy hyödyntämään pienen ohjelmoinnin avulla etäluettavana sää-asemana. [40 & 41 & 47.]

6.1 RuuviTracker-laitteen arkkitehtuuri

RuuviTracker-laitteen toimintaa on helppo kuvata alla olevalla arkkitehtuuri kuvalla 17. Kuvassa esitetään kaikki RuuviTracker- järjestelmän vaatimat elementit, sekä sovellusten hyödyntämät rajapinnat, jotka vaaditaan kokonaisvaltaisen toiminnallisuuden aikaansaamiseksi.



KUVA 17. RuuviTracker-järjestelmän yleisarkkitehtuuri

Arkkitehtuurin ensimmäinen toimintavaihe on RuuviTracker-laitteen itsenäinen toiminta. Laite suorittaa käynnistyessään oman *firmware* ohjelmansa, eli laitteen ARM-Cortex M4- mikro-ohjaimelle asennetun ohjelmakoodin. *Firmware* määrittää sen, mitä laite tekee milloinkin. Kun seurataan arkkitehtuurin mallia, niin voidaan todeta, että ensimmäisenä laitteen täytyy saada kelvollinen yhteys GPS-satelliitteihin. Kun satelliittiyhteys on muodostettu, niin voidaan laskea paikannusdataa laitteen sijainnista, toisin sanoen koordinaatteja. Koordinaattitiedot täytyy siirtää langattomasti erilliselle palvelimelle, mutta sitä varten laitteen täytyy muodostaa pakettikytkentäinen GPRS-yhteys datan siirtoa varten. [40 & 41 & 47.]

Kun GPRS-yhteys on muodostettu, niin laite yhdistää automaattisesti RuuviTrackerin omalle palvelimelle. Palvelimelle on määritelty oma Tracker API-rajapinta, jonka avulla paikantimet pystyvät kommunikoimaan palvelimen kanssa ja päinvastoin. Palvelin vastaanottaa paikantimien lähettämät paikannustiedot ja tallettaa ne hallitsemaansa tietokantaan. Tietokannan avulla pystytään hallitsemaan useiden eri käyttäjien tietoja samanaikaisesti. [40 & 41 & 47.]

Kun halutaan lukea paikantimen lähettämiä paikannustietoja, niin tällöin tiedot täytyy noutaa serveriltä. Serverille on määritelty mm. oma internet käyttöliittymä, joka tarkoittaa sitä, että paikannuslaitteiden tietoja pystyy tarkkailemaan aivan tavallisella

internet-selaimella. Serverille on myös laadittu toinen rajapinta, joka tukee mm. JSON-tiedonsiirtoa. Kyseinen tiedonsiirtomuoto on hyvin laajalti tuettu ja se onkin ensisijaisesti suunniteltu tietojen siirtämiseen palvelimen ja internet-sovellusten välillä. [40 & 41 & 47.]

Järjestelmän toiminta noudattaa ns. REST-arkkitehtuuria, jolla pyritään yksinkertaiseen toimintamalliin, jossa kaikki komponentit toimivat keskenään tiettyjen rajoitteiden ja sääntöjen mukaisesti. RuuviTracker-järjestelmän kannalta huomioon otettavia seikkoja ovat REST-arkkitehtuurin asiakas-palvelin-, sekä tilattomuus- ominaisuudet. Asiakas-palvelin- rajoitteella pyritään jakamaan toiminnalliset vastuut eri osapuolille. Asiakas kutsuu palvelinta ja palvelin kuuntelee kutsuja, tämän jälkeen palvelin muodostaa vastaukset kutsujen mukaisesti. Tilattomuudella viitataan palvelimelle lähetettävien käskyjen muotoon. Käskyjen täytyy sisältää kaikki tarvittava tieto, että palvelin osaa muodostaa siihen oikean vastauksen. [40 & 41 & 47.]

Pähkinänkuoressa: paikannuslaite yhdistää GPS-satelliitteihin ja laskee oman sijaintinsa koordinaatit. Muodostetaan GPRS-yhteys, jotta saadaan siirrettyä koordinaatit erilliselle serverille. Koordinaatit voidaan noutaa serveriltä esimerkiksi tietokoneen internet-selaimen kautta, tai matkapuhelimille suunnitelluilla sovelluksilla JSON-tiedonsiirtomuodossa. [40 & 41 & 47.]

6.2 Idea ja suunnittelu

Projekti sai alkunsa Ruuvipenkki.fi verkkosivun ylläpitäjän Lauri Jämsän aloitteesta, kun haluttiin etsiä vaihtoehtoista paikannusjärjestelmää nykyisten tilalle. Markkinoilla oli tarjolla lukuisia vastaavia laitteita ja järjestelmiä, joiden avulla pystyy paikantamaan kohteita ja siirtämään paikannustiedot langattomasti toisille laitteille luettavaksi. Nämä järjestelmät ovat kuitenkin kaupallisia ja toimivat erilaisten lisenssien alaisina, joten laitteet ovat kalliita ja niiden ohjelmistojen käyttäminen on usein maksullista. [43].

Tämän innoittamana Lauri keräsi elektroniikan ja ohjelmoinnin harrastelijoita mukaan suunnittelemaan vapaaseen lähdekoodiin perustuvaa GPS-järjestelmää, jonka käyttö ja ohjelmat olisivat ilmaisia kaikille haluaville. Projekti on toiminut alusta alkaen harrastelijapohjalta, eli kaikki projektiin tehty työ on vapaaehtoista, eikä siitä saa rahallista

korvausta. Tästä huolimatta projekti on saanut paljon mediallista huomiota ja saanut taakseen useamman sponsorinkin, jotka ovat auttaneet projektin kehityksessä. [43].

Laitteessa käytettävä elektroniikka koottiin yhteen eri käyttäjien ideoiden perusteella ja lopullisen kokoonpanon ja piirilevysuunnittelun teki Lauri Jämsä. Keskeisimpänä komponenttina mainittakoon SIMComin valmistamaa SIM908 moduulia, jossa on sisäänrakennettuna GPS-, GSM- ja GPRS-ominaisuudet. [43].

6.3 RuuviTracker- palvelinympäristö

RuuviTracker-palvelinsovellus on toteutettu dynaamisella Clojure-ohjelmointikielellä, jonka avulla voidaan mm. toteuttaa yksinkertainen palvelin-applikaatio. Toiminnaltaan palvelin muodostuu yksinkertaisesta silmukasta, joka on aktiivisessa tilassa niin kauan, kuin itse palvelinkin. Palvelinsovellukseen määritellyt portit ottavat vastaan käskyjä, joihin sitten lähetetään vastaukset Clojure-ohjelmakoodin mukaisesti. Tämä tarkoittaa sitä, että itse ohjelmakoodi on irrallaan suoritus alustasta ja koodi voidaan suorittaa kaikilla Clojure-kieltä tukevissa sovelluksissa. [42 & 44 & 45 & 46.]

RuuviTracker-palvelinohjelmiston tapauksessa Clojure-koodi voidaan suorittaa Jetty-ympäristöllä tai esim. Heroku-pilvipalvelulla. Kyseiset palvelut toimivat ns. ohjelmasäiliöinä, jotka mahdollistavat Clojure-koodin suorittamisen suljetummassa ympäristössä. Tällöin palvelinliikenne on turvallisempaa ja palveluihin on suhteellisen helppoa lisätä uusia toimintoja. [42 & 44 & 45 & 46.]

Palvelinympäristön tietokanta toteutetaan PostgreSQL-nimisellä toteutuksella, joka perustuu myös avoimeen lähdekoodiin. Tämä tarkoittaa sitä, että kaikki tietokannan ominaisuudet ovat suhteellisen helposti muokattavissa ja käyttölisenssit ovat joustavia. Tietokanta on myös hyvin yhteensopiva useiden eri ohjelmointikielien ja käyttöjärjestelmien kanssa. [48].

RuuviTracker-palvelimen tapauksessa tietokantaan talletetaan eri käyttäjien hallitsemat *Tracker*-profiilit, jotka sisältävät perustiedot fyysisistä paikannuslaitteista, sekä niiden lähettämistä paikannustiedoista, kuten koordinaateista. PostgreSQL-tietokanta siis toimii eräänlaisena välikätenä paikannustietojen tallettamiseen, sekä niiden turval-

liseen käyttämiseen. Tietoja voidaan suojata erilaisilla salauksilla, jotta vain oikeat käyttäjät pääsevät käsiksi omiin tietoihinsa. [48].

Palvelinympäristön rajapintoihin kuuluu *Tracker-API*, *Päätelaite-API*, sekä *WWW-käyttöliittymä*, joiden tarkoituksena on mahdollistaa RuuviTracker-järjestelmän käyttäminen useilla erilaisilla laitteilla tai järjestelmillä. [41].

Tracker-API on RuuviTracker-järjestelmän rajapinta, joka mahdollistaa fyysisen paikannuslaitteen tietojen lähettämisen GPRS-verkon kautta palvelimelle. Paikannustiedot pakataan mikrokontrollerin toimesta JSON-tiedostomuotoon ja lähetetään GPRS-yhteyden avulla palvelimen *Tracker-API*-porttiin, joka on ohjelmoitu Clojure-kielellä ottamaan vastaan viestejä paikannuslaitteelta. RuuviTracker-palvelin muuntaa JSON-muodossa olevan tiedon PostgreSQL-tietokannan mukaiseksi ja tallettaa sen käyttäjän profiilin tietoihin. [41].

Kun palvelimelta halutaan noutaa paikannuslaitteiden lähettämiä tietoja, tarvitaan *päätelaite-API*-rajapintaa. Kyseinen rajapinta osaa ottaa vastaan erilaisten päätelaitteiden lähettämiä käskyjä, joita palvelin sitten suorittaa. Käsky voi sisältää mm. tunnistustiedot, sekä suoritettavan käskyn tiedot. Tunnistustietojen avulla selvitetään, että kyseessä on oikea käyttäjä, jolle tietoja voidaan lähettää. Suoritettava käsky voi esim. käskeä palvelinta lähettämään tietyn RuuviTracker- paikannuslaitteen sijaintitiedot, jotka sittemmin muutetaan JSON-tiedostomuotoon ja lähetetään eteenpäin käyttäjälle. [41].

WWW-käyttöliittymä on palvelimelle sisällytetty verkkosivu, joka on suorassa yhteydessä palvelimen tietojen kanssa. Tämä mm. mahdollistaa eri käyttäjien rekisteröitymisen RuuviTracker-järjestelmään, jolloin saadaan muodostettua profiili tietokantaa varten. Tällöin myös oman RuuviTracker-paikannuslaitteen konfigurointi on mahdollista omien profiili sivujen kautta. Käyttöliittymä sisältää myös integroidun karttapalvelun, joka mahdollistaa oman paikannuslaitteen seuraamisen tavallisen internet-selaimen avulla. [41].

6.4 RuuviTracker-laitteen rakenne ja elektroniikka

RuuviTracker-laite koostuu monipuolisesta kokoelmasta erilaisia antureita ja toimielimiä, joista tärkeimpinä ovat ARM Cortex M4- merkkinen mikrokontrolleri, sekä SIM908- niminen GPS-, GSM-, GPRS- moduuli. Kyseiset komponentit mahdollistavat RuuviTracker- järjestelmän vaatimien perussovellusten toteuttamisen, mutta joukkoon kuuluu myös monia muita komponentteja, joita hyödyntämällä on mahdollista rakentaa vaativiakin toimintoja.

RuuviTracker- laitteen ohjelmointialusta toteutetaan eLUA- nimisellä pohjalla, joka soveltuu hyvin erilaisten sulautettujen järjestelmien suunnitteluun ja kehittämiseen. eLUA-pohja asennetaan RuuviTracker-laitteeseen ensiohjelmoinnin yhteydessä, jonka jälkeen laitteen ohjelmallisten muutosten tekeminen on helpompaa, kuin tavallisissa mikrokontrollereiden kehitysympäristöissä. eLUA-ympäristön avulla voidaan hallita kaikkia RuuviTracker-laitteen komponentteja skriptipohjaisilla komennoilla, joka tarkoittaa käytännössä sitä, että laitetta ei muutosten yhteydessä tarvitse aina ohjelmoida uudelleen. eLUA-ympäristöön suunniteltujen skriptien kirjoittaminen tapahtuu C-, tai Lua- ohjelmointikielillä. [51].

6.4.1 ARM Cortex M4- mikrokontrolleri

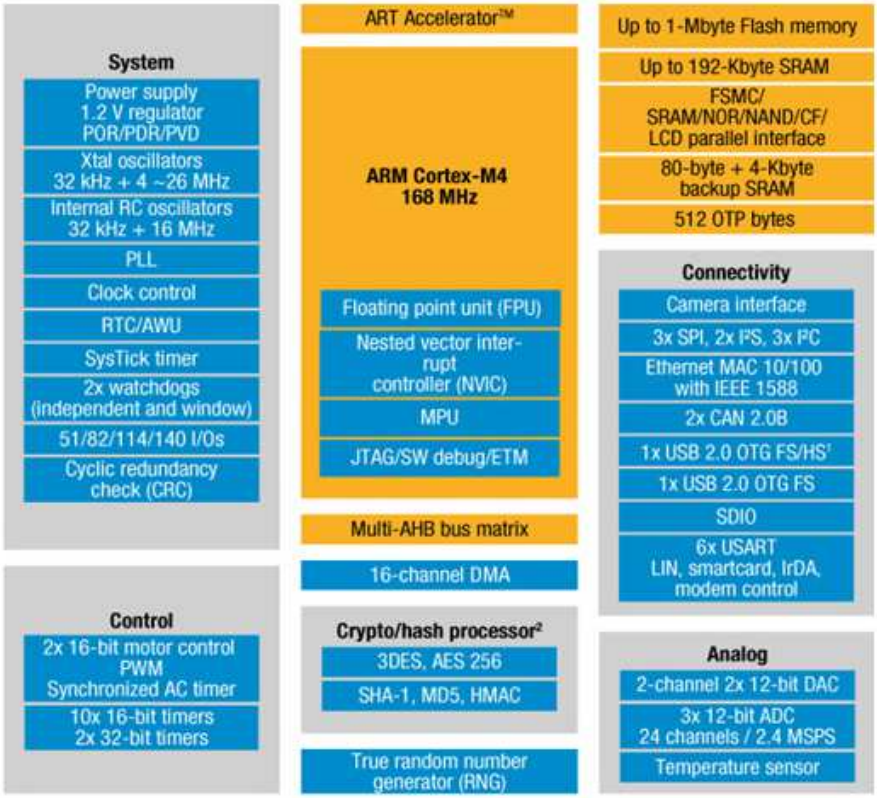
RuuviTracker- laitteen mikrokontrolleriksi on valittu STMicroelectronicsin valmistama ARM-mikroprosessoriarkkitehtuuriin kuuluva Cortex M4 piiri. ARM-mikrokontrollerit ovat hyvin suosittuja vaihtoehtoja sulautetuissa järjestelmissä. Nykyään piirejä käytetään lukuisissa matkapuhelimissa, kannettavissa minitietokoneissa ja muissa elektroniikka-laitteissa. ARM-piirit kuuluvat RISC-arkkitehtuuriin, jonka tarkoituksena on pitää konekielen käskyt mahdollisimman lyhyinä ja symmetrisinä. Tällä tavoin prosessorien tehokkuus pyritään pitämään mahdollisimman korkealla. [36].

ARM-mikrokontrollerit ovat pienikokoisia, vähävirtaisia, kustannustehokkaita, sekä niiden käyttömahdollisuudet ovat uskomattoman laajat. RuuviTracker RevB. levylle valittu kuvan 18 mukainen STM32F407 VGT6-piiri toimii jopa 168MHZ kellotaajuudella, sekä sisältää todella kattavan tuen lukuisille erilaisille käyttöliittymille. Pelkästään GPS-paikannukseen rakennettu RuuviTracker-laite toimisi pienempitehoisellakin

mikrokontrollerilla, mutta huomioon on otettu laitteen mahdollinen jatkokehitys, jotta laitteen teho, sekä liitännät olisivat varmasti riittävät laajempiin sovelluksiin. [36].

KUVA 18. ARM Cortex M4 STM32F407 VGT6- mikrokontrolleri

Kyseinen kuvan 18 mikrokontrolleri sisältää integroituna 1Mb ohjelmoitavaa FLASH-muistia, sekä 192kb RAM-muistia. Muistimäärät riittävät hyvinkin laajoihin sovelluksiin, mutta lisämuistia on mahdollista kytkeä mikrokontrollerin yhteyteen. Piiri sisältää myös A/D-, D/A- muuntimia, ajastimia, rinnakkaisporttiliitännän LCD-näytöille, käyttöliittymän kameraliitännälle, lukuisia ohjelmoitavia I/O-portteja, sekä paljon muita lisäominaisuuksia. Alla olevassa kuvassa 19 näkyy mikrokontrollerin tukemat ominaisuudet. [36].



KUVA 19. ARM Cortex M4- mikrokontrollerin toimintakaavio [36].

