KEMI-TORNION AMMATTIKORKEAKOULU
TEKNIIKKA

Valtonen Juha

Pisteidenlaskujärjestelmän tietokonesovellus

Tietotekniikan koulutusohjelman opinnäytetyö
Ohjelmistotekniikka
Kemi 2009

ALKUSANAT

Kiitän Markku Hietalaa ja Hannu Niemistä, jotka mahdollistivat tämän opinnäytetyön aiheen. Kiitän myös Teppo Aaltoa opinnäytetyöni ohjauksesta.

TIIVISTELMÄ

Kemi-Tornion ammattikorkeakoulu, Tekniikan yksikkö

Koulutusohjelma	Tietotekniikka
Opinnäytetyön tekijä	Juha Valtonen
Opinnäytetyön nimi	Pisteidenlaskujärjestelmän tietokonesovellus
Työn laji	Opinnäytetyö
päiväys	23.11.2009
sivumäärä	27
Opinnäytetyön ohjaaja	FM Teppo Aalto
Yritys	Meri-Lapin Paintball -vuokraamo ja laitesuunnittelu
Yrityksen yhteyshenkilö/valvoja	Markku Hietala

Tässä opinnäytetyössä käsitellään paintballissa käytettävän sulautetun pisteidenlaskujärjestelmän tietokonesovelluksen luontia. Työ on tehty Meri-Lapin Paintball -vuokraamo ja laitesuunnittelu yritykselle. Pisteidenlaskujärjestelmän tekijät näkivät hyväksi, että sulautetun järjestelmän ohella olisi tietokonesovellus, joka auttaisi pelin kulun seuraamista ja sen ohjaamista. Pisteidenlaskujärjestelmässä on tietty määrä kelloja sekä tukiasema, joka kerää tiedot kelloilta ja lähettää ne tietokonesovellukselle. Sovelluksen tarkoitus on antaa informaatiota pelin pisteiden ja vallattujen alueiden tilanteesta. Sovellus antaisi järjestelmälle käskyjä, jotka esimerkiksi käynnistäisivät tai lopettaisivat pelin.

Sovellus on luotu Java-ohjelmointikielellä ja työssä on käytetty NetBeans-ohjelmointiympäristöä. Työssä on myös käytetty Java Communications API -laajennusta, joka ei kuulu Java-ympäristöön oletuksena.

Sovellus kerää tiedot tukiasemalta langattoman radioliikenteen avulla. Sovelluksen kanssa käytetään USB-väylään asetettavaa tikkua, jossa on radiopiiri. USB-tikun avulla sovellus kommunikoi tukiaseman kanssa. Pelin tietojen on tarkoitus näkyä ruudulta selvästi niin, että ohikulkevat pelaajat pystyvät nopeasti katsastamaan pelin tilanteen tietokoneen ruudulta. Sovelluksen on myös tarkoitus kerätä pelin kulku lokitiedostoon, joka mahdollistaa kulun tarkkailun myös jälkikäteen.

Lopputuloksena syntyi toimiva ohjelma, joka kävi läpi yhden kunnollisen testauksen oikeassa pelitilanteessa. Koko pisteidenlaskujärjestelmä sai hyvää palautetta pelaajilta ja myös itse oikea pelitilanne poiki lisää ideoita, joita sovellukseen ja pisteidenlaskujärjestelmään voisi laajentaa. Sovelluksen jatkokehitys jatkuu myös pisteidenlaskujärjestelmän ohella tästä eteenpäin.

Asiasanat: Java, sarjaliikenne, langaton tiedonsiirto.

ABSTRACT

Kemi-Tornio University of Applied Sciences, Technology
Degree Programme Information Technology
Name Juha Valtonen
Title PC application of Points Counting System
Type of Study Bachelor's Thesis
Date 23 November 2009
Pages 27
Instructor Teppo Aalto, MSc
Company Meri-Lapin Paintball -vuokraamo ja laitesuunnittelu
ContactPerson/Supervisor from Company vuokraamo Markku Hietala, Meri-Lapin Paintball - ja laitesuunnittelu

This thesis deals with the creation of computer application for embedded points counting system used in paintball. The work was done to Meri-Lapin Paintball -vuokraamo ja laitesuunnittelu firm. The creators of the points counting system considered that it would be good if there was a computer application which would help to monitor the flow of the game and control it alongside with the embedded system. Points counting system has a certain number of clocks and a base station which collects all information from the clocks and sends it to the computer application. The purpose of the application is to give information about the status of the points and captured areas. The application would also give commands to the system that, for example, would start or stop the game.

The Application was created with Java programming language and NetBeans IDE was used. Java Communications API extension was also used, which is not part of the core Java environment.

The application collects information from the base station via wireless radio communication. An USB stick, which contains radio circuit, is used with the application. With the aid of the USB stick, the application communicates with the base station. The information of the game is meant to be seen clearly on the screen so that passing players can quickly check the situation of the game from the computer screen. The application is also designed to collect the flow of the game in a log file, which allows checking the flow afterwards.

The final result was an effective program that went through one proper testing in a real gaming situation. The whole points counting system got good feedback from players and the real gaming situation spawned more ideas which could be added to the application and points counting system. Further development of the application will continue alongside with the points counting system.

Keywords: Java, serial communication, wireless transfer.

SISÄLLYSLUETTELO

KÄYTETYT MERKIT JA LYHENTEET

JTAG	Joint Test Action Group, Mikrokontrollerin ohjelmointi ja testausväylä.
GUI Builder	Graphical user interface builder, graafisen käyttöliittymän luontiin tarkoitettu ohjelmointityökalu.
API	Application programming interface, ohjelmointirajapinta, jonka avulla ohjelmat voivat tehdä pyyntöjä sekä vaihtaa tietoja.

1. JOHDANTO

Tämä työ on osa Meri-Lapin Paintball -vuokraamo ja laitesuunnittelu -yrityksen kehittämää pisteidenlaskujärjestelmää, joka on kehitetty helpottamaan paintball-pelien pisteidenlaskua. Työssä käsitellään tietokonesovelluksen luontia pisteidenlaskujärjestelmälle, jonka avulla järjestelmää voidaan ohjata tietokoneelta. Sovellus toimii pelaajille tiedonlähteenä pelin kulusta.

Alussa päätettiin, että ohjelma luotaisiin Javalla ja paras työkalu tähän tarkoitukseen oli NetBeans-ohjelmointiympäristö. Työtä aloitettaessa selvisi myös, että Javan kanssa pitää käyttää erillistä Java Communications API -laajennusta, jonka avulla sarjaliikenteen ohjaaminen toimii. Sarjaliikennettä tarvitaan tietoliikenteen kanssa. Alustavina vaatimuksina ohjelmaan olivat tietoliikenteen luonti laitteiden ja tietokoneen välille, kokonaisajan näyttö ruudulla, kellojen käynnistämisen ja lopetuksen ohjaaminen, lokitiedostojen luonti pelin kulusta, kartan lisääminen ohjelmaan pelialueesta ja graafinen esitys pisteistä. Näistä jokainen osio tuli tehdyksi ohjelmaan ja ohjelman kehittyessä siihen lisättiin vielä monia muita ominaisuuksia.