RuuviTracker-laitteen kannalta tärkeimpiin ominaisuuksiin kuuluu mm. I²C-käyttöliittymä, jonka avulla mikrokontrolleri on yhteydessä eri komponentteihin, kuten kiihtyvyysanturiin, gyroskooppiin, ym. Toinen tärkeä väyläliittymä on USART-väylä, jonka välityksellä SIM908-moduuli kommunikoi mikrokontrollerin kanssa. Piiri myös mahdollistaa kommunikoinnin tietokoneen kanssa USB-liitännän välityksellä, koska mikrokontrollerissa on integroitu USB- liitäntä mahdollisuus. Toisin sanoen piiri sisältää valmiita liitäntöjä ja ominaisuuksia, jotka mahdollistavat piirin liittämisen tietokoneen USB-väylään kommunikointia varten, ilman tarvetta ylimääräisille ohjelmointikaapeleille. [36].

6.4.2 Virransyöttö

RuuviTracker- laitteen virtalähteenä voidaan hyödyntää useampia erilaisia ratkaisuja. Laite kykenee hyödyntämään melko laajaa jännitealuetta 3-17V, joten virtalähteeksi käy myös auton akku 12V tai USB-liitin 5V. Ensisijaisesti laite on suunniteltu käyttämään tavallista pientä Li-ion, LiPo tai LiFePo4 akkua, joiden käyttöjännite on useimmiten 3,7V. Samanlaisia akkuja käytetään myös useimmissa matkapuhelimissa, joten RuuviTracker-laite sisältää myös latausmahdollisuuden kyseisille akuille. Tämä

edellyttää sitä, että RuuviTracker-laite on kytkettynä samanaikaisesti omaan akkuunsa ja USB-virtaliittimeen. [28].

RuuviTracker-laitteessa käytetään BQ24190-nimistä piiriä, jota käytetään erilaisissa virranhallintasovelluksissa. Piirin avulla pystytään mm. lataamaan akkua USB-liitännän välityksellä, sekä samanaikaisesti tarkkailemaan akun tilaa ohjelmallisesti. Piiriin on integroitu erilaisia antureita, joiden avulla pystyy seuraamaan akkujen tilaa ohjelmallisesti, sekä myös hallinnoimaan akun latausvaiheita. Piirin avulla on mahdollista estää erilaisia vaaratilanteita, kuten akkujen ylikuumenemisen. [28].

BQ24190-piiri kytkeytyy RuuviTrackerin mikrokontrolleriin piirilevyllä hyödyntäen I²C- väylää, jonka toiminnasta mainittiin aiemmin mikrokontrolleri- kappaleessa. Piiri soveltuu hyvin mikrokontrolleriympäristöön, koska tällöin pystytään ottamaan akun tila huomioon sovellusten suunnittelussa. On esim. mahdollista määrittää ohjelma siten, että jos akunvaraus tippuu alle 40 %, niin virransäästötila käynnistyy automaattisesti. [28].

RuuviTrackerin elektroniikkaan on sisällytetty myös TPS737xx LDO-regulaattori, jonka tarkoituksena on pitää eri komponenttien käyttämää jännitettä mahdollisimman vakaana ja eliminoida ”ei halutut” jännitepiikit. Kyseessä on lineaarinen jänniteregulaattori, joka tasaa jännitettä kuormituksen mukaan sopivaksi.

6.4.3 Haptiikka-ajuri

Haptiikka-ajuri on elektroninen piiri, jonka tarkoituksena on tuoda elektronisten laitteiden käyttöön ns. kosketustuntuma. Samaa voidaan verrata siihen, jos painetaan jonkun laitteen mekaanista kytkintä tai painonappia, niin voidaan tuntea kun nappi painuu pohjaan. Haptiikka-ajureiden tarkoituksena on luoda värinä moottoreiden eli aktuaattoreiden avulla samanlaista mekaanista tuntumaa esimerkiksi kosketusnäyttöjen käyttöön. Joissakin matkapuhelimissa käytetään pieniä värinämoottoreita, joiden avulla puhelimet saadaan esimerkiksi tärisemään puhelun saapuessa. Kyseisen piirin avulla voidaan ohjata näitä moottoreita omien tarkoitusten mukaisesti.

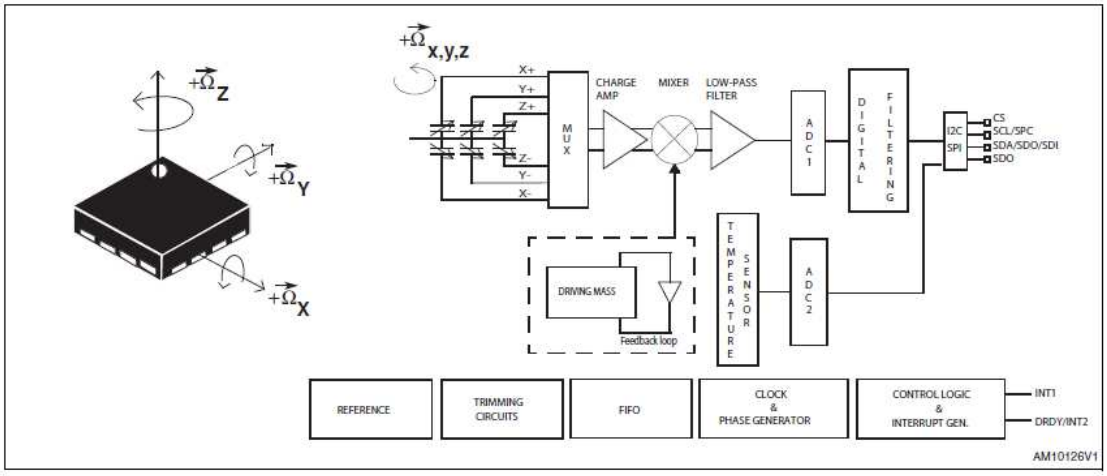
RuuviTracker-laitteessa on DRV2603-merkkinen haptiikka-ajuripiiri. Piiri kytketään suoraan ARM Cortex M4- mikrokontrollerin ohjainväyliin, joten piiriä voidaan ajaa

ohjelmallisesti. RuuviTracker-laitteen tapauksessa DRV2603-piirin avulla käytetään piirilevylle sisällytettyä lineaarivärähtelijää, jonka toiminta on samankaltaista värinämoottoreiden kanssa. [29].

6.4.4 Gyroskooppi

Gyroskooppi on laite, jonka avulla voidaan mitata kohteen asennon muutosta, sen oman keskuspisteensä suhteen. Mekaaninen gyroskooppi keksittiin 1800-luvulla ja sen avulla pystyttiin todistamaan maan pyöriminen. Nykyään gyroskoopin ominaisuuksia pystytään hyödyntämään myös lukuisissa elektronisissa laitteissa, kuten matkapuhelimissa, interaktiivisissa peleissä ja navigointilaitteissa.

RuuviTracker-laitteeseen kuuluu gyroskooppi, jota voidaan hyödyntää GPS-paikannus sovelluksissa tai muihin käyttötarkoituksiin. Etenkin kauko-ohjattavien lentokoneiden tai helikoptereiden täytyy jatkuvasti tietää oman asennon muutokset.



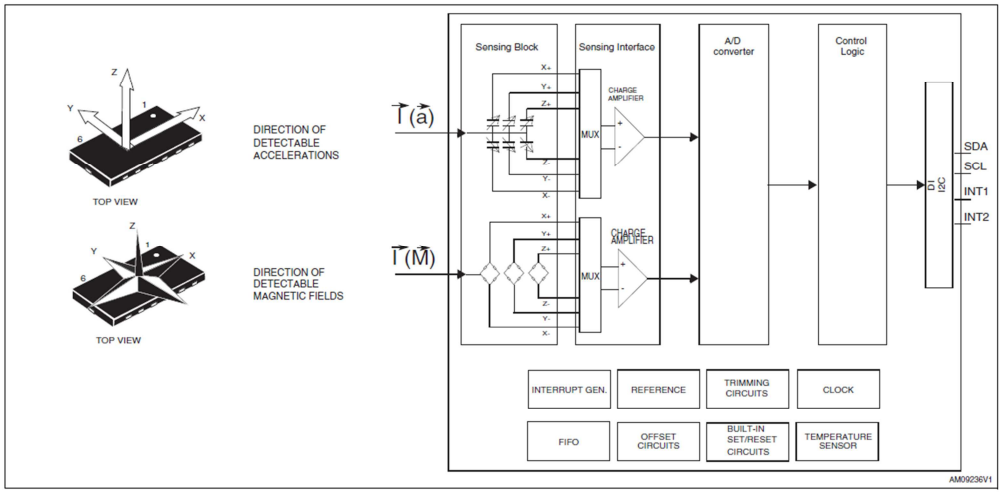
KUVA 20. L3GD20- gyroskoopin toimintamalli

RuuviTracker-laitteen L3GD20-gyroskoopin toiminta perustuu kuvan 20 esittämään toimintamalliin. Kuvanmukaiset geometriset vastusarvot X,Y ja Z muuttuvat aina piirin asennon mukaan. Piirin multiplekseri (MUX) yhdistää keräämänsä vastusarvot ja ne käännetään muun elektroniikan avulla helpommin tulkittavampaan muotoon. Piiriä voidaan tulkita ja hyödyntää mikrokontrollereiden avulla, esimerkiksi I²C-tiedonsiirtoväylän avulla, jota käytetään myös RuuviTracker- laitteessa gyroskoopin tulkitsemiseen. [30].

6.4.5 Kiihtyvyysanturi

Kiihtyvyysanturin avulla pystytään mittaamaan kappaleen kiihtyvyyttä, eli kappaleen nopeuden muutosta tietyssä ajassa. LSM303DLHC-piiri sisältää myös elektronisen kompassin, eli magnetometrin, jonka avulla pystytään mittaamaan magneettikentän voimakkuutta. Nykypäivänä kyseisiä antureita hyödynnetään navigaatiolaitteissa, roboteissa, matkapuhelimissa ja etenkin pelilaitteissa, kuten Nintendo Wii- pelikonsolissa. [31].

RuuviTrackerin tapauksessa molemmat anturit sopivat erittäin hyvin hyödynnettäväksi GPS-paikannuksen avuksi, sillä niiden avulla saadaan parannettua paikannustuloksia. Antureista on hyötyä myös reittien suunnittelussa, koska RuuviTracker tietää kompassin ja suunnan muutosten perusteella menosuuntansa. Antureita pystyy hyödyntämään tietenkin myös itse suunnittelemissaan sovelluksissa.



KUVA 21. LSM303DLHC – Elektronisen kompassin ja kiihtyvyysanturin toimintamalli

Piiri mittaa kuvan 21 mukaisesti kiihtyvyysanturin akseleita X,Y, Z, ja tulkitsee niihin vaikuttavia voimia eri arvoina I (a). Magnetometri mittaa samalla tavalla X,Y,Z akselien reagointia ympärillä olevaan magneettikenttään. Toimintaperiaate on aivan sama, kuin yksinkertaisella kompassilla. Molempien antureiden arvot yhdistetään multiplekseröillä, vahvistetaan ja muutetaan analogisesta muodosta digitaaliseen muotoon. Näin arvoja saadaan hyödynnettyä esimerkiksi mikroprosessorille luettavaksi ja ohjattavaksi. Tiedonsiirtoväylänä LSM303DLHC-piirin tapauksessa käytetään myös I²C-väylää. [31].

6.4.6 Muut komponentit

RuuviTracker-laitteen elektroniikkaan kuuluu myös paljon muita osa-alueita, kuten korttipaikat, lisämuisti, USB-liitin, laajennusportit, sekä erilaiset äänisignaalien käsittelyyn tarvittavat komponentit. Komponenttien yhteyteen on liitetty erilaisia ESD-suojauspiirejä, jotka turvaavat laitteen komponentteja vahingollisilta staattisilta sähköiskuilta. [32].

Korttipaikka muodostuu yhdestä kehikosta, johon voidaan liittää microSD-muistikortti, sekä tavallinen matkapuhelimissakin käytettävä SIM-operaattorikortti. Ilman SIM-korttia, ei GSM- ja GPRS-verkkojen käyttäminen olisi mahdollista. Muistikorttia voidaan käyttää laitteen massamuistina esim. omien sovellusten tukena, tai GPS-paikannuksen aikana paikannustietojen tallennuspaikkana. Hyvänä esimerkkinä mainittakoon tilanne, jossa GPRS- verkkoyhteys katkeaa tuntemattomasta syystä. Silloin GPS-koordinaatit voidaan tallettaa väliaikaisesti muistikortille ja siirtää palvelimelle jälleen, kun GPRS-yhteys palautuu. Kyseiseen operaatioon on myös mahdollista hyödyntää laitteen EEPROM- muistia, joka soveltuu ns. toistuviin kirjoitus- tai lukutilanteisiin paremmin, koska EEPROM- muisti kestää enemmän uudelleenkirjoitus prosesseja, kuin tavallinen Flash-muisti. [32].

RuuviTracker-laitteen mikrokontrollerilla on oma sisäinen ROM-muisti, mutta piirilevylle on lisätty ylimääräinen ROM-muistipiiri suurempikokoisten sovellusten varalle. Kyseessä on M24M01-merkkinen EEPROM-muistipiiri, joka on kooltaan 128 kilotavua. EEPROM- muisti on haihtumatonta puolijohdemuistia, jota on mahdollista uudelleenkirjoittaa kymmeniätuhansia kertoja. Muistipiiri liitetään RuuviTracker- laitteen mikrokontrollerin I²C-väylään, joka tekee muistin hallittavuudesta suhteellisen helppoa. [32].

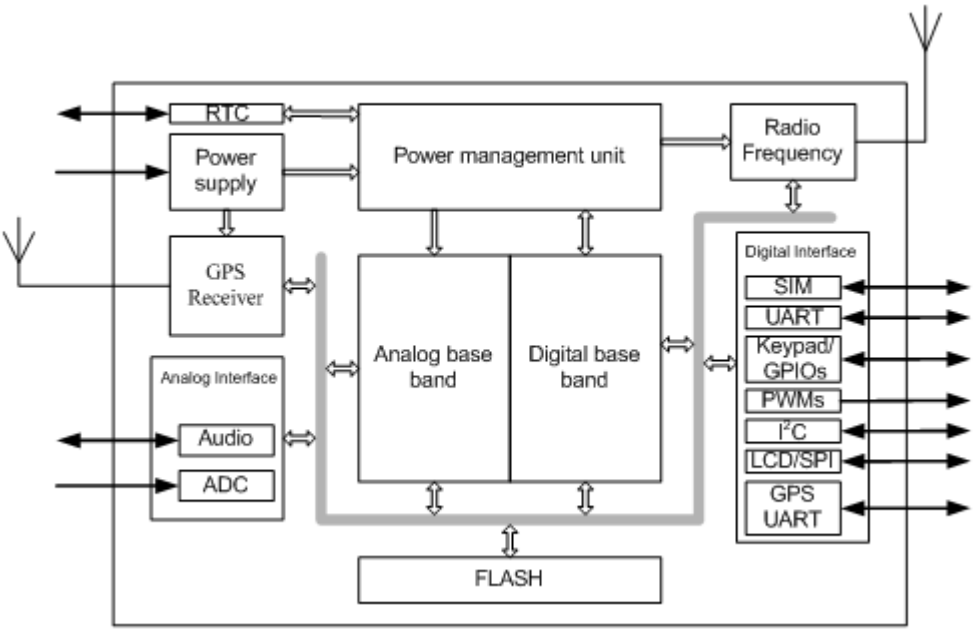
Äänisignaalien käsittelyä varten RuuviTracker-laitteesta löytyy mm. ADMP404-merkkinen MEMS-mikrofoni, joka on kytketty SIM908-moduulin mikrofoni vahvistimen liitäntään. Tällä tavoin piirilevylle ei tarvitse asentaa ylimääräisiä vahvistimia ja mikrofonia voidaan ohjailla SIM908-moduulin kautta AT-komennoilla. Mikrofoni kykenee valmistajan mukaan tulkitsemaan äänisignaaleita 100Hz-15kHz taajuusalueilla. [32].

Kaiutinvahvistuksena toimii LM4890M-merkkinen vahvistinpiiri, joka on teholtaan 1W ja soveltuu pääasiassa matkapuhelimiin ja muihin pienelektroniikkalaitteisiin. Kyseinen piiri on kytketty piirilevylle vastaanottamaan GSM-puheluiden äänisignaalia suoraan SIM908-moduulilta ja se soveltuu ensisijaisesti vain puheluiden vahvistamiseen. [32].

RuuviTracker-laitteessa on yleinen microUSB-AB-liitäntä, jonka avulla laite voidaan yhdistää tietokoneeseen esim. ohjelmointia, tai sisäisen akun latausta varten. Kyseinen liitinmalli, on yhteensopiva A- ja B- mallin microUSB kaapeleiden kanssa, joka käytännössä tarkoittaa laajempaa yhteensopivuutta eri laitteille. RuuviTracker-laitteen USB- liitäntä tukee myös ns. USB-OTG- toimintoa, joka mahdollistaa laitteen käytön isäntä-, tai orjalaitteena. Tavallisen USB- liitännän omaava erillislaite kykenee toimimaan ainoastaan orjana, ja tietokoneiden USB- liitännät isäntinä. Laitteeseen kuuluu myös muita liitäntöjä, kuten laajennusportit, joiden avulla mikrokontrollerin yhteyteen voidaan liittää lisää elektroniikkaa. Myös JTAG-liitäntä on sijoitettu piirilevylle, jotta laitteen sisäisten tai ohjelmallisten virheiden paikantaminen olisi mahdollisimman helppoa. [32].

6.5 SIM908- moduuli

SIM908 on SIMComin kehittämä kattava GSM/GPRS-moduuli, jossa on sisäänrakennettuna myös tuki GPS-teknologialle. Moduuli on kooltaan 30x30x3,2 mm ja se sisältää valmiin tuen GSM-, GPRS-, ja GPS-ominaisuuksien hyödyntämiseen erillisissä sovelluksissa. Moduulin elektroniikassa on myös integroitu tuki akun lataukselle ja äänisignaalin vahvistukselle, näin ollen se sopii erinomaisesti hyödynnettäväksi akullisiin käyttötarkoituksiin. Moduulin sisäinen latauspiiri on kuitenkin toiminnoiltaan suppeampi, joten RuuviTracker- laitteen elektroniikkaan onkin lisätty erillinen aikaisemmin mainittu BQ24190-latauspiiri. Alla olevassa toimintakaavio kuvassa 22 on nähtävillä tärkeimmät moduulin tukemat käyttöliittymät, sekä toiminnot. [33].



KUVA 22. SIM908-moduulin toimintakaavio [49, s. 8-12]

Moduuli sopii erittäin hyvin sovellettavaksi RuuviTracker-laitteen elektroniikan yhteydessä, sillä yksi paketti sisältää tuen kaikille tarvittaville yhteystyypeille. GSM- ja GPRS-verkon ominaisuuksilla mahdollistetaan RuuviTrackerin datasiirto GPRS- verkon kautta internet palveluihin ja GSM-verkko takaa laitteelle mahdollisuuden soittaa, sekä vastaanottaa puheluita ja konfiguroida RuuviTracker-laitteen ominaisuuksia tekstiviestien avulla. GPS-ominaisuudet ovat hyödynnettävissä samoilla käskyillä, kuin GSM- ja GPRS-verkon toiminnotkin, joka helpottaa laitteen käytettävyyttä. Moduuli ei sisällä valmiita antenneja vaan se tarvitsee toimiakseen erillisen GSM- ja GPS-antennin. Moduulissa on kaksi antenniliitäntää valmiina, joihin käytettävät antennit voidaan kytkeä suoraan. [33].

GPS- ominaisuuksiltaan moduuli on 42-kanavainen, joka tarkoittaa sitä, että moduuli kykenee vastaanottamaan GPS-signaalia samanaikaisesti 42:n kanavan kautta. Laite on suunniteltu vastaanottamaan satelliittien lähettämää C/A-koodia. Moduulin valmistaja lupaa alle 2,5m paikannustarkkuutta. Raaka GPS-data ilmoitetaan NMEA-muodossa, joka on yleisin GPS-laitteiden hyödyntämä tiedonsiirtomuoto. NMEA-data ilmoittaa tietyn ajanjakson välein kaiken keräämänsä GPS-datan, joihin sisältyy mm. laitekohtaista informaatiota ja muita paikannukseen liittyviä tietoja, kuten tärkeimpänä koordinaatit. RuuviTrackerin tapauksessa NMEA-data siirretään USART-sarjaliikenneportin kautta mikrokontrollerille, ellei muuten ole määritelty. [49, s. 8-12]

Moduuli tukee täysimittaisesti kaikkia GSM-verkkoja, joka tarkoittaa tukea 850/900/1800/1900 MHz taajuuksilla. Jotta moduulia on mahdollista käyttää GSM-verkossa, tarvitsee se käytettävän operaattorin tunnistetiedot. Tätä varten laitteeseen täytyy olla kytkettynä SIM-kortti, kuten GSM- kappaleessa mainittiin. Lähetystehoiltaan moduuli on verrattavissa tavallisten matkapuhelinten lähetystehoihin, jotka ovat 2W 850/900 MHz ja 1W 1800/1900 MHz. Moduulilla on siis mahdollista soittaa ja vastaanottaa tavallisia GSM-verkon puheluita esimerkiksi toisiin matkapuhelimiin. Myös tekstiviestien lähettäminen ja vastaanottaminen on mahdollista. [49, s. 8-12]

GPRS-ominaisuuksiltaan moduuli sijoittuu luokkaan B, joka on yleisin luokka myös matkapuhelimissa ja mokkuloissa. Käytännössä tämä tarkoittaa sitä, että moduuli voi olla GSM- ja GPRS verkkoihin yhteydessä samanaikaisesti, mutta vain toista palvelua on mahdollista käyttää kerrallaan. Yhteydet eivät kuitenkaan katkea vaan palveluita on mahdollista kytkeä väliaikaisesti pitoon. Esimerkiksi jos moduuliin saapuu GSM-puhelu samanaikaisesti kun GPRS- verkko on toiminnassa, niin GPRS- palvelu menee pitoon ja jatkuu automaattisesti, kun GSM- puhelu on lopetettu. Valmistaja lupaa moduulille maksimissaan 85,6 kbps tiedonsiirtonopeuden GPRS-verkon avulla. Moduulissa on myös integroitu tuki TCP/IP-protokollalle, joka parantaa kommunikointia erilaisissa internet palveluissa, sekä helpottaa laitteen datasiirto-ominaisuuksia etenkin ohjelmoitavuuden kannalta. [49, s. 8-12]

Moduulin ohjaus tapahtuu sisäänrakennettujen käyttöliittymien avulla. RuuviTracker-laitteen tapauksessa käytetään tavallista UART-sarjaporttiliitäntää, jonka välityksellä moduuli voi kommunikoida mikrokontrollerin kanssa. [34 & 50, s. 123.]

Moduulin varsinainen käskytys tapahtuu AT-komennoilla, jotka kehitettiin alunperin vanhojen modeemilaitteiden komennoiksi. AT-komentoja käytetään vielä nykyäänkin hyvin monissa laitteissa, koska ne ovat laajasti tuettuja ja melko yksinkertaisia. AT-käskyt muodostavat eräänlaisen komentokielen, joka rakennetaan erilaisista lyhenteistä ja arvoista, joita yhdistelemällä saadaan toteutettua monimutkaisia toimintaoperaatioita. Käskytystapahtuma muodostuu päätelaitteelle lähetettävästä komennosta ja päätelaitteen takaisin lähettämästä vastauksesta. Päätelaitteen vastaus on useimmiten ilmoitus ”*OK*” jos käskyn suorittaminen onnistuu halutulla tavalla. Jos käskyn suorittaminen epäonnistuu, saadaan vastauksena virheilmoitus, kuten ”*ERROR*”. Yksinker-

taisena esimerkkinä voidaan tarkastella taulukon 3 AT-komentoa ja sen vastausta, jonka avulla tarkistetaan komentoyhteyden tila. [34 & 50, s. 123.]

TAULUKKO 3. Yhteyden tilan tarkistaminen AT-komennolla [34 & 50, s. 123.]

Päätelaitteelle lähetettävä käsky:	AT
Päätelaitteen takaisin lähettämä vastaus:	OK

AT-komennot vaihtelevat yleisesti hieman riippuen aina laitteesta, tästä johtuen kaikkiin AT-komennoilla ohjattaviin laitteisiin tulisi olla oma AT-komentokirjasto. Komentokirjaston avulla saadaan selville eri laitteiden tukemat käskyt ja käskyjen muodostamisperiaatteet. Seuraavan taulukon 4 käsky perustuu SIM908 moduulin käskykirjastoon ja sen avulla saadaan selville moduulin valmistaja, nimi ja ohjelmaversio. [34 & 50, s. 123.]

TAULUKKO 4. Pyydetään moduulin laitetiedot AT-komennolla [50, s. 123.]

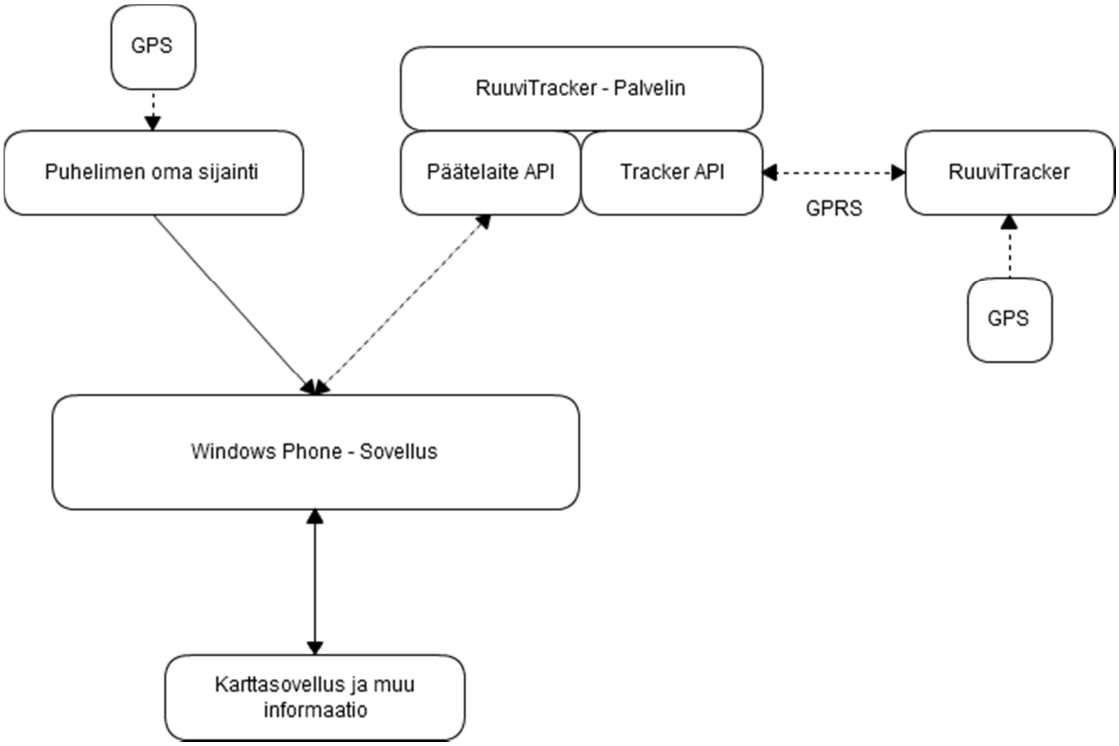
Päätelaitteelle lähetettävä käsky:	AT+GSV
Päätelaitteen takaisin lähettämä vastaus:	SIMCOM_Ltd SIMCOM_SIM908 Revision:1137B02SIM908M64_ST

Kyseiset AT-komennot antavat hyvin perustietoa siitä, miten mikrokontrolleri kommunikoi SIM908-moduulin kanssa. Mikrokontrolleri muodostaa ohjelmoitujen käskyjen mukaisesti AT-komentoja, joita sitten lähetetään USART- käyttöliittymän välityksellä moduulille. Moduuli toimii AT-komentojen mukaisesti ja halutut vastaukset, kuten GPS-sijaintidata, lähetetään saman USART-linjan kautta takaisin mikrokontrollerille. Järjestelmä toimii siis sulautetussa ympäristössä, joka tarkoittaa sitä, että käyttäjällä ei tarvitse olla perustietoa AT-komentojen muodostamisesta, lukuun ottamatta mikrokontrollerin firmwaren ohjelmointityötä. [34 & 50, s. 123.]

7 OHJELMOINTITYÖ

Ohjelmointityön tarkoituksena on rakentaa Windows Phone- käyttöjärjestelmälle sovellus, jolla on mahdollista noutaa yksinkertaisia tietoja RuuviTracker- palvelimelta. Työssä käytetään Microsoftin tarjoamaa Visual Studio- kehitysympäristöä ja testaus suoritetaan Windows Phone- emulaattorin ja sen lisätyökalujen avulla. Ohjelmoitavan

sovelluksen tarkoituksena on demonstroida tärkeimpiä ohjelmointimetodeja yksinkertaisella tavalla.



KUVA 23. Windows Phone- sovelluksen toimintaperiaate

Ohjelmointityössä tarvitaan visuaalista karttakäyttöliittymää, jonka tarkoituksena on näyttää käytettävän Windows Phone- puhelimen, sekä paikannettavan RuuviTracker-laitteen sijainti karttanäkymässä, kuten kuvassa 23 esitetään. Tähän tarkoitukseen olen valinnut Windows Phone- kehitysympäristön oman karttatyökalun, joka hyödyntää Microsoftin omistamaa Bing-karttapalvelua. Visual Studio- kehitysympäristö sisältää valmiin karttatyökalun, jolla voidaan lisätä karttaelementti suoraan ohjelmointinäkymään, aivan kuten tekstikentän tai painonapin. Karttaelementti sisältää valmiina kaikki tavanomaisimmat ominaisuudet, kuten selailu-, paikannus- ja zoom- efektit. Karttaelementtien käyttö vaatii Bing- karttapalvelun tunnukset, joista tarkemmin seuraavassa kappaleessa. [27, s. 156-158].

Toinen tärkeä ominaisuus tämän sovelluksen kehityksen kannalta on oman Windows Phone- laitteen sijainnin määrittäminen, johon käytän *GeoCoordinateWatcher*- luokkaa. Kyseinen luokka on suunniteltu ensisijaisesti juurikin oman sijainnin määrittämistä varten, joten sillä onkin helppoa hakea oman sijainnin koordinaattitiedot omia sovellutuksia varten. [27, s. 152-156].

Tärkeimpänä ominaisuutena tämän sovelluksen laatimisen kannalta pidän metodia, jolla noudetaan tietoa RuuviTracker-palvelimelta. Kyseessä on metodi, jolla muodostetaan URL-muotoinen käsky lähetettäväksi RuuviTracker-palvelimelle. Samaan metodiin sisältyy myös palvelimen vastauksena lähettämän JSON-muotoisen tiedon vastaanotto ja parsinta oman sovelluksen käytettäväksi.

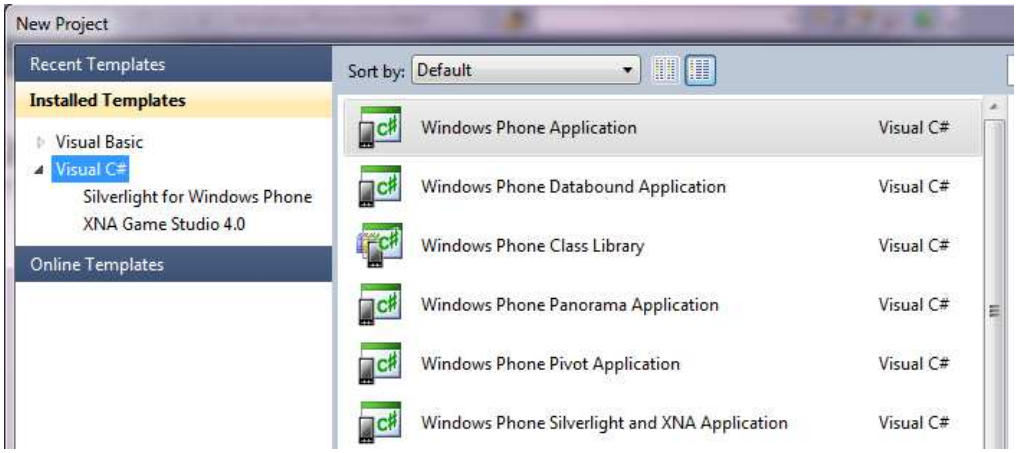
7.1 Sovelluskehitystyökalujen käyttöönotto

Aivan aluksi tarvitsee ladata ja asentaa Windows Phone- kehitystyökalut tietokoneelle. Työkalut löytyvät Microsoftin omilta kotisivuilta Download Center- osiosta. Kyseessä on valmis asennuspaketti, joka sisältää kaikki tarvittavat työkalut ja Windows-käyttöjärjestelmän vaatimat päivitykset. Asennusprosessi on hyvin yksinkertainen, eikä käyttäjän tarvitse huolehtia juurikaan mistään ylimääräisestä, sillä asennusohjelma tarkistaa ja päivittää kaikki tarvittavat sovellukset. Kuvassa 24 esitetään Microsoftin Download Center- näkymää, josta kehitystyökalut voidaan ladata.