Sovelluksen luonti alkoi tietoliikenteen luonnista pisteidenlaskujärjestelmän ja sovelluksen välille. Tämän jälkeen muiden osa-alueiden luonti oli mahdollista. Näiden osa-alueiden luonti tapahtui siinä järjestyksessä, kun niitä oli mahdollista lisätä ohjelmaan. Työn ohella myös pisteidenlaskujärjestelmän ohjelmointi tapahtui samaan aikaan, joten jotkin asiat olivat lähes mahdottomia toteuttaa, ennen kuin pisteidenlaskujärjestelmän vastaavat ominaisuudet valmistuivat. Tämä kuitenkin helpotti jonkin verran työn suoritusta, kun pisteidenlaskujärjestelmän ja tietokonesovelluksen ohjelmointi tapahtui samaan aikaan.

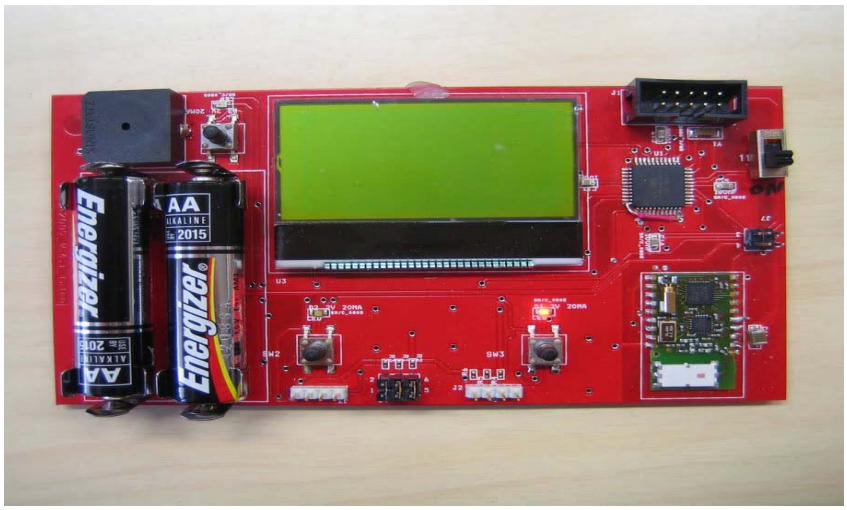
Java oli entuudestaan tuttu kieli, mutta ohjelmoinnin aikana oppi lisää uusia asioita Javasta. NetBeans-ohjelmointiympäristö tuli uutena asiana, mutta sen käyttö on sen verran helppoa, että sen opetteluun ei mene juuri ollenkaan aikaa. Työn ohella C-kielen ja sulautetun ohjelmoinnin taidotkin kehittyivät, koska pisteidenlaskujärjestelmän ohjelmointia piti suorittaa samanaikaisesti.

2. PISTEIDENLASKUJÄRJESTELMÄ

Pisteidenlaskujärjestelmä on Meri-Lapin Paintball -vuokraamo ja laitesuunnittelu -yrityksen kehittämä tuote, joka on tehty helpottamaan paintball-pelien pisteidenlaskua. Järjestelmään kuuluu tietty määrä kelloja, jotka asetetaan pelialueelle yleensä bunkkereihin. Kellojen lisäksi käytössä on tukiasema, joka asetetaan mielellään mahdollisimman keskelle pelikenttää, jotta kuuluvuus kaikille kelloille pysyisi hyvänä. Järjestelmä soveltuu paintballin lisäksi muillekin samantyyppisille lajeille, kuten esimerkiksi airsoftiin.

2.1. Pelikello

Pelikello on sulautettu laite, josta löytyy LCD-näyttö, piezo, kolme näppäintä, 3 lediä, radiopiiri sekä järjestelmää ohjaava Atmel Atmega644 -mikroprosessori. Pelikellon tarkoitus on toimia valloituspisteenä, jonka joukkue voi valloittaa kerätäkseen pisteitä pelissä. Tämä tapahtuu, kun joukkueen pelaaja valloittaa pisteen painamalla oman joukkueensa värin väristä näppäintä pelikellolta. Tämän jälkeen pelikello sytyttää valloittaneen joukkueen värisen ledin ja piezo antaa äänimerkin valtauksesta. Pelikello aloittaa valtauksesta ajan laskennan kyseiselle joukkueelle ja ajan eteneminen on myös näkyvillä LCD-näytöllä. Valloituksesta lähetetään ilmoitus tukiasemalle, joka suorittaa lopullisen pisteidenlaskun. Pelikello lähettää ainoastaan tietoja tukiasemalle, joten se ei ole ollenkaan yhteydessä toisiin kelloihin tai tietokonesovellukseen. Pelikellot ovat erotettavissa toisistaan niille annetuilla ID- numeroilla. Kuvassa 1 on pisteidenlaskujärjestelmän pelikellon prototyyppi, jota on käytetty tätä sovellusta tehtäessä.

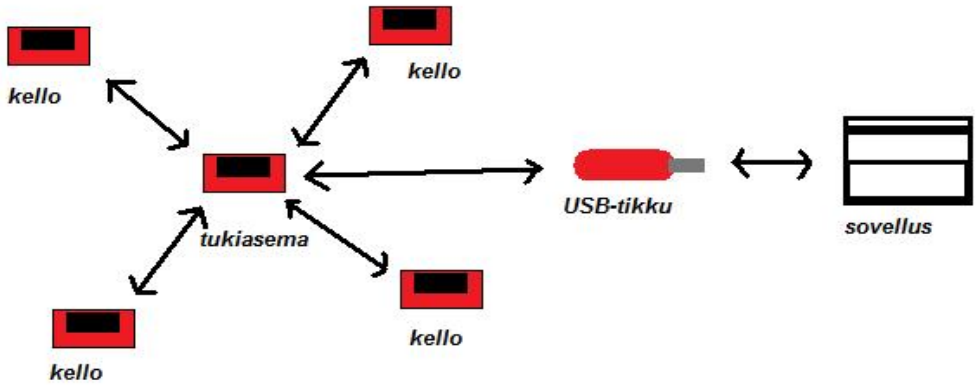


Kuva 1. Pisteidenlaskujärjestelmän pelikello

2.2. Tukiasema

Tukiasema on ainakin toistaiseksi rakenteeltaan täysin samanlainen kuin pelikello, mutta sen mikroprosessori on ohjelmoitu tekemään eri asioita. Tukiasema ei ole pelaajien ulottu-

villa ja sen tarkoitus on antaa komennot pelikelloille sekä käsitellä niiltä tuleva informaatio. Tukiasema myös lähettää tiedot eteenpäin USB-väylään asetettavalle tikulle, jolta tietokonesovellus lukee tiedot. Tukiasema etsii aluksi jokaisen kellon, jonka se pelialueelta löytää ja tämän jälkeen pelin voi aloittaa. Tukiasema voi antaa myös kelloille keskeytys- ja lopetuskäskyt. Pelkällä tukiasemalla pelin ylläpito on hyvin karua, joten tietokonesovellus tuleekin tässä avuksi. Tukiasemaan on ohjelmoitu mahdollisuus käyttää erilaisia peliasetuksia, mutta ne ovat tällä hetkellä vain säädettävissä tietokonesovelluksen kautta. Kuvassa 2 on nähtävillä, kuinka tukiasema toimii järjestelmän tietoliikenteen keskipisteenä.