KUVA 24. Windows Phone- kehityspaketti Microsoftin Download Centerissä

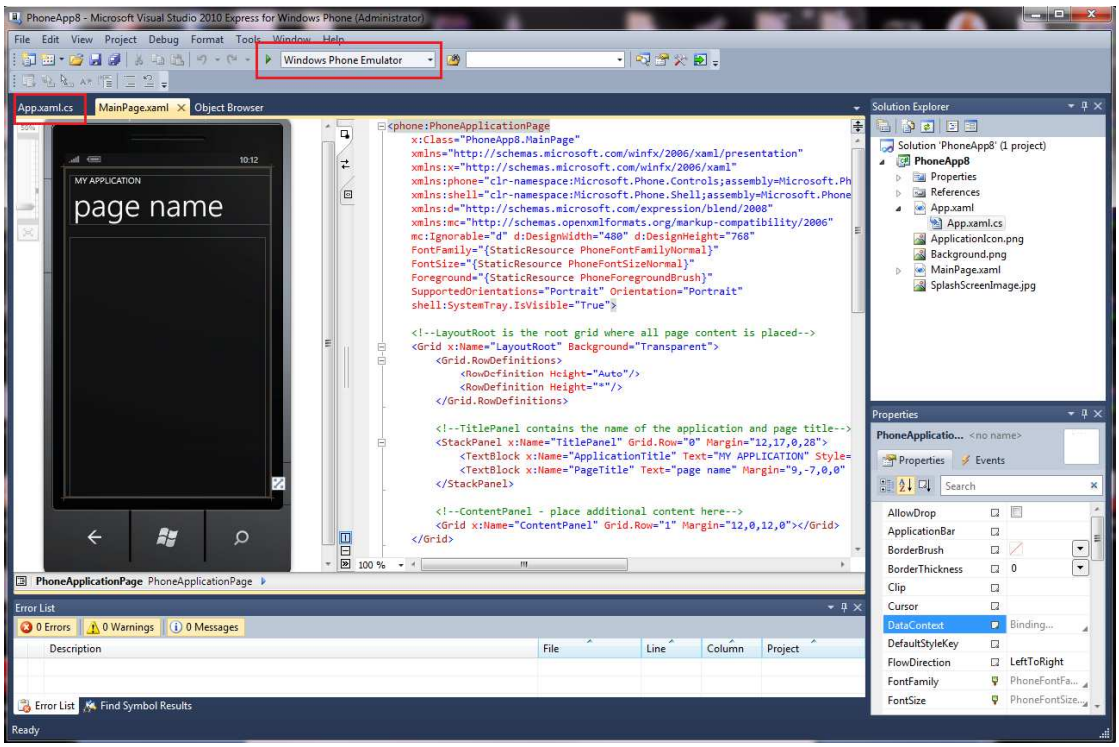
Kun asennusprosessi on saatu valmiiksi, voidaan avata Visual Studio 2010- sovellus, jolla suoritetaan kaikki tätä ohjelmointityötä varten tarvittava. Asennuspaketti asentaa automaattisesti myös muut ohjelmointityökalut, kuten XNA- Game Studio- ja Expression Blend- sovellukset, mutta niitä ei tässä tapauksessa tarvita.

Ensimmäisellä käyttökerralla täytyy valita uuden projektin aloitus, johon valitaan tietenkin *Windows Phone Application*. Samassa näkymässä täytyy myös valita projektissa käytettävä ohjelmointikieli, joita on vaihtoehtoina C# ja Visual Basic. Tässä tapauksessa käytetään hieman yleisempää C#- kieltä, joka on myös minulle tutumpaa. Kuva 25 esittää näkymää uuden projektin aloittamisesta ja ohjelmointikielen valinnasta.



KUVA 25. Visual Studio 2010- Uusi projekti

Kun uusi projekti on saatu aloitettua, päädytään Visual Studio 2010 Express- version tavanomaiseen aloitusruutuun, joka koostuu kuvan 26 näkymästä. Vasemmassa laidassa sijaitsee Windows Phone- näkymä, jota voidaan käyttää ohjelman visuaalisen suunnittelun apuvälineenä. Näkymä päivittyy sitä mukaa, kun ohjelman XAML-koodia muutetaan tai näkymään lisätään työkalujen avulla valmiita elementtejä, kuten tekstikenttiä tai painonappeja. Vaihtoehtoisesti elementit voidaan lisätä tai muokata keskellä sijaitsevan XAML-editorin avulla kirjoittamalla. Oikeassa alanurkassa sijaitsee XAML-koodin kirjoittamista varten helpottavia työkaluja, joiden avulla voidaan määritellä erilaisille objekteille ominaisuuksia tai määritteitä, ilman että XAML-koodiin tarvitsee kajota.

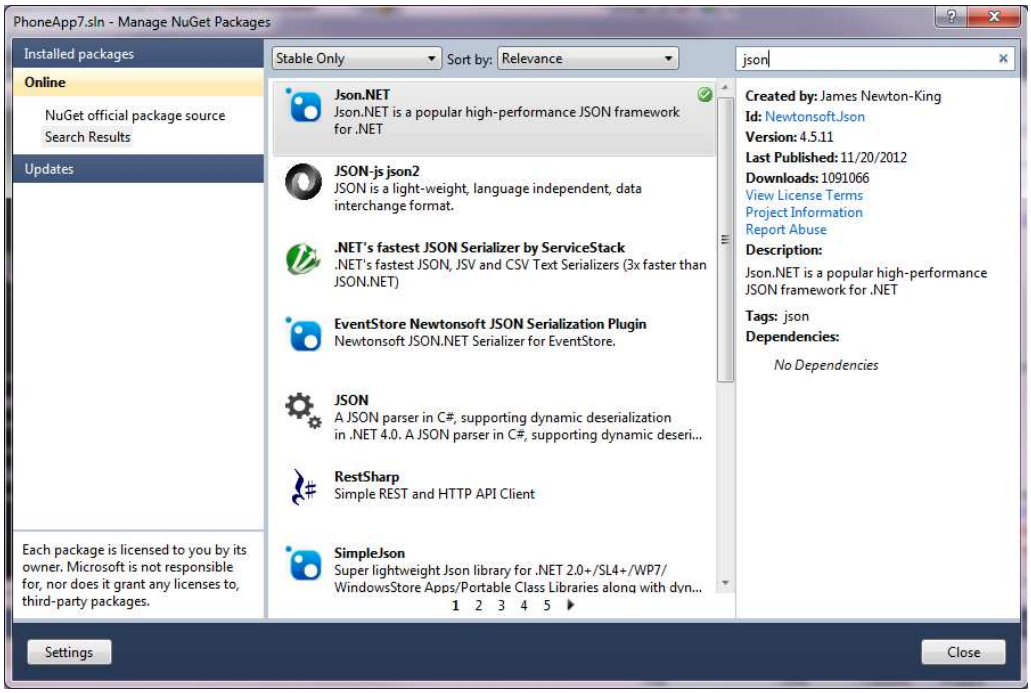


KUVA 25. Visual Studio 2010- näkymä

Oikeassa yläreunassa sijaitsevalla *Solution Explorer*- ikkunan välityksellä nähdään kaikki projektin sisältämät tiedostot, jotka sijaitsevat myös fyysisesti tietokoneen kiintolevyllä omassa kansiossaan. Kyseisen ikkunan avulla voidaan myös avata projektin kooditiedosto, johon kirjoitetaan kaikki sovelluksen C#- muotoinen ohjelmakoodi. Kuvan ylärivillä sijaitsee punaisella ympyröity palkki, josta voidaan halutessa käynnistää oman sovelluksen testaus Windows Phone- emulaattorin avulla. Emulaattori voidaan käynnistää vain, kun ohjelmakoodi on kirjoitettu virheettömäksi, joka voidaan todeta alareunan virheenpaikantimen avulla. Paikannin ilmoittaa aina, kun sovelluksessa ilmenee virheellistä koodia. Paikannin myös ilmoittaa virheiden sijainnin, sekä ehdottaa erilaisia ratkaisumahdollisuuksia.

Lopuksi asennetaan Visual Studioon käytettäväksi kolmannen osapuolen tekemä kirjasto JSON- tiedonhallintaa varten. Kyseinen kirjasto on toiminnoiltaan tehokkaampi, kuin Visual Studion sisäiset käskykannat. Uusien kirjastojen lisäämistä Visual Studioon voidaan kontrolloida *NuGet*- paketinhallintatyökalun avulla, joka voidaan asentaa Visual Studion yhteyteen erikseen esim. sen omilta kotisivuilta http://nuget.codeplex.com/. Kyseinen työkalu pitäisi löytyä Visual Studion uudemmista versioista valmiiksi asennettuna. Työkalun asennusvaihe on yksinkertainen, eikä se vaadi käyttäjiltä juurikaan muuta, kuin seuraava- napin painamista.

Kun työkalu on asennettu, voidaan se ottaa käyttöön esim. klikkaamalla Visual Studion aloitusruudun *Solution Explorer*- ikkunassa sijaitsevaa omaa projektia hiiren oikealla painikkeella. Tällöin ruudulle ilmestyy teksti ”*Manage NuGet packages for solution*”, jota painamalla saadaan lisättyä uusia kirjastoja suoraan oman sovelluksen käytettäväksi. *NuGet*- työkalun aloitusruutu näyttää kuvan 26 mukaiselta ja sen avulla voidaan etsiä haettavia kirjastoja työkalun omasta tietokannasta hakukentän avulla.



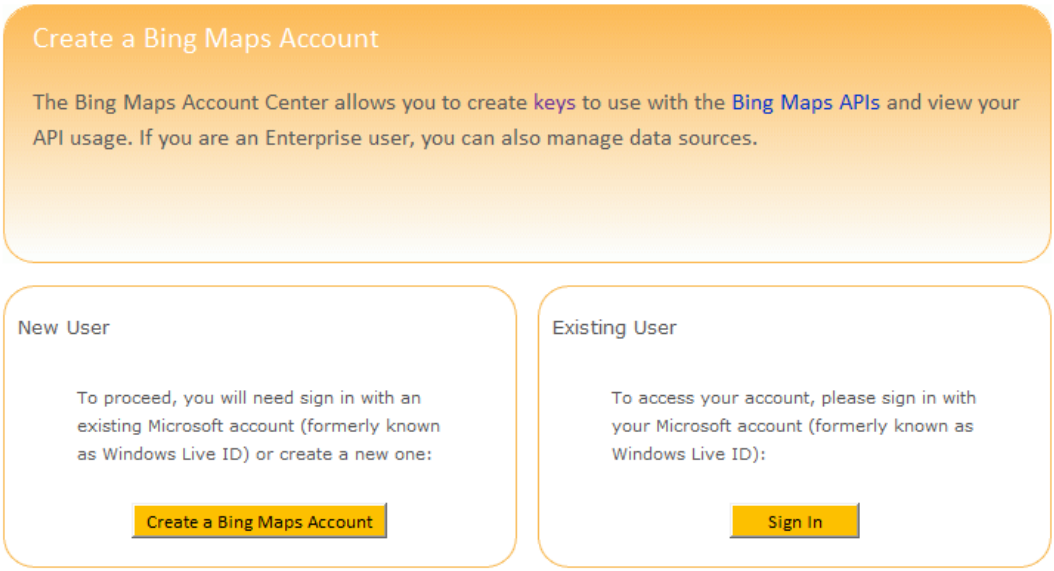
KUVA 26. NuGet- paketinhallinta työkalu

Kyseistä projektia varten etsitään hakukentän avulla *JSON.NET* nimistä kirjastoa, joka ilmestyy kuvan 26 keskellä sijaitsevaan näkymään. Kyseessä on James Newton-King-henkilön tekemä kirjasto, joka on kehittynyt apuväline JSON- tietojen hallintaa varten. Kirjaston asentaminen tapahtuu valitsemalla se *NuGet*- työkalun keskinäkymästä ja klikkaamalla *install*. Tällöin kirjasto asentuu suoraan oman projektin käytettäväksi, eikä muunlaisia valintoja enää tarvitse tehdä.

7.2 Bing- karttapalvelun käyttöönotto

Kuten kappaleen 7 alussa mainittiin, niin Visual Studio- työkalujen avulla voidaan lisätä sovelluksiin valmiita karttaelementtejä, joiden toimintoja on mahdollista hyödyntää omissa sovellutuksissa. Kyseiset elementit käyttävät hyväkseen Microsoftin Bing- karttapalvelua, jonka käyttämiseen tarvitaan käyttäjätunnukset.

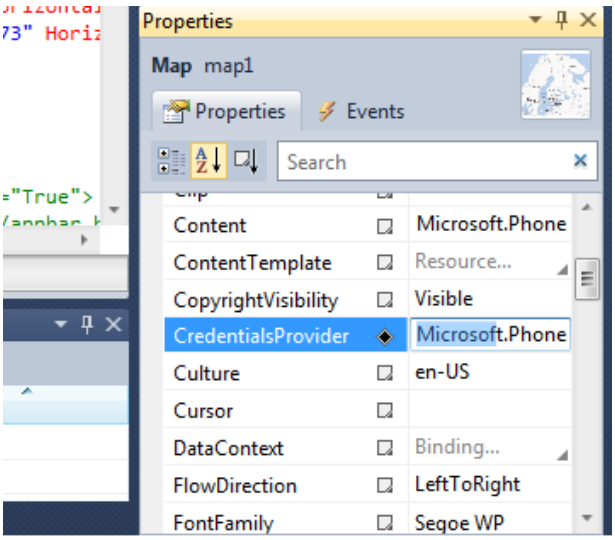
Bing-karttojen käyttöönotto vaatii ensimmäisenä rekisteröitymisen Bing-palvelun käyttäjäksi, sillä karttapalvelun käyttäminen omissa sovelluksissa edellyttää ns. kartta-avainta, joka kertoo Microsoftille kuka karttoja käyttää. Rekisteröityminen onnistuu http://www.bingmapsportal.com/ osoitteessa ja käyttäjätilin luominen vaatii Windows Live ID-tunnuksen.



KUVA 27. Rekisteröityminen Bing- karttapalveluun

Rekisteröityminen tapahtuu kuvan 27 näyttämän ruudun mukaisesti ja prosessin päätteeksi saadaan käytettäväksi henkilökohtainen kartta-avain, joka on pitkä jono kirjaimia ja numeroita. Avaimen tarkoituksena on tunnistaa karttapalvelun käyttäjät ja parantaa palvelun turvallisuutta.

Lopuksi avain täytyy ottaa käyttöön ohjelmoitavassa sovelluksessa, jotta tunnistusprosessi toimii aina, kun sovellus suorittaa karttaelementin ominaisuuksia. Käytännössä avaimen lisääminen sovellukseen voidaan suorittaa lisäämällä avain sovelluksen XAML- koodin yhteyteen kuvan 28 mukaisella tavalla. Kuvassa kartta-avain lisätään suoraan karttaelementtiin *Properties*- ikkunan kautta *CredentialsProvider*- kenttään. Kyseessä on valmis ominaisuus, joka suorittaa tunnistamisprosessin automaattisesti.



KUVA 28. Bing- palvelun kartta-avaimen lisääminen XAML- koodiin

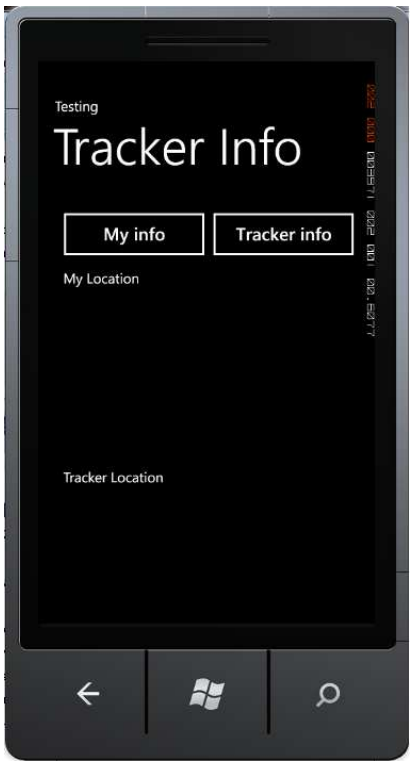
7.3 Sovelluksen visuaalinen näkymä

Ohjelmointityöni koostuu kolmesta erillisestä XAML- koodilla laaditusta sivusta, jotka ovat kytköksissä sovelluksen pääsivulle. Sovelluksen pääsivu sisältää kaksi painonappia, joiden avulla voidaan siirtyä toisille sivuille. Visuaalisesti ohjelman pääsivu näyttää kuvan 29 mukaiselta. Tarkempi kuvaus rakentamani ohjelman XAML- koodista löytyy opinnäytetyön ensimmäisestä liitetiedostosta. Kaikki visuaaliset elementit, kuten painonapit, tekstikentät ja karttakuva ovat lisätty Visual Studion omilla työkaluilla, joten varsinaista XAML- koodia, minun ei ole juurikaan tarvinnut kirjoittaa.

KUVA 29. Ohjelmointityön pääsivu

Pääsivu koostuu kaiken kaikkiaan neljästä eri painonapista, sekä yhdestä tekstikentästä. Kuvan 29 alemmat painonapit: *Info* ja *Map* ovat navigointi painikkeita, joiden avulla voidaan siirtyä sovelluksen kahdelle muulle sivulle. *Info*- sivu koostuu ominaisuuksista, joiden avulla voidaan tarkastella paikannustietoja tekstimuodossa ja *Map*-sivu koostuu karttanäkymästä, jonka avulla voidaan tarkastella paikannettavia kohteita visuaalisessa muodossa.

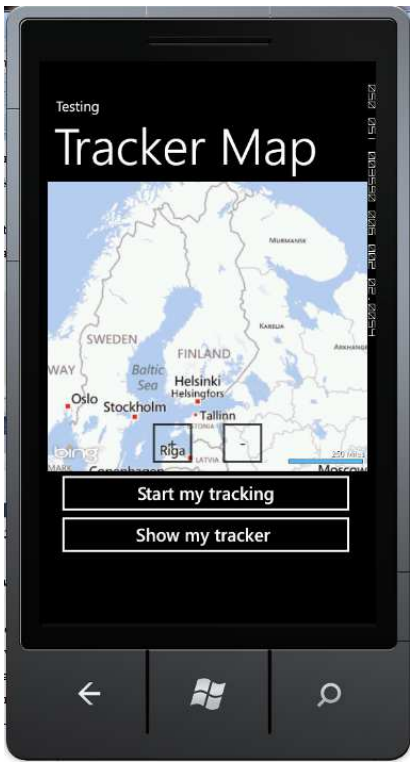
Pääsivun ylemmät painikkeet: *Server Status* ja *Find Tracker* ovat tarkoitettu RuuviTracker- palvelimen tilan tarkastamiseen. *Server Status*- painike suorittaa yksinkertaisen ping- kutsun palvelimelle ja ilmoittaa pääsivun tyhjään tekstikenttään palvelimen antaman vastauksen. Kyseinen tapahtuma voidaan toteuttaa mm. eri tilojen mukaisesti ilmoittamalla yksinkertaisesti, ”yhteys palvelimeen mahdollinen” tai ”palvelimeen ei saada yhteyttä”. *Find Tracker*- painikkeella on tarkoitus lähettää palvelimelle pyyntö, jossa määritetään oman RuuviTracker- paikannuslaitteen tiedot ja halutaan saada selville, onko laite aktiivisessa tilassa.



KUVA 30. Ohjelmointityön Info- sivu

Ohjelmointityöni *Info*- sivu koostuu kuvan 30 mukaisesti kahdesta painonapista, sekä kahdesta tekstikentästä. *Info*- sivun tarkoituksena on esittää paikannustiedot tekstimuodossa omasta sijainnista, sekä RuuviTracker- laitteen sijainnista. *My Info*- painonappi noutaa käytettävän Windows Phone- laitteen sijaintitiedot ja kirjoittaa ne ylempään *My Location*- tekstikenttään. *Tracker Info*- painikkeen toimintaperiaate on aivan sama, paitsi sijaintitiedot noudetaan RuuviTracker- palvelimelta käsin.

Sijaintitietojen tekstimuotoinen esitys koostuu laitteiden koordinaattitiedoista, joihin kuuluu: leveysaste, pituusaste, korkeus, nopeus ja kurssi. Periaatteessa kyseisten tietojen esittämisestä tekstimuotoisena ei ole suurta hyötyä, mutta tarkoituksenani onkin sovelluksen avulla demonstroida erilaisten tietojen seurantaa. Tällä tavoin on helpompaa saada parempi käsitys siitä, kuinka monimutkaisemmatkin operaatiot toimivat.



KUVA 31. Ohjelmointityön Map- sivu

Map- sivu, kuvassa 31 koostuu karttanäkymästä, joka hyödyntää aiemmin mainittua Bing- karttapalvelua. Kartta on lisätty sivun näkymään valmiina elementtinä ja kartta-avain on tallennettu suoraan elementin XAML- koodiin. Jo valmiina elementtinä kartta sisältää kaikki tavanomaisimmat interaktiiviset ominaisuudet, kuten esim. zoom-efektiä varten karttakuvassa näkyvät omat painikkeet. Tarkempaa kuvausta ohjelman *Map*- sivusta saa tämän opinnäytetyön 1. liitetiedostosta, jossa esim. koodin yhteyteen lisätty Bing- karttapalveluiden tunnus on yliviivattu.

Sivun kaksi painonappia sijaitsevat karttakuvan alapuolella ja niiden tarkoituksena on kohdistaa paikannettavat kohteet pisteiksi karttanäkymään. Ensimmäinen *Start my tracking*- painonappi aloittaa Windows Phone- laitteen paikannuspalvelun, joka lisää kartalle *pushpin*- nastan sijainnin merkiksi. *Show my tracker*- painikkeella on käytännössä sama toiminto, mutta sen avulla lisätään toinen nasta karttanäkymään, jotta on mahdollista hahmottaa kummankin laitteen samanaikainen sijainti kartalla.

Painonappien alapuolella on myös tekstikenttä, jonka tarkoituksena on esittää erilaisia tilannetietoja käyttäjälle. Esimerkiksi GPS- sijainnin määrityksen aikana esitetään teksti ”*Searching you...*”, eli etsitään sinua. Muulloin tekstikenttään voidaan ilmoittaa virheitä, kuten paikannussignaalin katkeaminen, verkkoyhteyden katkeaminen, ym.

7.4 Sovelluksen ohjelmakoodi

Olen laatinut ohjelmointityön suunnittelun kolmeen erilaiseen aiheeseen, joihin kuuluvat käytettävän Windows Phone- laitteen sijainnin määrittäminen, JSON- tiedostomuotoisen datan siirto ja parsinta RuuviTracker- palvelimelta, sekä RuuviTracker-laitteen paikantaminen. Kyseisten ominaisuuksien avulla on mahdollista rakentaa sovellukseen tarvittavat peruskomennot, joiden avulla mahdollistetaan tietojen siirtäminen RuuviTracker- palvelimelta langattomasti matkapuhelimeen, sekä voidaan hyödyntää Windows Phone- laitteen sijaintia sovelluksen toiminnoissa.

Kyseessä on kuitenkin melko yksinkertaiset toiminnot, joiden lisäksi tarvitaan paljon hienosäätöä ja enemmän testikäyttöä. Jokaisessa aiheessa selostetaan, kuinka eri toiminnot toimivat ja mitä niillä on tarkoitus saada aikaiseksi. Kaikista vaiheista on myös esillä C#- ohjelmakoodia, jolla on onnistuneesti testattu aiheeseen liittyviä toimintoja.

7.4.1 Oman sijainnin määrittäminen

Windows Phone- laitteen omaa sijaintia määritettäessä voidaan käyttää valmista *GeoCoordinateWatcher*- luokkaa, jonka ensisijaisena tarkoituksena on mahdollistaa laitteen käyttäjälle pääsy paikannustietoihin. Kyseisen luokan avulla voidaan määrittää aktiivinen palvelu, joka tarkkailee jatkuvasti laitteen omaa sijaintia ja päivittää sijaintitiedot sitä mukaan, kun ne muuttuvat.

```
using Microsoft.Phone.Controls.Maps;
using System.Device.Location;

namespace PhoneApp7
{
    public partial class MapPage : PhoneApplicationPage
    {

        GeoCoordinateWatcher watcher;

        public MapPage()
        {
            InitializeComponent();
        }


        private void button1_Click(object sender, RoutedEventArgs e)
        {
            if (watcher == null)
            {
                watcher = new GeoCoordinateWatcher(GeoPositionAccuracy.High);
                watcher.MovementThreshold = 20;
                watcher.StatusChanged += new EventHandler<GeoPositionStatusChangedEventArgs>(watcher_StatusChanged);
                watcher.PositionChanged += new EventHandler<GeoPositionChangedEventArgs<GeoCoordinate>>(watcher_PositionChanged);
            }

            watcher.Start();

        }
```

KUVA 32. GeoCoordinateWatcher- luokan määrittäminen

Kuvassa 32 esitetään ohjelmointityöni C#- koodia, jossa *GeoCoordinateWatcher*-luokka otetaan käyttöön painiketta painamalla. Aluksi ohjelmaan täytyy määrittää paikannukseen käytettävät resurssit using- lauseen avulla, jotta ne saadaan oman ohjelman käyttöön. Tarvittavat resurssit saadaan käytettäväksi lisäämällä koodin alkuun kuvassa 32 ylhäällä esiintyvät *Microsoft.Phone.Controls.Maps* ja *System.Device.Location* määritteet.

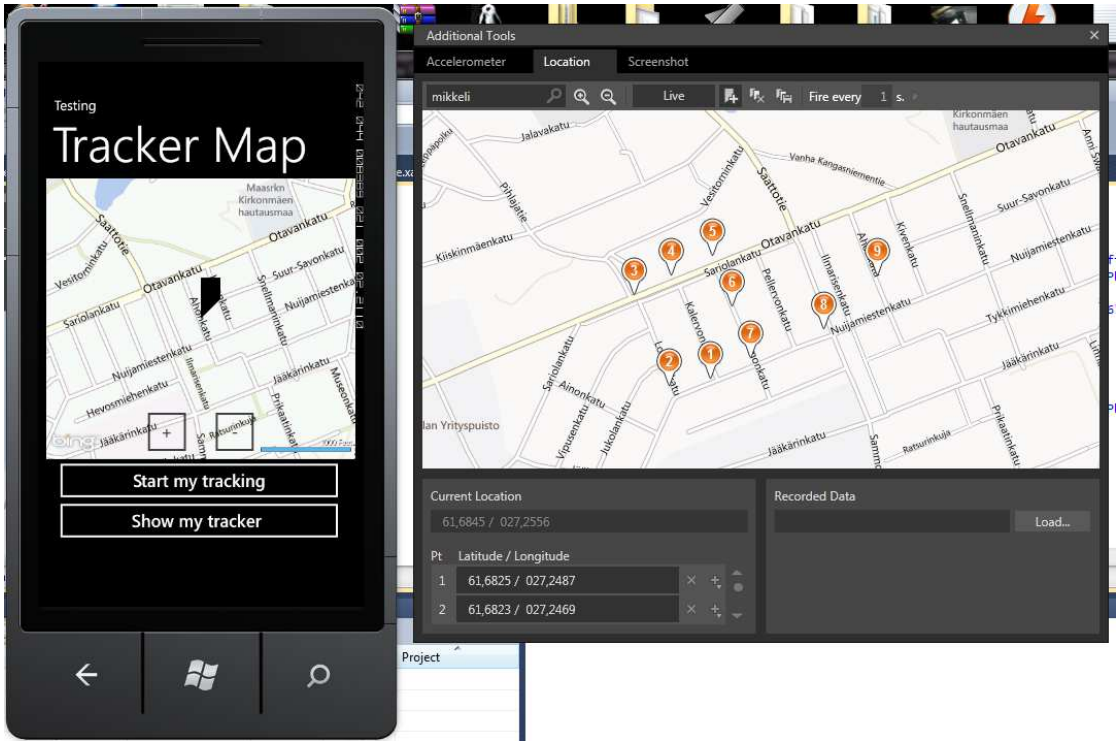
Kun painonappi aktivoidaan sovellus käynnistää *GeoCoordinateWatcher*- luokan samalla määrittäen sille mahdollisimman tarkan paikannustuloksen antamalla *GeoPositionAccuracy*- määreelle arvon *High*. *MovementThreshold*- määritteen avulla voidaan päättää kuinka usein uuden sijainnin tieto haetaan uudelleen. Kyseisessä sovelluksessa määritteelle on annettu arvo 20, joka ilmaisee metrejä. Tällöin ohjelma noutaa uudet sijaintitiedot aina 20 metrin liikkumisen jälkeen. Kyseisten määritteiden alapuolella sijaitsevat *StatusChanged*- ja *PositionChanged*- tapahtumat, joiden avulla voidaan lisätä ohjelmaan toimintoja eri tilojen mukaisesti.

```
void watcher_StatusChanged(object sender, GeoPositionStatusChangedEventArgs e)
{
    switch (e.Status)
    {
        case GeoPositionStatus.Disabled:
            MessageBox.Show("Location Service is not enabled on the device");
            break;

        case GeoPositionStatus.NoData:
            MessageBox.Show(" The Location Service is working, but it cannot get location data.");
            break;
    }
}

void watcher_PositionChanged(object sender, GeoPositionChangedEventArgs<GeoCoordinate> e)
{
    if (e.Position.Location.IsUnknown)
    {
        this.notification.Text = "Searching you...";
        return;
    }

    map1.Children.Clear();
    Pushpin pin = new Pushpin();
    pin.Location = new GeoCoordinate(e.Position.Location.Latitude, e.Position.Location.Longitude);
    map1.Children.Add(pin);

    map1.Center = new GeoCoordinate(e.Position.Location.Latitude, e.Position.Location.Longitude);
    map1.ZoomLevel = 15;
```

KUVA 33. GeoCoordinateWatcher- luokan tapahtumat

StatusChanged- tapahtuma aktivoituu, kun paikannuspalvelun tilassa tapahtuu muutoksia. Kyseisissä tapauksissa suoritetaan kuvan 33 mukaiset käskyt, jotka ilmoittavat erillisellä ponnahdusikkunalla käyttäjälle virheilmoituksen ilmenneestä viasta. Tässä tapauksessa ohjelma ilmoittaa, jos GPS- paikannus ei ole laitteessa aktiivinen, tai GPS- dataa ei ole mahdollista vastaanottaa esim. huonon sijainnin takia.

PositionChanged- tapahtuma taas aktivoituu, kun palvelu huomaa muutoksia sijainnissa. Sijainnin määrittämisen aikana tapahtuma näyttää erillisessä tekstikentässä viestiä ”*Searching you...*”, mutta muissa tilanteissa karttanäkymään lisätään *pushpin*- elementti ilmaisemaan käyttäjän omaa sijaintia. *Pushpin*- elementin sijainti määritetään *GeoCoordinateWatcher*- luokan omilla komennoilla: *e.Position.Location.Latitude* ja *e.Position.Location.Longitude*, jotka ilmaisevat yhtä koordinaattipistettä kartalle sijoitettaessa. Samalla karttakuvaa zoomataan *ZoomLevel*- määritteen avulla tarkemmaksi käyttäjää varten.



KUVA 34. Oman sijainnin määrittäminen kartalle

Käytännössä ohjelma toimii kuvan 34 esittämällä tavalla. Kuvan vasemmassa laidassa näkyy Windows Phone- emulaattori, joka suorittaa äsken selitettyä ohjelma rakennetta. Oikeassa laidassa on emulaattorin lisätyökalut, joiden avulla voidaan mm. määrittää GPS- sijainti virtuaalisesti erillisen karttakuvan avulla. Kun sijainti on määritetty lisätyökalun kartalle, on emulaattorin noutama GPS- sijainti sen mukainen. Kyseistä

GPS- sijainnin muuttamista voidaan suorittaa samanaikaisesti emulaattorin ollessa aktiivinen, jolloin voidaan todeta, että ohjelman *PositionChanged*- tapahtuma varmasti toimii.

Lisätyökalujen karttakuvassa näkyy useampi sijaintimerkintä, jotka ovat numerojärjestyksessä. Emulaattorin suorittamaa sovellusta on seurattu siten, että lisätyökalujen sijaintia muutettaessa, on muuttunut myös emulaattorin havaitsema GPS- sijainti. Myös *pushpin*- elementti on liikkunut aina oikeaan sijaintipisteeseen, kuten kuvasta 34. voidaan huomata.

7.4.2 JSON- tiedonsiirto ja datan purkaminen

Seuraavaksi tärkein osa-alue ohjelman kannalta on osata muodostaa ohjelmallisesti kutsuja, joiden avulla RuuviTracker- palvelin osaa vastata oikeilla tiedoilla. Kaikki tiedonsiirto tapahtuu JSON- muodossa, joten ohjelman täytyy osata parsia, eli purkaa palvelimen lähettämät JSON- vastaukset ohjelman ymmärtämään muotoon. RuuviTracker- palvelimen rajanpinta tukee URL- muotoisia kutsuja, eli käytännössä kutsut muodostetaan samalla tavalla, kuin internet sivustojen osoitteet. Ohjelmallisesti prosessi toteutetaan muodostamalla ohjelman sisäinen *WebClient*- palvelu, joka osaa lähettää kutsun palvelimelle URL- muodossa.