Kuva 2. Järjestelmän tietoliikenne

2.3. Pelin kulku

Kun kaikki tarvittavat kellot ovat ilmoittautuneet tukiasemalle, peli voidaan aloittaa. Pelin käynnistyskäsky annetaan samaan aikaan, kun pelaajille annetaan lupa aloittaa peli. Pelin käynnistyttyä pelikellot vaihtavat tilaa, jolloin ne ovat vallattavissa. Kun pelaajat valtaavat pelin edetessä pelikelloja, he saavat jokaisesta kymmenestä sekunnista yhden pisteen, jonka he ovat kyseistä pelikelloa pitäneet hallussaan. On myös mahdollista saada tietty määrä pisteitä jokaisesta valtauksesta, jonka joukkue suorittaa, jos tämä asetus on asetettu tietokonesovelluksesta. Keskeytys- tai lopetuskäskyn voi antaa milloin vain pelin kuluessa, silloin kun tarve sitä vaati. Ohjelmistolta on säädettävissä myös, että pelin lopetus tapahtuu esimerkiksi toisen joukkueen saavuttaessa tietyn määrän pisteitä tai aikarajan jälkeen. Pelin voitto määräytyy luonnollisesti sille joukkueelle, joka on saanut eniten pisteitä pelin lopetuskäskyn tultua. Tämä on vain yksi kaikista mahdollisista pelimuodoista, joita tällä järjestelmällä voidaan toteuttaa. Tulevaisuudessa järjestelmässä tulee olemaan useampi pelimoodi, joka on valittavissa ennen pelin aloitusta.

2.4. Pisteidenlaskujärjestelmän ohjelmointi

Osallistuin myös pisteidenlaskujärjestelmän ohjelmointiin, jonka ohjelmoinnista vastasin pääosin. Laitteen mikroprosessorin ohjelmointi tapahtuu JTAG-liitännän kautta. Laitteelle

syötettävä lähdekoodi kirjoitettiin C-kielellä käyttäen Atmelin omaa AVR Studio -ohjelmointiympäristöä. JTAG-liitännän kautta ohjelmointi mahdollistaa myös debuggauksen, joka helpottaa huomattavasti virheiden etsimistä koodista. Ohjelmointi koostuu suurilta osin rekistereiden käsittelystä.

Järjestelmän kaksi isointa osa-aluetta olivat radioliikenteen sekä näytön toiminnan toteutus. Kun näiden perustoiminnot saatiin toimimaan, pisteidenlaskujärjestelmän ja tietokonesovelluksen osioiden toteuttaminen tapahtui samanaikaisesti. Näistä järjestelmä oli se, jonka ohjelmointi vei enemmän aikaa ja oli vaikeampi osio saada toimimaan oikein. Pisteidenlaskujärjestelmän toteutuksessa täytyi pelikelloille sekä tukiasemalle tehdä oma ohjelmointinsa. Isoa osaa lähdekoodista pystyi kuitenkin käyttämään sekä pelikellossa että tukiasemassa, sillä niissä on hyvin monta samaa ominaisuutta. Esimerkiksi näytön ohjaus sekä radioliikenteen käyttö on laaja osa kummankin toimintaa. Näistä tukiaseman koodi on laajempi, sillä se joutuu käsittelemään paljon tietoja, joita sille lähetetään pelikelloilta tai tietokonesovellukselta. Pelikellon toiminta pelin aikana perustuu siihen, että se laskee aikaa, lähettää tietoja valtauksista tukiasemalle sekä vastaanottaa mahdolliset pelin tilanvaihdokset.

Pelikellossa sekä tukiasemassa pyörivät tilakoneet määrittävät, mitä järjestelmässä milloinkin tapahtuu. Tila tukiasemalla vaihtuu käyttäjän toimesta. Tukiasema lähettää tämän jälkeen käskyn kaikille kelloille siirtyä samaan tilaan. Kun tilaan siirrytään, niin osa sen sisällä olevista funktioista ajetaan vain kerran ja osaa ajetaan niin kauan kun kyseisessä tilassa ollaan. Nämä ikuisen toistolauseen sisällä sijaitsevat funktiot ovat pääasiassa näytön päivitystä varten.

Tilakoneen runko näyttää seuraavalta.

```
while(1)
{
…..
    switch(state)
    {
        case waiting:
        …..
        case game_runs:
        …..
        case game_paused:
        …..
        case game_end:
         …..
    }
}
```

Tapahtumat, kuten radioliikenne ja näppäinten painaminen, eivät tapahdu tilakoneessa, vaan ne toimivat keskeytyksillä. Esimerkiksi näppäinpainalluksilla voidaan vaihtaa tilakoneen tilaa. Kun käyttäjä painaa näppäintä, niin mikroprosessorilla tapahtuu keskeytys. Keskeytyksen avulla voidaan ajaa haluttu funktio, joka tässä tapauksessa vaihtaisi tilakoneen tilaa. Sama periaate pätee myös radioliikenteessä. Mikroprosessorilla tapahtuu keskeytys,

kun dataa on saatavilla radiopiiriltä. Tämän keskeytyksen keskeytysrutiini käsittelee piiriltä saadun datan. Tämä data on viesti toiselta laitteelta, jonka avulla voidaan vaihtaa vastaanottajan tilakoneen tilaa. Järjestelmässä olevat laskurit, joita on käytetty ajanlaskun toteutuksessa, on myös toteutettu keskeytyksillä. Tämä keskeytys on asetettu tapahtumaan sekunnin välein, jonka avulla tarkkojen sekuntien laskeminen on mahdollista.

Keskeytykset on aktivoitava koodin alussa rekisterien avulla ja tämä tapahtuu seuraavasti.

```
PCICR=0x0A;   //aktivoi B ja D portissa olevat keskeytykset (00001010)
PCMSK1=0x0C;  //aktivoi keskeytykset B portin pinnissä 3 ja  2 (00001100) (näppäimet)
PCMSK3=0x10;  //aktivoi keskeytykset D portin pinnissä 5 (00010000) (dataa saatavilla)
```

Radioliikenteen laukaisema keskeytys, kun dataa on saatavilla, näyttää seuraavalta.

```
ISR(PCINT3_vect)
{
    incomingdata();  //datan käsittely
}
```

3. PAINTBALL

Paintball on vauhdikas joukkuepeli, jossa on tarkoitus eliminoida vihollisen pelaajia värikuulia ampuvalla merkkaimella. Pelissä on myös yleensä pelimuodosta riippuen jokin tavoite, joka joukkueiden pitää yrittää saavuttaa.

3.1. Historia

Paintball on lajina hyvin nuori, sillä ensimmäinen peli on pelattu vasta vuonna 1981. Lajin idea lähti muutama vuosi aikaisemmin Floridassa kolmen henkilön keskustelusta. Ajatuksena oli, miten saataisiin yhdistettyä vaaratilanteessa ihmisen tuntema adrenaliinin ja terävöityneiden aistien aiheuttama jännityksen ja pelon tunne vaarattomaan ympäristöön niin, että sen voisi kokea aina uudestaan. Välineeksi alussa päätyi kuula-ase, joka oli tarkoitettu erilaisten kohteiden värillä merkitsemiseen, kuten kaatuneiden puiden merkitseminen hankalassa maastossa sekä riistan ja karjan laskennan helpottaminen. Lajin liikkeellelähtö oli hidas ja markkinoinnissa ongelmia, sillä kukaan ei halunnut investoida tuntemattomaan lajiin, jonka varusteet olivat suhteellisen kalliita. Ongelma kuitenkin ratkesi vuokraustoiminnalla, jonka jälkeen lajin kasvu oli Yhdysvalloissa huimaa. Muutamassa vuodessa Yhdysvaltoihin perustettiin 400 kenttää. /2/

3.2. Paintball nykyisin

Jokaisella pelaajalla pelissä on merkkain, jolla hänen pitää eliminoida vihollisen pelaajia pois pelistä. Merkkaimilla ammutaan värikuulia, jotka hajoavat osuttaessa. Merkkaimet ovat nykyään joko pumpputoimisia tai puoliautomaattisia. Näistä yleisempi on puoliautomaattimerkkain. Voimanlähteenä merkkaimissa on kaasupullo, joka sisältää joko paineilmaa tai hiilidioksidia (CO2). Kuvassa 3 on nähtävillä perusmallinen puoliautomaatti merkkainmallia Tippman 98 Custom. Jos pelaaja saa osuman mihin tahansa kohtaa vartaloa, hän eliminoituu pelistä. Kuulan sisällä oleva aine on vesiliukoista ja helposti pesussa lähtevää elintarvikevärillä värjättyä kasvisrasvaa. Lajissa käytetään myös suojavarusteita, joista pakollinen on maski, joka suojaa silmät, korvat ja suun. Peliä on mahdollista pelata joko ulkona rajatulla alueella tai halleihin tehdyillä sisäkentillä. Lajissa on monia eri pelimuotoja, joista turnauksissa yleisin on lipunryöstöpelin eri variaatiot. Tässä opinnäytetyössä kyseessä oleva pistelaskujärjestelmä on kumminkin suunnattu kellopeli- eli aluehallintatyyppiseen pelimuotoon. Suomessa lajia pelataan SM-tasolla sekä erilaisia turnauksia ja liigoja järjestetään useampia vuodessa. /3/

Kuva 3. Tippmann 98 Custom -merkkain

4. JAVA JA SEN SARJALIIKENNE

Java on laitteistoriippumaton oliopohjainen ohjelmointikieli, jonka on kehittänyt Sun Microsystems. Muista kielistä poiketen Java käännetään tavukoodiksi, joka sen jälkeen

ajetaan virtuaalikoneessa. Tämän takia Java-ohjelmat eivät pääse vaikuttamaan muihin prosesseihin, joten ohjelmat ovat tavanomaisia konekieliohjelmia turvallisempia. Tosin tämä ominaisuus myös tekee Java-ohjelmista hieman hitaampia kuin muut ohjelmat. Javassa on myös haettu helppokäyttöisyyttä, joka esimerkiksi näkyy muistinhallinnassa, jossa on käytössä roskienkeräin, joka vapauttaa muistia sitä mukaa kun sitä ei tarvitse. Javassa on myös runsaasti ominaisuuksia, jotka ovat toisissa kielissä yleensä käyttöjärjestelmäriippuvaisia tai kolmansien osapuolten kirjastojen varassa. Näistä esimerkkinä ovat graafinen käyttöliittymäkirjasto ja verkko-ominaisuudet. /5/

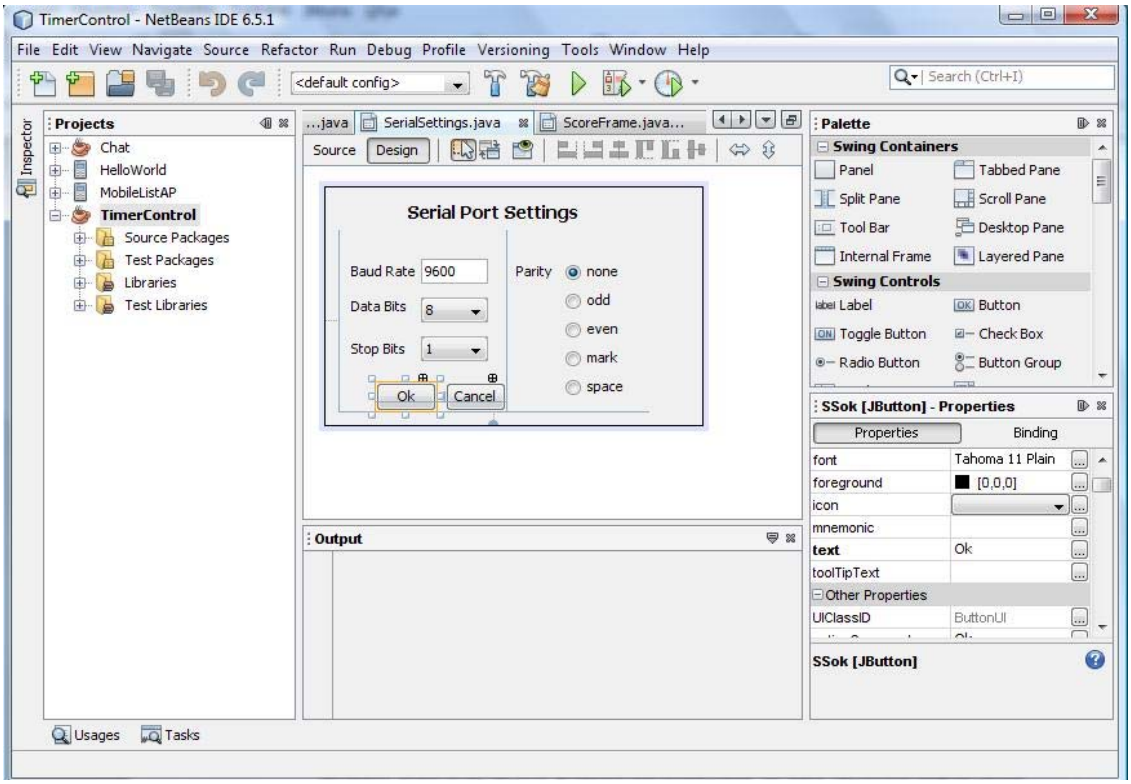
Koska sovelluksen kanssa käytettävän USB-tikun kanssa kommunikoidaan sarjaportin avulla, on sarjaliikenteen käsittely oltava mahdollista. Javasta ei löydy oletuksena sarjaportin hallintaa, joten tätä varten tarvitaan erillinen laajennus, jolla sarjaporttiliikenteen käsittely onnistuu. Käytettävä laajennus on nimeltään Java Communications API, joka myös tunnetaan nimellä javax.comm. Tämä laajennus antaa mahdollisuudet muun muassa säätää porttien asetuksia, niiden eri signaaleja ja lähettää sekä vastaanottaa dataa. Tarvittavien tiedostojen lisäksi paketissa on mukana kattava määrä esimerkkejä, jotka ohjeistavat, kuinka sarjaliikenteen ohjaus onnistuu Javassa.

5. NETBEANS

NetBeans on kattava avoimeen lähdekoodiin perustuva ohjelmointiympäristö. Se on kirjoitettu Javalla ja se on alunperin ollut Java-ohjelmointiympäristö, mutta ajan saatossa se on

laajentunut ja se mahdollistaa ohjelmoinnin myös C, C++, JavaScript sekä monella muulla kielellä.