Aluksi tarvitsee selvittää palvelimelle lähetettävien kutsujen rakenne, jotta saadaan siirrettyä vain haluttua tietoa. RuuviTracker- palvelimen ja oman ohjelmointityöni tapauksessa tarvitaan muodostaa käskyjä, joiden avulla saadaan selvitettyä palvelimen tila, sekä paikannettavan laitteen tiedot ja koordinaatit. Kutsujen muodostaminen tapahtuu RuuviTracker- palvelimen osoitteen mukaisesti: http://dev-server.ruuvitracker.fi/api/v1-dev/. Kun halutaan lisätä osoitekenttään esim. palvelimen tilan selvittävä *ping*- kutsu, niin muodostetaan kutsu seuraavalla tavalla: http://dev-server.ruuvitracker.fi/api/v1-dev/ping. Kun palvelin on vastaanottanut ohjelman lähettämän kutsun, se lähettää takaisin JSON- muotoisen tiedoston, josta selviää RuuviTracker- palvelimen tila. Käytännössä JSON- tiedosto näyttää kuvan 35 mukaiselta.

```
{"ruuvi-tracker-protocol-version":"1",
 "server-software":"CLJ-RTServer/0.0.1",
 "time": "2012-03-18T21:08:38.899+02:00"}
```


KUVA. 35. RuuviTracker- palvelimen vastaus ping- kutsuun

JSON- tiedostomuoto näyttää melko sekavalta, mutta se sisältää tarkemmin katseltuna selvän hierarkisen rakenteen. Ihmissilmälle kyseinen tekstijono ei tuota paljon iloa, joten tieto täytyy ohjelmallisesti purkaa ymmärrettävämpään muotoon. Kyseistä toimenpidettä varten olen käyttänyt kappaleessa 7.1 mainittua *JSON.NET*- kirjastoa, jonka avulla on helppo käsitellä JSON- tietoa.

```
private void button2_Click(object sender, RoutedEventArgs e)
{
    try
    {
        WebClient webClient = new WebClient();
        webClient.DownloadStringCompleted += new DownloadStringCompletedEventHandler(webClient_DownloadStringCompleted);
        webClient.DownloadStringAsync(new Uri("http://dev-server.ruuvitracker.fi/api/v1-dev/trackers/9"));

    }
    catch (Exception ex)
    {
        MessageBox.Show(ex.Message);
    }
}
```

KUVA 36. WebClient- toiminnon muodostaminen painonapilla

Kuvassa 36 on esitelty ohjelmointityöni etusivulla sijaitsevan painonapin ohjelmakoodia. Kyseinen toiminto muodostaa uuden *WebClient*- elementin lähettämään RuuviTracker- palvelimelle http://dev-server.ruuvitracker.fi/api/v1-dev/trackers/9 kutsun, jonka avulla on mahdollista selvittää perustiedot ohjelmointityöni apuna olleesta paikannuslaitteesta. Kuvan 36 koodi myös ottaa vastaan palvelimen lähettämän JSON-muotoisen vastauksen ja siirtää sen purettavaksi.

```
public class RootObject
{

    public List<trackers> trackers { get; set; }
}

void webClient_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
{
    var rootObject = JsonConvert.DeserializeObject<RootObject>(e.Result);

    foreach (var trackers in rootObject.trackers)
    {
        listBox1.Items.Add(trackers);
        break;
    }

}
```

KUVA 37. JSON- tiedon purkaminen ohjelman tuettavaksi

Kuvassa 37 esitetään, kuinka JSON- tiedon purkaminen tapahtuu ohjelmallisesti. Kuvassa 36 esiintynyt *WebClient*- toiminto siirtää JSON- tiedon käsiteltäväksi välittömästi sen latauksen valmistuttua. Purkaminen tapahtuu *JsonConvert.Deserialize*- metodilla, joka purkaa tiedon Windows Phone- järjestelmän ymmärtämäksi listaksi, joka esiintyy kuvassa 37 *RootObject*- luokkana. Kyseinen luokka tarvitsee apuvälineeksi ohjelmakoodin käytettäväksi määritteet JSON- datasta etsittäville tiedoille, joita ovat tässä tilanteessa kuvassa 38 esiintyvät tiedot: *created_on, name, tracker_code* ja *id*. Kyseisten määritteiden avulla ohjelma osaa paikantaa JSON- tiedoista juuri halutut tiedot. Lopuksi ohjelmaan on määritelty *foreach*- lause, joka käy läpi kaikki JSON-datan sisältämät tiedot ja tallettaa jokaisen kuvan 38 luokan kriteerit täyttävän tietojoukon aiemmin mainittuun listaan.

```
public class trackers
{
    public string created_on { get; set; }
    public string name { get; set; }
    public string tracker_code { get; set; }
    public int id { get; set; }
}
```

KUVA 38. JSON- datan talletettavat tiedot

Kuvassa 39 esitellään JSON- tietojen purkua Windows Phone- emulaattorin avulla. Kuvassa näkyvä *Find Tracker*- painike on juuri aktivoitu, jolloin *WebClient*- toiminto on lähettänyt palvelimelle kutsun ja saatu vastaus on purettu yllä olevien koodien mukaisesti. Puhelinnäkymään on luotu XAML- koodin avulla vierityslista haettujen tieto-

jen tulostamista varten. Kuvasta huomataan, kuinka yksittäisen RuuviTracker- laitteen tiedot saadaan tulostettua omien määritteiden mukaisesti.

KUVA 39. Purettua JSON- dataa

7.4.3 RuuviTracker- laitteen sijainnin määrittäminen

Lopuksi hyödynnetään JSON- tietojen kautta ladattavia RuuviTracker- laitteen koordinaattitietoja ohjelman karttanäkymässä, jolloin on mahdollista tutkailla samanaikaisesti yhden karttakuvan avulla niin Windows Phone- laitteen kuin RuuviTracker- laitteen sijainteja. Periaatteeltaan JSON- tietojen käsittely tapahtuu samalla tavalla kuin edellisessä kappaleessa, jossa palvelimelta noudettiin paikannettavan laitteen tietoja. Erona edelliseen prosessiin on se, että ns. raakana noudettu tieto ei sovellu suoranaisesti *GeoCoordinate*- luokan käytettäväksi, vaan se vaatii hieman muutakin käsittelyä.

```
public class location
{
    public string speed { get; set; }
    public string heading { get; set; }
    public string latitude { get; set; }
    public string longitude { get; set; }
}

public class events
{
    public location location { get; set; }
    public string store_time { get; set; }
    public string event_session_id { get; set; }
    public string tracker_id { get; set; }
    public string event_time { get; set; }
    public string id { get; set; }
}
```

KUVA 40. RuuviTracker- laitteen sijaintitietojen luokat

Kuten kuvasta 40 huomataan, niin paikannustietojen purkamiseen vaaditaan kaksi eri luokkaa, sillä JSON- tiedosto on rakenteeltaan hieman erilaisempi kuin aikaisemmassa kappaleessa purettu tieto. Vaikka olen ohjelmaan lisännytkin molemmat luokat, niin käytännössä tarvitaan vain ylemmän *location*- luokan tietoja. Toin ohjelmassa käsiteltäväksi myös *events*- luokan tiedot, jotta JSON- datan rakennetta olisi helpompi havainnollistaa. Kyseisen tiedon rakenne kattaa tarvittavan *location*- luokan tiedot *events*- luokan sisälle, joten kaikkien tietojen on hyvä olla esillä.

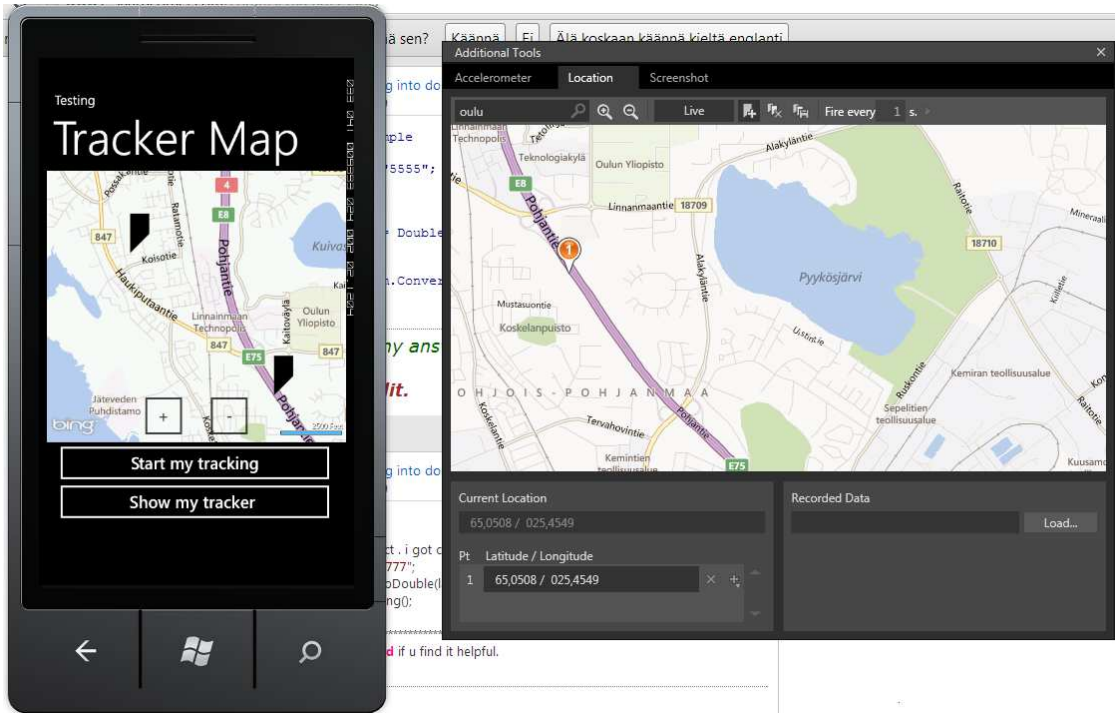
```
public class RootObject
{

    public List<events> events { get; set; }
}

void webClient_DownloadStringCompleted(object sender, DownloadStringCompletedEventArgs e)
{
    var rootObject = JsonConvert.DeserializeObject<RootObject>(e.Result);

    foreach (var events in rootObject.events)
    {
        double latitude_double = Double.Parse(events.location.latitude);
        double longitude_double = Double.Parse(events.location.longitude);

        Pushpin pin = new Pushpin();
        pin.Location = new GeoCoordinate(latitude_double, longitude_double);
        map1.Children.Add(pin);

        map1.Center = new GeoCoordinate(latitude_double, longitude_double);
        map1.ZoomLevel = 10;
        break;
    }

}
```

KUVA 41. JSON- tietojen purkaminen kartalle näytettäväksi

Ohjelman *WebClient*- ominaisuus toteutetaan aivan samalla tavalla kuin edellisen kappaleen esimerkissä, paitsi URL- muotoinen käsky on rakenteeltaan seuraavanlainen: http://dev-server.ruuvitracker.fi/api/v1-dev/trackers/9/events/latest. Kyseinen kutsu pyytää palvelinta lähettämään käytettävän RuuviTracker- laitteen viimeisimmät tapahtumatiedot, joihin sisältyy kuvassa 40 esiintyvät luokkatiedot.

Kuva 41 esittää ohjelmassa tapahtuvia toimintoja, jotka aktivoituvat välittömästi JSON- tiedoston latauksen valmistuttua. Tarvittavien luokkatietojen paikannus tapahtuu samalla *foreach*- lauseella, jota käytettiin aikaisemmankin kappaleen esimerkissä. Kyseisen lauseen yhteyteen jouduttiin lisäämään operaattorit, jotka hakevat JSON-datan ensimmäiset *latitude*- ja *longitude* tiedot, sillä käyttöön haluttiin tietenkin vain RuuviTracker- laitteen viimeisin sijainti. Tämän takia *foreach*- lause sisältää *break*;-komennon lopussa, sillä muutoin ohjelma kävisi läpi kaikki palvelimen lähettämät sijaintitiedot, joita olisi useita kymmeniä.

Seuraavaksi huomattiin, että JSON- tiedostosta puretut sijaintitiedot olivat *string*-tyyppiä, joka taas tarkoitti sitä, että käyttämäni *GeoCoordinate*- toiminto ei kyennyt tunnistamaan niitä. Tämän seurauksena *foreach*- lauseen alkuun on määritelty toiminnot, joiden avulla JSON- tiedoista puretut *latitude*- ja *longitude*- arvot ovat muunnettu *GeoCoordinate*- toiminnon tukemaan *double*- muotoon. Kun kyseiset määritykset oli selvitetty, niin JSON- tiedostosta purettuja ja muunnettuja *latitude_double*- ja *longitude_double*- arvoja voitiin hyödyntää aivan samalla tavalla, kuin kappaleessa 7.4.1.



KUVA 42. RuuviTracker- laitteen ja Windows Phone- laitteen saman aikainen paikannus Map- sivulla

Kuvassa 42 esitellään ohjelman toimintaa emulaattorin avulla ja näytetään sekä Windows Phone- laitteen, että RuuviTracker- laitteen samanaikaiset sijainnit yhdellä karttakuvalla. Kuvan oikean laidan emulaattorin lisätyökalujen avulla on määritetty Windows Phone- laitteen virtuaalinen sijainti, jolloin sama sijainti merkitään *pushpin*- elementillä ohjelman karttakuvalle *Start my tracking*- painikkeen aktivoituessa. Samalla voidaan aktivoida myös *Show my tracker*- painike, jolloin kartalle ilmestyy toinen *pushpin*- elementti ilmaisemaan RuuviTracker- laitteen sijaintia. Ohjelman Info-sivu on rakennettu samojen periaatteiden perusteella kuin karttasivu ja siitä voidaan myös havaita molempien laitteiden samanaikainen paikannusmahdollisuus kuvan 43 esittämällä tavalla.

KUVA 43. RuuviTracker- laitteen ja Windows Phone- laitteen samanaikainen paikannus Info-sivulla

8 JOHTOPÄÄTÖKSET

Näin heti alkuun haluan todeta, että en ollut täysin tyytyväinen ohjelmointityöni lopputulokseen, sillä aihe oli hieman vaikeampi mitä alun perin osasin odottaa. Suurimmiksi vaikeuksiksi muodostuivat kiireelliset aikataulut, sekä paljon aikaa vaatinut asioiden opettelu ja tekstin kirjoittaminen. Olen kuitenkin tyytyväinen siihen, että onnistuin ratkaisemaan työn alussa asettamani tavoitteet edes perustoiminnoiltaan. Työtä aloittaessa Windows Phone- ympäristö oli minulle aivan uusi asia, joten jouduin opettelemaan liki main kaiken alusta alkaen.

Päätin toteuttaa ohjelmointityön käyttöliittymän hyödyntämään useampaa eri ohjelmasivua, sillä kyseisen mallin rakentaminen oli minulle hieman tutumpaa Visual Studio- ympäristöstä. Tutustuin myös Windows Phone- ympäristöissä eniten käytettyihin ratkaisuihin, kuten panoraamanäkymään, mutta totesin niiden opettelun vievän liikaa aikaa. Sain kuitenkin aikaiseksi käyttöliittymän, jonka avulla ainakin omasta mieles-

täni on helppo demonstroida Windows Phone- laitteen perustoimintoja, kuten esimerkiksi karttaelementtejä.

Eniten olin tyytyväinen *GeoCoordinate*- luokan käyttöönottamisessa ja Windows Phone- laitteen oman sijainnin määrittämisessä, sillä sain kyseisen osion toimimaan mielestäni parhaiten omassa työssäni. Toiminto mahdollisti laitteen sijainnin esittämisen myös jatkuvassa liikkeessä hyödyntäen paikannustietoja tarkkailevaa palvelua, sekä visuaalisesti sijaintia esittävää *pushpin*- elementtiä karttanäkymässä.

Suurimmat vaikeudet ilmenivät JSON- tietojen siirtämisessä, sillä siihen alueeseen liittyvät asiat olivat minulle aivan uusia. Kyseinen aihealue vei myös ehdottomasti eniten aikaa ohjelmointityöstä, koska erilaisia toimintatapoja oli melkoisen paljon. Löysin kuitenkin lopulta omasta mielestäni toimivimman ratkaisun lisäämällä ohjelmaani työssä mainitsemani Newtonsoftin valmistaman JSON.NET- kirjaston, joka oli käytettävyydeltään mielestäni paljon parempi, kuin esim. .NET- ympäristön tarjoamat vakiotyökalut.

Toinen vaikeus ilmeni paikannustietojen siirtämisessä palvelimelta, sillä yhden kutsukäskyn lähettäminen palautti ohjelmalle useita kymmeniä paikannustietoja. Kyseisessä tapauksessa minun olisi tarvinnut saada käytettäväksi vain viimeisin sijaintitieto, joten JSON- datasta täytyi saada otettua talteen vain ensimmäiset luokkatiedot. Toteutin kyseisen toiminnon samalla *foreach*- lauseella, kuin muissakin tapauksissa, mutta lisäsin lauseen sisälle *break*- komennon keskeyttämään tietojen listauksen ensimmäisen luokan määrityttyä. Olen sitä mieltä, että kyseisen toiminnon voisi toteuttaa paljon järkevämmälläkin tavalla, mutta aika loppui kesken.