NetBeans sai alkunsa vuonna 1996 opiskelijaprojektina, jolloin se tunnettiin nimellä Xelfi ja sen ensimmäiset julkaisut tulivat vuonna 1997. Se oli ensimmäinen Javalla kirjoitettu ohjelmointiympäristö. Samana vuonna sen ympärille perustettiin yritys, kunnes vuonna 1999 Sun Microsystems osti kyseisen ohjelmistoympäristön. Vuonna 2000 Sun päätti, että NetBeans vaihtuu avoimeksi lähdekoodiksi. NetBeansin kehitys jatkuu edelleen ja sen käyttäjämäärät ovat koko ajan kasvussa. /1/



Kuva 4. NetBeans IDE

NetBeans soveltui tähän työhön todella hyvin sen helppokäyttöisen Swing GUI -builderin ansiosta. Tämä antaa mahdollisuudet helposti luoda ohjelmalle graafinen käyttöliittymä pelkästään vetämällä tarvittavia käyttöliittymäkomponentteja haluamaan ikkunaan. Käyttöliittymäkomponenttien asetuksia voi asettelun jälkeen säätää mieleisekseen valikosta. Esimerkiksi näppäinkomponentille voidaan antaa funktio, jonka se ajaa, kun näppäintä on painettu. Ohjelma luo tarvittavat koodit kyseisille komponenteille, jotka ainakin ohjelman oletusasetuksilla ovat vielä lukittu niin, että käyttäjä ei pysty muokkaamaan näitä valmiiksi luotuja koodirivejä. Näin käyttäjä ei voi ainakaan vahingossa hajottaa komponentteja poistamalla tai muokkaamalla niille tärkeitä koodirivejä. Jos komponentin taas haluaa poistaa, se tapahtuu poistamalla kyseinen komponentti joko graafisesta näkymästä tai projektin komponenttilistasta. Ohjelma myös automaattisesti poistaa sille tarkoitetut koodit projektista. Ilman tämän kaltaista systeemiä graafisten käyttöliittymien luonti kestäisi tuhottoman kauan varsinkin, jos kyseessä on vähänkin laajempi projekti.

NetBeans on kuitenkin suhteellisen raskas ohjelmisto, joten pienemmät projektit, jotka eivät tarvitse graafista käyttöliittymää on ehkä suositeltavampaa luoda jollain toisella ohjelmistolla. Kuvassa 4 on nähtävillä NetBeansin näkymä, jossa graafisen käyttöliittymän luonti tapahtuu. Itse sovelluksen näkymät ovat nähtävillä keskellä. Tarvittavat komponentit ovat valittavissa oikealla puolella olevasta laatikosta. Tämän laatikon alapuolelta löytyy valitun komponentin asetusvalikko, josta voidaan helposti säätää komponentille tarvittavat asetukset.

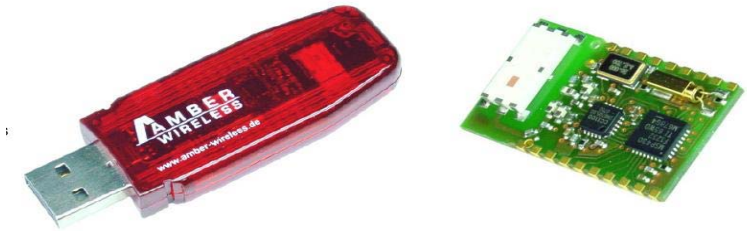
6. OHJELMAN TOTEUTUS

Ohjelma on toteutettu NetBeans ympäristön GUI esimerkin pohjalle. Tämän ympärille on helppo rakentaa yksinkertainen ohjelma, joka tarvitsee pääasiassa vain peruskomponentteja, kuten näppäimiä, tekstiruutuja sekä valikoita. Myös tämän ansiosta ohjelman lähes koko graafisen puolen luonti ei vaatinut yhtään itsekirjoitettua koodiriviä. Jokaisen tarvittavan komponentin voi valita komponenttivalikosta drag and drop -tyyliin. Käsin ohjelmoitavaksi jäi luoda tapahtumat luoduille komponenteille sekä tietoliikenne ja tietojen käsittely.

Ohjelman päänäkymä jakautuu kahteen osaan. Toisessa on sarjaliikenteeseen liittyvät valikot sekä tila teksti, joka ilmoittaa missä tilassa peli on milläkin hetkellä. Toisessa taas on itse peliin liittyvät asiat, jotka ovat eri välilehtien sisällä. Välilehtiä käyttämällä kaikkia sovelluksen ominaisuuksia ei tarvinnut yrittää saada mahtumaan yhteen näkymään ja välilehden vaihtaminen käy kätevästi milloin tahansa. Jokainen näistä välilehdistä sisältää yhden ohjelman osa-alueista. Nämä kaikki osa-alueet käsitellään tämä kappaleen sisällä.

6.1. Tietoliikenne

Pisteidenlaskujärjestelmän ja ohjelmiston välinen tietoliikenne tapahtuu langattomasti radiopiirien avulla. Pisteidenlaskujärjestelmän radiopiiri lähettää tiedot USB-väylään asetettavalle tikulle, joka sisältää samanlaisen radiopiirin. USB-tikku sekä radiopiiri on nähtävillä kuvassa 5. Tikun ja tietokoneen välinen kommunikaatio tapahtuu tikun valmistajan kotisivuilta ladattavilla ajureilla. Nämä ajurit laittavat USB-tikun näkymään sarjaportissa ja tämän jälkeen tikun ja ohjelmiston välinen tietoliikenne onnistuu sarjaliikenteen avulla.



Kuva 5. USB-tikku ja radiopiiri /4/

Ennen varsinaisen käyttöliittymän tekemistä sarjaliikenne testattiin pienellä ohjelmalla, jossa oli otettu mallia Java Communications API:n mukana tulleista esimerkeistä. Tällä ohjelmalla testattiin yksinkertainen viestin vastaanotto sekä lähetys. Ensimmäiseksi ohjelma tarkistaa, onko koodissa annettu portti olemassa, ennen kuin siihen yritetään edes yhdistää. Tässä vaiheessa portti annettiin vielä vain muuttujana koodin sisällä eikä siihen pystytty ohjelmaa ajettaessa vaikuttamaan. Ohjelma ottaa listaan talteen kaikkien käytössä olevien sarjaporttien tunnisteet ja näiden tunnisteiden avulla jokaista listassa olevaa portin nimeä verrataan annetun portin kanssa. Jos listassa olevan porttitunnisteen nimi täsmää muuttujana annetun portin nimen kanssa, laite on tällöin löytynyt halutusta portista. Kyseinen porttitunniste annetaan eteenpäin ja tämän porttitunnisteen avulla porttiin lopulta yhdistetään.

```
String defaultPort = "COM6"; //portti johon halutaan yrittää yhdistää
portList = CommPortIdentifier.getPortIdentifiers(); //kaikki porttitunnisteet listataan
while (portList.hasMoreElements()) //käydään jokainen listassa oleva läpi
{
    portId = (CommPortIdentifier) portList.nextElement(); //seuraava listan elementti
    if (portId.getPortType() == CommPortIdentifier.PORT_SERIAL)//vain sarjaportit
    {
        if (portId.getName().equals(defaultPort)) //nimen vertaus
        {
        ....  //yhdistäminen yms. jos nimeen täsmäävä portti on löytynyt
        }
```

```
    }
}
```

Yhdistäminen tapahtuu seuraavalla tavalla.

```
Connection=true;
try
{
    serialPort = (SerialPort) portId.open("TimerControl", 2000); //yhdistäminen porttiin
}
catch (PortInUseException e) // jos portti on jo käytössä eikä siihen voida yhdistää
{
     System.out.println("Port in use!");
    Connection=false;
}
```