Tarkoituksenani on jatkaa sovelluksen kehittämistä Windows Phone- ympäristöön opinnäytetyön valmistuttua ja myöhemmin myös julkaista valmis sovellus julkiseen käyttöön. Ensimmäiseksi minun on tarkoitus suunnitella sovelluksen käyttöliittymä kokonaan uusiksi vastaamaan enemmän RuuviTracker- järjestelmän ominaisuuksia, sekä Windows Phone- ympäristön visuaalisuutta. Mielestäni tämän hetkinen ohjelmointityöni on karun näköinen ja suppea toiminnoiltaan, mutta sain sillä tuotua esille tärkeimpiä ominaisuuksia, joita RuuviTracker- järjestelmän avulla on mahdollista hyödyntää. Sovellus toimii myös varmasti hyvänä pohjana jatkokehitykselle, sekä muille asiasta kiinnostuneille kehittelijöille.

9 LOPUKSI

Kuten opinnäytetyön alussakin mainitsin, niin lähdin tekemään kyseistä työtä täysin oma-aloitteisesti ilman toimeksiantajaa. Kiinnostuin Lauri Jämsän aloittamasta RuuviTracker- projektista ja halusin osallistua mukaan kehittämään projektia oman osaamiseni mukaan. Mielestäni tähän oli loistava mahdollisuus, koska projekti aloitettiin harrastelijapohjalta, joka ei myöskään vaatinut osallistuvilta jäseniltä tarkempia velvoitteita.

RuuviTracker- järjestelmä koostuu valmiiksi rakennetusta ympäristöstä, joka mahdollistaa RuuviTracker- laitteiden etäpaikannuksen. Samanlaisia järjestelmiä on ollut julkisessa käytössä jo pitkän aikaa, mutta RuuviTracker tuo harrastelijoille mahdollisuuden soveltaa ja kehittää järjestelmää laajempiin tarkoituksiin ilman suurempia rahallisia sijoituksia. Oman näkemykseni mukaan kyseessä on kenties maailman ensimmäinen valmiiksi rakennettu järjestelmä, joka on täysin avoin ja kenen tahansa kehitettävissä.

Tiesin aivan alusta alkaen, että opinnäytetyön tekeminen RuuviTracker- aiheesta olisi vaativa projekti, vaikka järjestelmän perustoiminnoista ja ohjelmointitekniikoista minulla olikin kokemusta. Jouduin kuitenkin opettelemaan kaikkea järjestelmään liittyvää perustietoa perusteellisemmin, sekä opettelemaan Windows Phone- ohjelmoinnin toiminnallisuuden alusta alkaen. Kaikki opeteltava tieto oli kuitenkin mielenkiintoista ja asioihin tuli perehdyttyä hyvällä innolla, samalla oppien koko ajan uutta tietoa. Olin myös erittäin tyytyväinen, että pääsin seuraamaan RuuviTracker- laitteen tuotantoa Savonlinnan 3-K tehtaalle, jossa olen myös aikaisemmin ollut työharjoittelussa.

Opinnäytetyötä tehdessä ilmeni myös vaikeuksia ja viivästyksiä, sillä RuuviTracker-projekti oli tätä työtä kirjoitettaessa keskeneräinen ja kehittyvä työ. Omalta kannalta vaikuttavimmat viivästykset johtuivat B- revision RuuviTracker- laitteen koe-erästä, jossa oli useita viallisia laitteita. Tämän takia en saanut testilaitetta käytettäväksi omaa työtäni varten. Projektin kuuluminen avoimen lähdekoodin piiriin, ei kuitenkaan tuottanut ylimääräistä viivettä, kuten olisi alustavasti voinut olettaa. Yllätyin suuresti, kuinka nopeasti projekti eteni suunnittelukaavioista tuotantolinjalle ja siitä testauk-

seen. Myös osaavia alan harrastelijoita ja ammattilaisia liittyi projektiin mukaan yllättävän paljon, joka varmasti myös nopeutti projektin eri osa-alueiden kehitystä.

Loppujen lopuksi olen kuitenkin hyvin tyytyväinen omaan työhöni ja tarkoituksenani on myös jatkaa toimintaa RuuviTracker- järjestelmän parissa järjestelmän parantamiseksi. Ensisijaisin tavoitteeni on laatia lopullinen sovellus Windows Phone- käyttöjärjestelmää varten ja myöhemmin myös julkaista se julkiseen käyttöön. Toivon myös, että opinnäytetyöni toimisi hyvänä tietolähteenä RuuviTracker- järjestelmän kokonaisvaltaisemmalle dokumentoinnille, sekä innoittajana muille RuuviTracker- järjestelmästä kiinnostuneille ihmisille.

LÄHTEET

1. Brain, Marshall. How MicrocontrollersWork. WWW-sivut. http://electronics.howstuffworks.com/microcontroller. Päivitetty 1.4.2000. Luettu 11.3.2013.

2. Engineers Garage. Microcontrollers. WWW-sivut. http://www.engineersgarage.com/microcontroller?page=1. Ei päivitystietoja. Luettu 11.3.2013

3. I²C-Bus. I²C-Bus: What's that?.WWW-sivut. http://www.I²C-bus.org. Päivitetty 30.10.2012. Luettu 11.3.2013.

4. Philips Semiconductors. The I²C-bus Specification. Pdf-dokumentti. http://www.classic.nxp.com/acrobat_download2/literature/9398/39340011.pdf. Päivitetty 1/2000. Luettu 11.3.2013.

5. Kalinsky, David &Kalinsky, Roee. Introduction to Serial Peripheral Interface. WWW-sivut. http://www.embedded.com/electronics-blogs/beginner-s-corner/4023908/Introduction-to-Serial-Peripheral-Interface. Päivitetty 1.2.2002. Luettu 11.3.2013.

6. Lawyer, David S. Serial HOWTO. WWW-sivut. http://www.tldp.org/HOWTO/Serial-HOWTO.html#toc4. Päivitetty 2/2011. Luettu 11.3.2013.

7. Durda, Frank. Serial and UART Tutorial. WWW-sivut. http://www.freebsd.org/doc/en/articles/serial-uart/. Päivitetty 13.1.1996. Luettu 11.3.2013.

8. Goler, Vernon. Using the Universal Asynchronous Receiver Transmitter (UART) eTPU Function. WWW-sivut. http://www.freescale.com/files/32bit/doc/app_note/AN2853.pdf. Päivitetty 10/2004. Luettu 11.3.2013.

9. Robotix. USART-UART. WWW-sivut. http://robotix.in/tutorials/category/avr/usart. Päivitetty 2012. Luettu 11.3.2013.

10. McLaughlin, Richard. Controller Area Network. Pdf-dokumentti. http://www.warwickcontrol.com/ata/Infobackup_and_library/Basicbus.pdf. Ei päivitystietoja. Luettu 11.3.2013.

11. BOSCH. CAN Specification. Pdf-dokumentti. http://www.bosch-semiconductors.de/media/pdf_1/canliteratur/can2spec.pdf. Päivitetty 8/1991. Luettu 11.3.2013.

12. Corelis. JTAG Tutorial. WWW-sivut. http://www.corelis.com/education/JTAG_Tutorial.htm. Ei päivitystietoja. Luettu 11.3.2013.

13. XJTAG. High-level Guide to JTAG. WWW-sivut. http://www.xjtag.com/support-jtag/jtag-high-level-guide.php. Ei päivitystietoja. Luettu 11.3.2013.

14. El-Rabbany, Ahmed. Introduction to Gps: The Global Positioning System. Lontoo: Artech House. 2002.

15. Marshall, Brain & Harris, Tom. How GPS Receivers Work. WWW-sivut. http://electronics.howstuffworks.com/gadgets/travel/gps.htm. Päivitetty 25.9.2006. Luettu 11.3.2013.

16. DePriest, Dale. How Your GPS Works. WWW-sivut. http://www.gpsinformation.org/dale/theory.htm. Päivitetty 14.11.2002.Luettu 11.3.2013.

17. Nubarrón, Jefe& Hill, Simon. Evolution Of Mobile Technology. WWW-sivut. http://www.brighthub.com/mobile/emerging-platforms/articles/30965.aspx. Päivitetty 1.12.2011. Luettu 11.3.2013

18. JDSU. Mobile Networks Tutorial. Pdf-dokumentti. http://www.jdsu.com/ProductLiterature/Mobile_Networks_Tutorial.pdf. Päivitetty 2010. Luettu 11.3.2013.

19. Nurmi, Mikko & Sankari, Markus. NMT vs GSM. WWW-sivut. http://www.tml.tkk.fi/Studies/Tik-110.300/1999/Essays/nmt.html. Päivitetty 1.11.1999. Luettu 11.3.2013.

20. Sonera. 3G ja 4G. WWW-sivut. http://www5.sonera.fi/ohjeet/3G_ja_4G. Ei päivitystietoja. Luettu 11.3.2013.

21. Eberspächer, Jörg, Vögel, Hans-Jörg, Bettstetter, Christian & Hartmann, Christian. GSM Architecture, Protocols and Services. New Jersey: John Wiley & Sons. 2009.

22. Nokia. GSM Air Interface&Network Planning. Pdf-dokumentti. http://www.roggeweck.net/uploads/media/Student_-_GSM_Air_Interface___NW_Planning.pdf. Päivitetty 1/2002. Luettu 11.3.2013.

23. Sicher, Alan & Heaton, Randall. GPRS technologyoverview. Pdf-dokumentti. http://webpc.ciat.cgiar.org/wireless/Documents/2002-gprs_overview.pdf. Päivitetty 2/2002. Luettu 11.3.2013.

24. Sanders, Geoff ,Thorens, Lionel, Reisky, Manfred , Rulik, Oliver &Deylitz, Stefan. GPRS Networks. New Jersey: John Wiley & Sons. 2003.

25. Radio-Electronics.com. GSM Network Architecture. WWW-sivut. http://www.radio-electronics.com/info/cellulartelecomms/gsm_technical/gsm_architecture.php. Ei päivitystietoja. Luettu 11.3.2013.

26. Agilent Technologies. Understanding General Packet Radio Service. Pdf-dokumentti. http://cp.literature.agilent.com/litweb/pdf/5988-2598EN.pdf. Ei päivitystietoja. Luettu 11.3.2013.

27. Järvinen, Jani. Windows Phone sovelluskehitys. Jyväskylä: Docendo Finland oy. 2012.

28. Texas Instruments. BQ24190 Datasheets. Pdf-dokumentti. http://www.ti.com/lit/ds/slusaw5/slusaw5.pdf. Päivitetty 1/2012. Luettu 11.3.2013.

29. Texas Instruments. DRV2603 Datasheets. Pdf-dokumentti. http://www.ti.com/lit/ds/symlink/drv2603.pdf. Päivitetty 4/2012. Luettu 11.3.2013.

30. STMicroelectronics. L3GD20 Datasheets. Pdf-dokumentti. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00036465.pdf. Päivitetty 2/2013. Luettu 11.3.2013.

31. STMicroelectronics. LSM303DLHC Datasheets. Pdf-dokumentti. http://www.st.com/st-web-ui/static/active/en/resource/technical/document/datasheet/DM00036465.pdf. Päivitetty 4/2011. Luettu 11.3.2013.

32. RuuviTracker. RuuviTracker RevB Schematics. Pdf-dokumentti. https://raw.github.com/RuuviTracker/ruuvitracker_hw/master/ruuvitracker_revb_kicad/schematics/ruuvitracker_revb1_schema.pdf. Päivitetty 30.1.2013. Luettu 11.3.2013.

33. Micron Electronics. SIM908 GSM/GPRS+GPS Module. WWW-sivut. http://www.simcom.us/product_detail.php?cid=1&pid=38. Päivitetty. 2013. Luettu 11.3.2013.

34. Developer’s Home. Introduction to AT Commands. WWW-sivut. http://www.developershome.com/sms/atCommandsIntro.asp. Ei päivitystietoja. Luettu 11.3.2013.

35. RuuviTracker. What is it?.WWW-sivut. http://www.ruuvitracker.fi/. Ei päivitystietoja. Luettu 11.3.2013.

36. Digikey. STM32 F4 series of high-performance MCUs with DSP and FPU instructions. WWW-sivut. http://www.digikey.com/product-highlights/us/en/stmicroelectronics-stm32-f4/1486. Ei päivitystietoja. Luettu 11.3.2013.

37. PIC Tutorials. What is a Microcontroller?.WWW-sivut. http://www.pictutorials.com/what_is_microcontroller.htm. Ei päivitystietoja. Luettu 12.3.2013.

38. Karbo, Michael. PC Architecture. WWW-sivut. http://karbosguide.com/. Ei päivitystietoja. Luettu 12.3.2013.

39. LayadCircuits. UART Basics. WWW-sivut. http://www.layadcircuits.com/layad_articles/UART_Basics.htm. Ei päivitystietoja. Luettu 12.3.2013.

40. RuuviTracker. FAQ. WWW-sivut. http://wiki.ruuvitracker.fi/wiki/FAQ. Ei päivitystietoja. Luettu 12.3.2013.

41. RuuviTracker. Architecture. WWW-sivut. http://wiki.ruuvitracker.fi/wiki/Architecture. Ei päivitystietoja. Luettu 12.3.2013.

42. Swallow, Keith. A Simple Web Server in Clojure. WWW-sivut. http://keithcelt.com/a-simple-web-server-in-clojure. Päivitetty 2012. Luettu 12.3.2013.

43. Jämsä, Lauri. Tämän halusit RuuviTrackerista tietää!. WWW-sivut. http://www.ruuvipenkki.fi/2012/09/18/taman-halusit-ruuvitrackerista-tietaa. Päivitetty 18.9.2012. Luettu 12.3.2013.

44. Codehaus foundation. Jetty. WWW-sivut. http://jetty.codehaus.org/jetty/. Päivitetty 12.4.2011. Luettu 12.3.2013.

45. Heroku. Cloud application platform. WWW-sivut. http://www.heroku.com/. Ei päivitystietoja. Luettu 12.3.2013.

46. Hickey, Rich. Clojure. WWW-sivut. http://clojure.org/. Päivitetty 2012. Luettu 12.3.2013.

47. Tilkov, Stefan. A Brief Introduction to REST. WWW-sivut. http://www.infoq.com/articles/rest-introduction. Päivitetty 10.12.2007. Luettu 12.3.2013.

48. PostgreSQL. About. WWW-sivut. http://www.postgresql.org/about/. Ei päivitystietoja. Luettu 12.3.2013.

49. SIMCom. SIM908 Hardware Design. Pdf-dokumentti. http://www.reyax.com/Module/GSM/SIM908/SIM908_Hardware_Design_V1.00.pdf. Päivitetty 28.6.2011. Luettu 12.3.2013.

50. SIMCom. SIM908 AT Command Manual. Pdf-dokumentti. http://www.simcom.us/act_admin/supportfile/SIM908_AT%20Command%20Manual_V1.01.pdf. Päivitetty 7.7.2011. Luettu 12.3.2013.

51. eLUA. Overview. WWW-sivut. http://www.eluaproject.net/overview. Ei päivitystietoja. Luettu 12.3.2013.

LIITE 1.

Ohjelmointityön XAML- koodia Map- sivulta

```
<phone:PhoneApplicationPage
    x:Class="PhoneApp7.MapPage"
    xmlns="http://schemas.microsoft.com/winfx/2006/xaml/presentation"
    xmlns:x="http://schemas.microsoft.com/winfx/2006/xaml"
    xmlns:phone="clr-namespace:Microsoft.Phone.Controls;assembly=Microsoft.Phone"
    xmlns:shell="clr-namespace:Microsoft.Phone.Shell;assembly=Microsoft.Phone"
    xmlns:d="http://schemas.microsoft.com/expression/blend/2008"
    xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
    FontFamily="{StaticResource PhoneFontFamilyNormal}"
    FontSize="{StaticResource PhoneFontSizeNormal}"
    Foreground="{StaticResource PhoneForegroundBrush}"
    SupportedOrientations="Portrait" Orientation="Portrait"
    mc:Ignorable="d" d:DesignHeight="768" d:DesignWidth="480"
    shell:SystemTray.IsVisible="True"
    xmlns:my="clr-namespace:Microsoft.Phone.Controls.Maps;assembly=Microsoft.Phone.Controls.Maps"
    xmlns:my1="clr-namespace:System.Device.Location;assembly=System.Device">

    <!--LayoutRoot is the root grid where all page content is placed-->
    <Grid x:Name="LayoutRoot" Background="Transparent">
        <Grid.RowDefinitions>
            <RowDefinition Height="Auto"/>
            <RowDefinition Height="*"/>
        </Grid.RowDefinitions>

        <!--TitlePanel contains the name of the application and page title-->
        <StackPanel x:Name="TitlePanel" Grid.Row="0" Margin="12,17,0,28">
            <TextBlock x:Name="ApplicationTitle" Text="Testing"
                       Style="{StaticResource PhoneTextNormalStyle}"/>
            <TextBlock x:Name="PageTitle" Text="Tracker Map" Margin="9,-7,0,0"
                       Style="{StaticResource PhoneTextTitle1Style}"/>
        </StackPanel>

        <!--ContentPanel - place additional content here-->
        <Grid x:Name="ContentPanel" Grid.Row="1" Margin="12,0,12,0">
            <my:Map Height="412" HorizontalAlignment="Left" Name="map1" VerticalAlignment="Top" Width="456"
                    CredentialsProvider="[redacted]"
                    ZoomLevel="4.2" Margin="0,-22,0,0" ZoomBarVisibility="Visible" CopyrightVisibility="Collapsed"
                    ScaleVisibility="Visible">
                <my:Map.Center>
                    <my1:GeoCoordinate Altitude="NaN" Course="NaN" HorizontalAccuracy="NaN" Latitude="64" Longitude="26"
                                       Speed="NaN" VerticalAccuracy="NaN" />
                </my:Map.Center>
            </my:Map>
            <Button Content="Start my tracking" Height="73" HorizontalAlignment="Left" Margin="9,385,0,0" Name="button1"
                    VerticalAlignment="Top" Width="433" Click ="button1_Click" />
            <Button Content="Show my tracker " Height="73" HorizontalAlignment="Left" Margin="9,444,0,0" Name="button2"
                    VerticalAlignment="Top" Width="433" Click="button2_Click" />
            <TextBlock Height="69" HorizontalAlignment="Left" Margin="12,523,0,0" Name="notification" Text=""
                       VerticalAlignment="Top" Width="430" />
        </Grid>
    </Grid>

</phone:PhoneApplicationPage>
```

Tuotantoprosessi

RuuviTracker- laitteen B- revision piirilevyt tuotettiin Savonlinnassa 3K nimisessä elektroniikkayrityksessä, jonne pääsin seuraamaan tuotantoprosessia alusta loppuun. Tuotantoprosessi koostuu pääosin kolmesta erilaisesta vaiheesta, mutta mukaan kuuluu myös paljon erilaisia laadunvalvontaprosesseja, jotta lopputulos säilyy mahdollisimman hyvänä.