Yhdistämisen jälkeen määritetään yhteyden eri asetukset sekä luodaan input ja output streamit. Yhteyteen luodaan kuuntelija, joka ilmoittaa aina, kun kyseisessä sarjaliikenteessä tapahtuu jokin tapahtuma. Ilmoitettavat tapahtumat voidaan määrittää, joten tapahtumia joita ei tulla tarvitsemaan, voidaan jättää aktivoimatta. Näistä tapahtumista tässä tapauksessa on käytössä ainoastaan ilmoitus, kun dataa on saatavilla.

```
 try
{
    serialPort.addEventListener(this);  //lisää sarjaporttitapahtumien kuuntelijan
}
catch (TooManyListenersException e) {}
serialPort.notifyOnDataAvailable(true);   //aktivoi dataa saatavilla ilmoitukset
```

Kun sarjaliikenteessä tapahtuu jotakin, niin serialEvent() funktio käsittelee sen.

```
public void serialEvent(SerialPortEvent event)
{
     switch (event.getEventType())
     {
     case SerialPortEvent.DATA_AVAILABLE:   //data saatavilla tapahtuma ja sen käsittely
     …...
     }
}
```

Sarjaportin parametrit asetetaan yhdistämisen jälkeen. Parametreihin kuuluvat siirtonopeus(baud rate), data bitit(data bits), pysäytys bitit(stop bits) sekä pariteetti(parity). Databitit määrittävät, kuinka monta databittiä lähetetään aloitusbitin jälkeen. Databitit ovat säädettävissä 5-8 väliltä, mutta lähes kaikki laitteet käyttävät arvoja 7 tai 8. Lopetusbitti lähetetään datan jälkeen ja se voi olla pituudeltaan 1, 1.5 tai 2. Pariteettibitti antaa pienen virheentarkistuksen mahdollisuuden, jos data on korruptoitunut. Pariteettibitin eri vaihtoehtoja ovat pariton(odd), parillinen(even), merkki(mark) ja väli(space) tai ei mikään(none). USB-tikun manuaali antaa ohjeet asettaa siirtonopeuden arvoksi 9600, 8 databittiä, 1 pysäytysbitti ja ei pariteettibittiä.

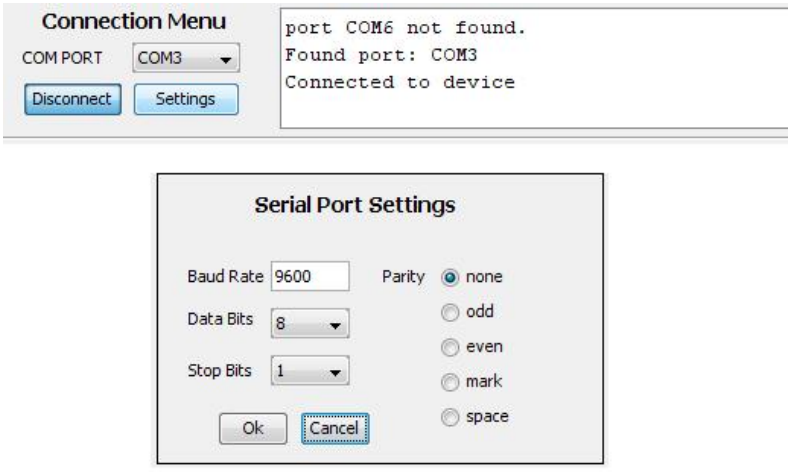
```
serialPort.setSerialPortParams(baudrate, databits, stopbits, parity);
```

Manuaali käskee asettamaan DTR-signaalin matalaksi. Ilman kyseistä asetusta tietoliikenne ei tule toimimaan kunnolla tikun ja ohjelman välillä.

```
if(serialPort.isDTR()) //jos DTR on päällä niin se asetetaan pois
{
    serialPort.setDTR(false);
}
```

Asetusten jälkeen ohjelma pyörii odottaen milloin dataa on saatavilla. Kun tikulle saapuu dataa, ohjelman kuuntelija antaa huomautuksen, että dataa on saatavilla. Tämän jälkeen input streamistä voidaan lukea data tavuina, jotka muutetaan char-muotoon ja lisätään merkkijonoon. Tämä merkkijono tulostetaan ja tämän jälkeen ohjelma lähettää oman viestinsä takaisin tikulle, joka lähettää sen eteenpäin pisteidenlaskujärjestelmälle. Tämä viesti pystyttiin lukemaan järjestelmän näytöltä, kun käytössä oli tähän sopiva testikoodi. Lähetettävä viesti, joka on merkkijonomuodossa, pitää myös muuntaa Javan omalla funktiolla tavumuotoon ennen sen lähettämistä.

Tietoliikenteen siirtäminen graafiseen ohjelmaan tapahtuu helposti siirtämällä testissä ollut sarjaliikenneluokka mukaan ohjelmaan. Kun luokka luodaan, sille annetaan parametreinä tarvittavat asetukset, jotka tullaan asettamaan luokassa. Nämä parametrit käyttäjä voi muuttaa ohjelmasta valikkojen avulla. Portti, mihin halutaan yhdistää, on valittavissa tiputusvalikosta. Mahdollisia sarjaportin asetusten muutoksia varten on luotu säätövara. Sarjaporttiasetukset ovat muokattavissa erillisestä ikkunaan aukeavasta valikosta, joka on nähtävillä kuvassa 6. Testissä käytettyyn sarjaliikenneluokkaan on lisättävä yhteyden sulkeminen. Yhteyden sulkevassa funktiossa streamit suljetaan sekä sarjaporttitapahtumien kuuntelija suljetaan ennen varsinaisen portin sulkemista. Yhteyden sulkeminen tapahtuu samasta näppäimestä kuin laitteeseen yhdistäminen.



Kuva 6. Sarjaliikenteen valikot

Sarjaliikennettä testatessa oli apuna erillinen terminaaliohjelma, jolla pystyi ottamaan yhteyden sarjaporttiin. Tämän ohjelman avulla pystyi näppärästi esimerkiksi määrittämään, onko virhe Java-sovelluksessa vai pisteidenlaskujärjestelmässä. Jos viestit näkyivät oikein

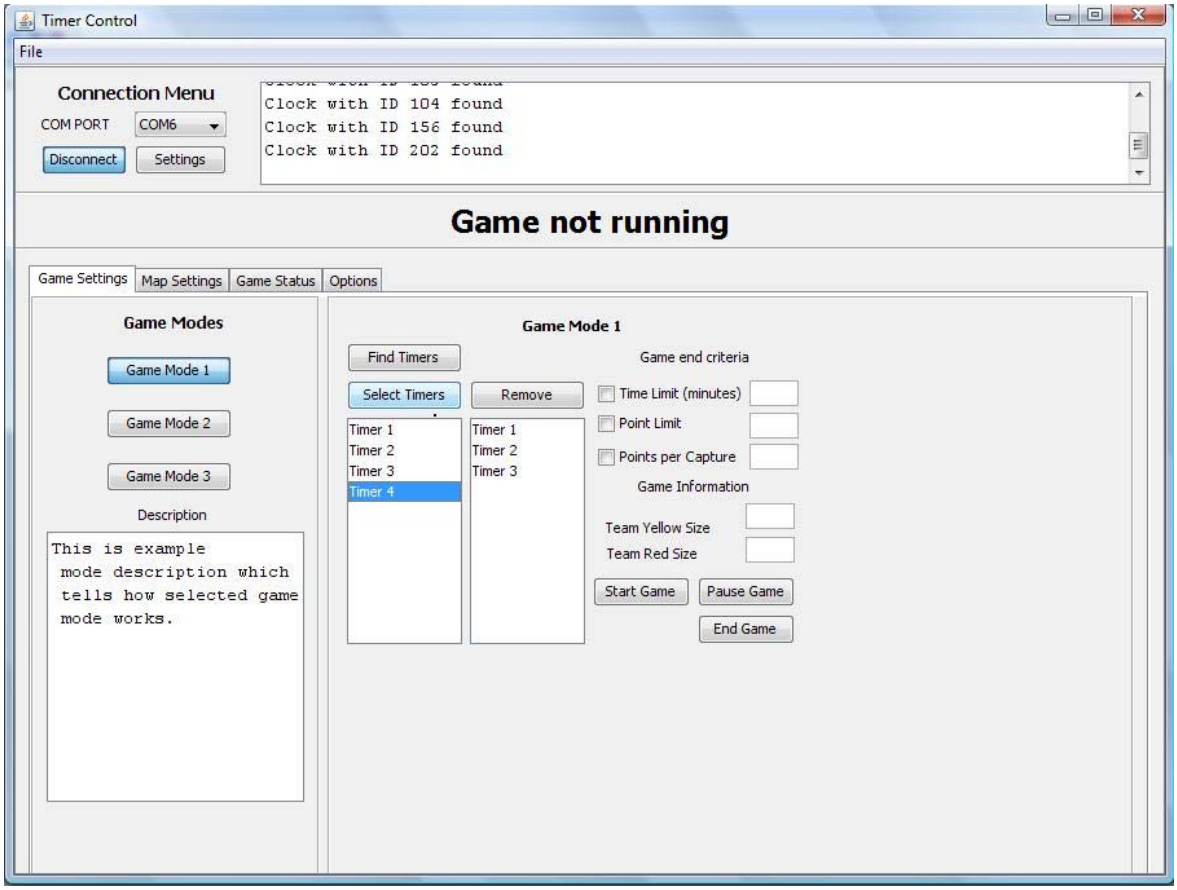
tässä terminaaliohjelmassa, mutta Java-sovellus ei käsitellyt niitä oikein, oli selvää, että virhe täytyi olla Java-sovelluksessa. Terminaaliohjelmassa jo väärin näkyvistä viesteistä pystyi havaitsemaan, että vika on todennäköisesti järjestelmän päässä. Myös terminaaliohjelmalla pystyi lähettämään viestejä tikulle, joka auttoi järjestelmän testauksessa.

6.2. Pelin ohjaus

Peliä voidaan ohjata ja antaa sille erilaisia asetuksia asetusikkunasta. Asetukset ja ohjaaminen ovat riippuvaisia siitä, mitä pelimoodia pelissä tullaan käyttämään. Pelimoodeista on tässä versiossa vain yksi valmiina, joten muiden pelimoodien lisääminen on jäänyt myöhempiin päivityksiin. Kaikille pelimoodeille yhteistä tulee luultavasti olemaan ohjaus, jolla peli voidaan aloittaa, lopettaa sekä pysäyttää.

Aluevaltaus eli kellopeli on pisteidenlaskujärjestelmän oletuspelimoodi. Tämä on myös ohjelman ainut tällä hetkellä käytettävissä oleva pelimoodi. Pelimoodissa joukkueet yrittävät valloittaa kentällä olevia pelikelloja sekä pitää niitä hallussaan mahdollisimman kauan. Pisteytys tapahtuu niin, että joukkue saa pisteen jokaisesta kymmenestä sekunnista, mitä se on pitänyt hallussaan yhtä pelikelloa. Pisteet jokaiselta pelikellolta yhdistetään. Se joukkue, jolla on eniten pisteitä, kun peli lopetetaan, on voittaja. Tässä pelimoodissa säädettävyys on piste- sekä aikarajoissa. Oletusasetuksilla peli ei lopu käynnistämisen jälkeen itsestään, vaan se on sammutettava manuaalisesti. Ohjelmassa säädettävillä asetuksilla voidaan peli määrätä loppumaan, kunnes toinen joukkue saavuttaa tietyn pistemäärän tai peliä on pelattu tietty aikamäärä. On myös mahdollista käyttää kumpaakin asetusta, joten peli loppuu, kun toinen näistä on saavutettu. Sen lisäksi, että tarvittava aikaraja tai pisteraja on kirjoitettu tekstiruutuun, on myös rastitettava valintaruutu. Nämä ovat nähtävissä kuvassa 7. Ikkunassa on myös mahdollista ilmoittaa tietoja, jotka eivät vaikuta itse peliin, mutta näkyvät tapahtumalokissa. Nämä tiedot ovat tällä hetkellä jääneet vain pelaajamäärien ilmoitukseen. Asetukset eivät ole enää muutettavissa sen jälkeen, kun peli on aloitettu.

Sovelluksen luonnin myöhäisessä vaiheessa mukaan haluttiin lisäasetus, joka päälle asetettuna antaisi joukkueille lisäpisteitä jokaisesta valtauksesta. Tämä lisäpistemäärä on käyttäjän mahdollista määrittää samalla tavalla kuin aikaraja tai pisteraja. Jos joukkue valtaa esimerkiksi toiselta joukkueelta kellon, niin valtaava joukkue saa asetetun lisäpistemäärän verran lisäpisteitä heidän kokonaispisteisiinsä. Asetus oli helppo toteuttaa, kun vain lisäsi kummallekin joukkueelle muuttujan, joka laskee kummankin joukkueen tekemät valtaukset. Tämä määrä kerrotaan luvulla, joka on annettu asetuksiin pelin alussa.