Pastaus

Kun tyhjät piirilevyt ovat saatu valmiiksi, niin niiden perusteella valmistetaan stensiilit, joiden avulla juotospasta saadaan asetettua piirilevylle juotospaikkoihin.Stensiilit ovat hyvin ohuita metallilevyjä, joihin tehdään reiät piirilevyjen juotospaikkojen mukaan. Kyseessä on eräänlainen maski, joka pastausprosessin aikana estää pastan päätymisen muualle, kuin niille tarkoitetuille paikoille.

Juotospasta koostuu yleisimmin tinapulverista, johon on sekoitettu nestemäistä juoksutetta. Pasta itsessään on tahmeaa ainetta, joka pysyy piirilevylle asetettuna melko hyvin paikallaan, sekä pitää pastan päälle asetetut komponentit kiinni, kunhan levyjä ei pahemmin heilutella. Kun juotospastaa lämmitetään, niin se toimii samalla tavalla kuin tinalanka, eli kiinnittää komponentit piirilevyn juotospaikoille, joita kutsutaan myös ”pädeiksi”.

Pastausprosessi suoritetaan asettamalla stensiili piirilevyjen päälle, siten että stensiilin reiät kohdistuvat piirilevyjen juotospaikkojen mukaisesti. Pastauksessa käytettiin alla olevassa kuvassa 1. näkyvää DEK 265 Horizon 03-laitetta, joka pyyhkäisee juotospastan stensiilin ylitse, annostellen pastaa piirilevyjen juotospaikoille juuri oikean määrän. Pastauksen jälkeen levyt tarkistetaan silmämääräisesti, että kohdistukset ovat sijoittuneet oikein, eikä levyillä ole ylimääräistä pastaa.

KUVA 1. Juotospastan levitys piirilevyille

Komponenttien ladonta

Pastauksen jälkeen piirilevyille voidaan ruveta latomaan komponentteja niiden oikeille paikoille. Juotospasta pitää niiden päälle asetetut komponentit paikoillaan suhteellisen hyvin, eikä yleensä tarvita mitään muita ylimääräisiä kiinnitysaineita. Ladontaprosessi on hyvin tarkkaa työtä, sillä jos komponentti liikahtaa sitä asetettaessa juotospaikalleen, niin se saattaa sotkea juotospastaa. Tämä voi aiheuttaa kuumennusvaiheessa komponenttien välille ylimääräisiä tinasiltoja tai pienimmät komponentit voivat jopa liikkua pois paikoiltaan.

Ladontaprosessissa käytetään Mydata TP9-UFP-laitetta, joka on suunniteltu pintaliitoskomponenttien ladontaa varten. Ennen ladontaprosessia ladontakoneen tietokone ohjelmoidaan ladottavan piirilevyn elektroniikkasuunnitelmien mukaisesti, joten laite tietää mitä komponentteja sen täytyy piirilevyille latoa. Kun tietokoneelle on määritelty käytettävät komponentit, niin niiden ladontatapakin täytyy määritellä. Jotkut komponentit täytyy latoa tiettyyn asentoon, kuten LED-valoilla on tietynlainen polarisaatio suunta. Asetusten määrittelyn jälkeen komponentit kohdistetaan piirilevyille oikeisiin

paikkoihinsa ladontakoneen sisäisen kameran avulla. Kamera näyttää tietokoneen näytölle tarkennettua kuvaa piirilevystä, sekä ladottavan komponentin kehyskuvan. Kehyskuvat kohdistetaan komponenttien juotospaikkojen päälle ja sijainnit talletetaan koneen muistiin.

Kone tarvitsee myös piirilevyille ladottavat komponentit, jotka kytketään koneen syöttölaitteeseen. Syöttölaite osaa hyödyntää kokonaisia komponenttirullia, joten yksittäisten komponenttien ladonta kyseisellä koneella on liki mahdotonta. Kun kaikki tarvittavat määrittelyt on suoritettu, kone alkaa latoa komponentteja piirilevyille yksi kerrallaan. Ladontanopeus vaihtelee komponenttien määrästä riippuen, mutta keskimääräinen nopeus pysyy yleensä yhdessä sekunnissa per komponentti. Ladontaprosessissa voi joskus esiintyä pienimuotoisia ongelmia, jotka johtuvat useimmiten laitteen syöttöhäiriöistä tai piirilevyjen liikahtamisesta johtuvista kohdistusongelmista. Ladontakoneen tietokone ilmoittaa prosessin aikana ilmenneistä virheistä, joten puuttuvat tai huonosti ladotut komponentit on helppo paikantaa. Ladontaprosessin seuraaminen on myös tärkeää, sillä ensikertaa ladottaville levyille ilmenee helpommin virheitä. Virheet pystyy kuitenkin seuratessa huomaamaan nopeasti ja tarvittaessa korjaamaan kesken ladonnan.

KUVA 2. Pintaliitoskomponenttien ladontakone

Yllä olevassa kuvassa 2. näkyy keskellä sijaitseva ladontayksikkö, joka noutaa vasempaan reunaan kiinnitetyistä komponenttirullista komponentteja ja latoo ne piirilevyille oikeisiin paikkoihin.

Ladontaprosessin päätteeksi tuotantolaatu tarkistetaan mikroskoopin avulla ja suoritetaan tarvittavat toimenpiteet virheiden korjaamiseksi. Puuttuvat komponentit tarkastetaan ja lisätään levyille myöhemmin käsin. Jos puuttuvia komponentteja on useampia, niin niiden ladonta kannattaa ajan säästämisen kannalta suorittaa ladontakoneella yksittäisinä lisäyksinä, joka on myös mahdollista.

Käsinladonta

Piirilevyillä on usein myös sellaisia komponentteja, jotka eivät käy ladontakoneeseen. Tällaiset komponentit joudutaan latomaan levyille käsin. Käsinladontaa ei onneksi tarvitse suorittaa aivan omien käsien varassa, sillä 3K-tehtaalla on käytössä useampia laitteita, jotka helpottavat ladontatyötä.

Ensimmäisenä apuvälineenä käytetään Fineline SMFL-3000-merkkistä laitetta, jolla pieniä komponentteja saadaan aseteltua piirilevyille suurennuslasin ja komponenttien asennuslaitteiston avulla. Komponenttien keräin pitää komponentit otteessaan alipaineen avulla, jonka jälkeen ne asetetaan piirilevylle oikeisiin paikkoihinsa. Asennuslaite pysyy jatkuvasti tasaisessa asennossa, joten pieniäkin komponentteja on suhteellisen helppo asettaa paikoilleen.

KUVA 3. Metcal BGA-3592-juotoslaitteisto

Toinen apuväline on tarkoitettu komponenteille, jotka vaativat suurempaa tarkkuutta. Yllä näkyvässä kuvassa 3. on Metcal BGA-3592-laitteisto, joka on tarkoitettu mm. BGA- komponenttien asennusta varten, joiden juotospisteet sijaitsevat piirien alapuolella. Kyseisiä komponentteja on liki mahdotonta kohdistaa piirilevyille tarkasti, sillä juotospaikat jäävät piirien alapuolelle näkymättömiin. Metcal- laitteisto koostuu useammasta kamerasta, jotka ovat sijoitettu siten, että näytölle piirretty kuva muodostuu samanaikaisesti piirilevystä, sekä sille asetettavan komponentin tulevasta sijainnista. Tällä tavoin esim. BGA komponentitkin pystytään asentamaan piirilevyille, koska asennettavan piirin tuleva sijainti pystytään näkemään tarkasti koko asennuksen ajan.

Reflow- juotosprosessi

Kun kaikki komponentit on saatu ladottua piirilevyille ja mahdolliset virheet korjattua, siirrytään itse juotosprosessiin. Reflow- juotoskone on eräänlainen lämmitetty tunneli, jonka läpi piirilevyt kulkevat ja juotospasta kiinnittää komponentit piirilevyjen juotoskohtiin. Juotoskone on suunniteltu siten, että piirilevy liikkuu tasaista tahtia koneen läpi, joka on lämmitetty erisuuruisiin lämpötiloihin tietyistä paikoista. Järjestelmän on tarkoitus saada aikaan mahdollisimman optimaalinen juotostyö, että juotoskontakteista tulee mahdollisimman kestäviä ja lämpöherkätkin komponentit säilyvät kunnossa prosessin alusta loppuun. Juotosprosessi kestää muutaman minuutin ajan ja lämpötila käy

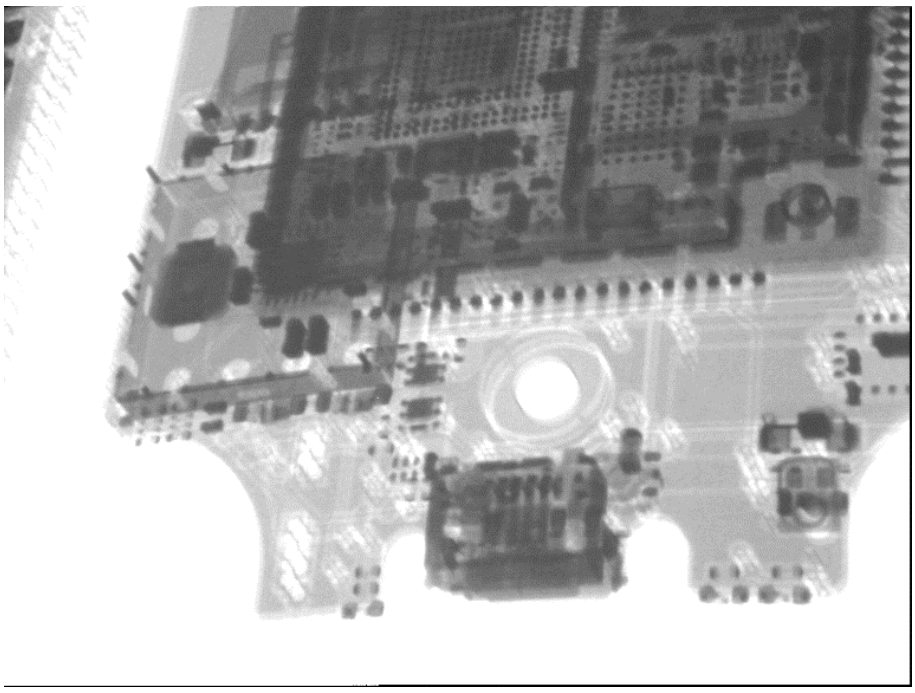
huipussaan +250 celsius asteessa. Laitteen lämmitysjärjestelmä on täysin säädettävissä oleva, joten asetuksia pystyy muuttamaan esim. erilaisia juotospastalaatuja käyttäessä.

Kun levyt saapuvat juotoskoneesta, annetaan niiden jäähtyä rauhassa mahdollisimman paikoillaan, jotta juotokset asettuvat varmasti. Jäähtymisen jälkeen suoritetaan juotoslaadun tarkastus, jossa mahdolliset juotosvirheet merkataan levyihin korjaamista varten. Yleisimpiä virheitä ovat tinasillat kontaktien välillä, jossa niitä ei pitäisi olla. Toisia yleisiä virheitä ovat komponenttien mahdollinen liikehdintä juotosprosessin aikana, joka johtuu yleensä siitä, etteivät komponentit ole asettuneet kunnolla paikoilleen ladontavaiheessa.

Kun kaikki mahdolliset juotosvirheet on saatu paikannettua, ne korjataan käsin tarpeen vaatimien työkalujen avulla. Yleisimmin virheet ovat korjattavissa juotoskolveilla tai korjausasemilla, joiden avulla tinasillat saadaan poistettua, tai juotostinaa lisättyä. Myös komponenttien mahdollisesti tarvittava siirtäminen onnistuu käsityökaluilla, eikä isompia tuotantokoneita tarvitse enää käyttää.

Röntgentarkistus

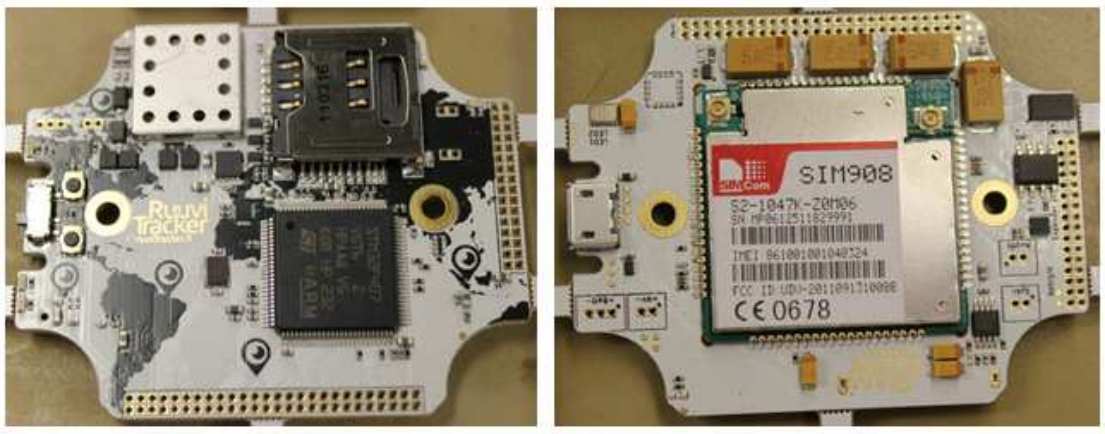
Lopulliseen laaduntarkastukseen kuuluu myös piirilevyjen tarkastaminen niille tarkoitetulla röntgenlaitteistolla. Laitteiston avulla pystytään kuvaamaan valmiita piirilevyjä kerroksittain, joka mahdollistaa ihmissilmälle piilossa olevien virheiden paikantamisen. Laitteella on mahdollista mm. tarkastella piirilevyjen juotoskohtia, vaikka tutkittava elektroniikka olisikin suljettu. Esim. matkapuhelimen juotoksia voi tutkailla purkamatta sitä lainkaan. Laitteisto on myös oivallinen tapa tarkastaa BGA- juotosten lopputulosta, sillä millään muulla tavalla juotoskohtia ei olisikaan mahdollista nähdä. Alla olevassa kuvassa 4 on röntgenlaitteella tallennettu kuva RuuviTracker- laitteen piirilevystä. Kuvan 4 yläosassa nähtävillä myös SIM908- moduulin BGA-piirien juotosjälkeä.



KUVA 4. Röntgenlaitteella kuvattu RuuviTracker- piirilevy

3K-tehdas käyttää Feinfocus FXS-160.4 T.I.G.E.R- järjestelmää, joka koostuu röntgenlaitteistosta, kamerasta, sekä tietokoneyksiköstä. Kaiken ohjauksen voi suorittaa tietokoneelta käsin ja kameran ottamaa kuvaa seurataan monitorilta reaaliajassa. Tämä mahdollistaa myös tilannekuvien tallentamisen tietokoneelle tutkimista varten. Koneella pystytään myös kallistamaan tutkittavia piirilevyjä, joka mahdollistaa laajemman kuvakulman tutkittaville piirilevyille.

Kuvassa 5 esitetään valmiiksi juotettuja RuuviTracker- laitteita kuvattuna etu- ja takapuolelta.



KUVA 5. Valmiiksi juotetut RuuviTracker RevB- piirilevyt