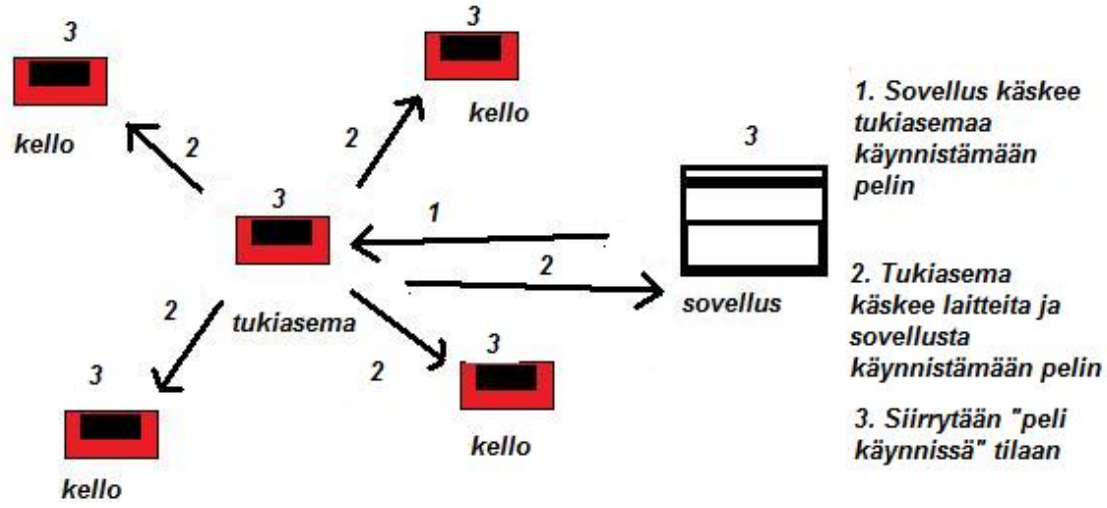
Kuva 7. Kellopelimoodin ohjausikkuna

Ennen pelin aloitusta ja säätöjä on ohjelman tiedettävä ensin, mitkä kellot tulevat olemaan pelissä mukana. Tämä tapahtuu niin, että ohjelma käskee tukiaseman etsiä jokainen alueella oleva pelikello. Tukiasema lähettää viestit alueella oleville pelikelloille, jossa se käskee niiden ilmoittautua tukiasemalle. Pelikellot lähettävät ilmoittautumisviestit tukiasemalle, joka ottaa jokaisen kellon ID-numeron talteen. Kun tukiasema on käynyt haun läpi, se lähettää löydetyt kellot ohjelmalle, jossa ne ilmestyvät listaan. Tästä listasta voidaan määrittää, mitkä kellot otetaan mukaan peliin. Yleisimmin tullaan kuitenkin käyttämään kaikkia löydettyjä kelloja, mutta kellojen valikointi tehtiin varmuuden vuoksi, jos sille tarvetta sattuu joskus olemaan. Kuten kuvassa 7 nähdään, kuinka tukiasema on löytänyt neljä eri kelloa, mutta yksi on jätetty valitsematta. Kun pelikellot ovat valittuina, ne ilmestyvät myös karttaikkunan pelikellolistaan, mistä ne voidaan asettaa kartalle jo ennen pelin aloitusta.

Kun ilmoitus kellosta tulee ohjelmalle, se lisätään aluksi listaan seuraavasti.

```
public void addtimer(int timer) //timer  = kellon ID numero
{
    if(!timers.contains(timer))
    {
        timers.add(timer);
        GM1.addTimer("Timer "+(timers.size()));
    }
}
```

Pelin ohjaus tapahtuu lähettämällä komentoja sarjaportille, joka ohjaa ne tikun kautta tukiasemalle. Kun tukiasema saa viestin, se käsittelee tapahtuman ja lähettää tarvittavat viestit kelloille sekä ohjelmalle. Jos ohjelmasta esimerkiksi annetaan pelin aloituskomento, itse ohjelmassa ei tapahdu mitään, ennen kuin tukiasema on lähettänyt aloitusviestin ohjelmalle takaisin. Tällä tavoin peli ei lähde turhaan päälle, jos yhteyttä tukiasemaan ei enää saada. Kuvassa 8 on nähtävillä, miten viestiliikenne tapahtuu, jos sovelluksesta annetaan aloituskomento. Jos peli käynnistetään tukiasemalta, peli lähtee käyntiin myös sovelluksessa. Tällöin aika ja pisteasetusten asettaminen ei ole mahdollista. Jos aloituskomento annetaan sovelluksen puolelta ja pelille asetetaan esimerkiksi aikaraja, niin tukiasemalle lähetetään ensimmäisenä viesti, jossa se määrätään asettamaan aikaraja ja vasta tämän jälkeen aloituskomento. Viestien välissä on myös pieni viive, jotta tukiasema tunnistaa ne eri viesteinä.

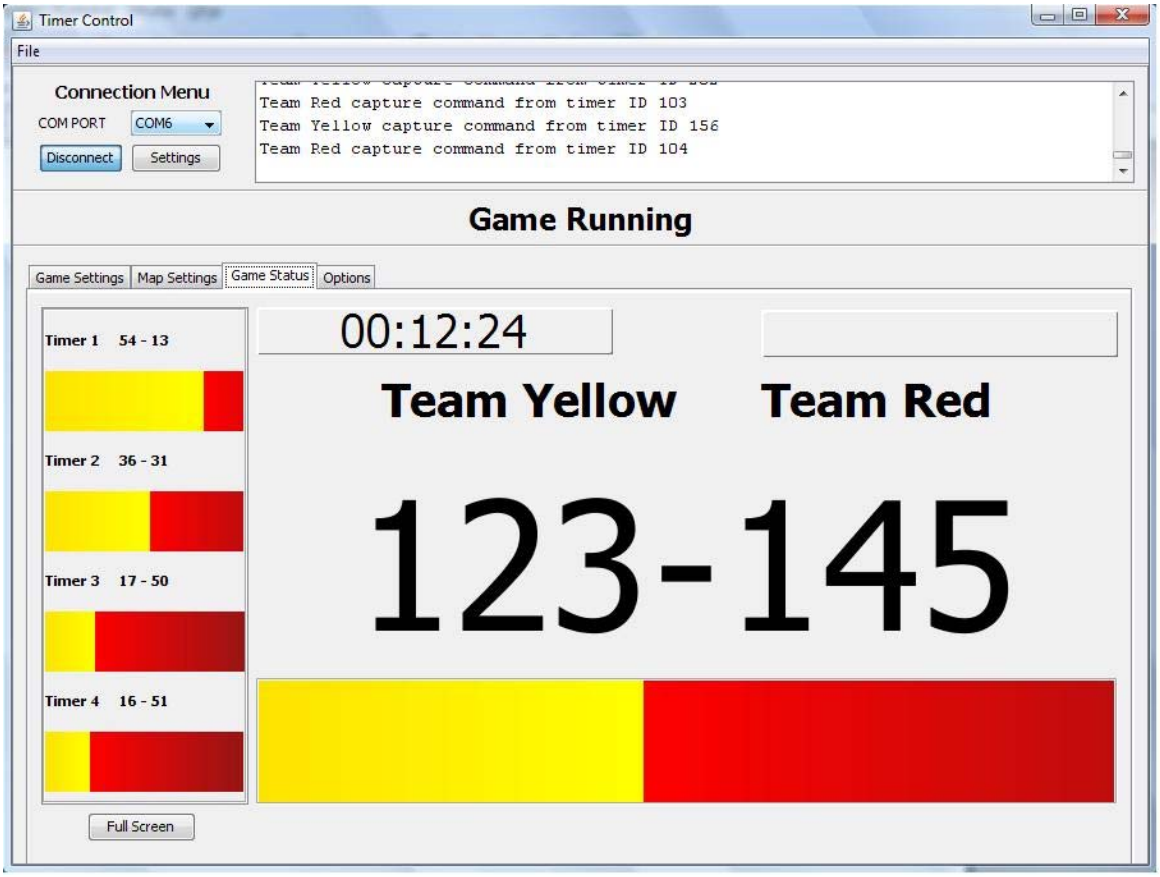


Kuva 8. Viestiliikenne sovelluksesta käynnistettäessä

Kun tukiasemalta tulevat komennot vastaanotetaan, ne käsitellään parserissa, joka määrittää, mikä komento oli kyseessä ja millaisia parametreja se tuo mukanaan. Peruspelimoodissa ainoat tulkittavat viestit ovat: aloituskomento, pysäytyskomento, lopetuskomento, kellojen omistajan vaihtumiskomento sekä ennen pelin alkua tulevat kellojen ilmoittautumiskomennot. Jos esimerkiksi tukiasemalta tulee pelikellon omistajan vaihtumiskomento, niin sen mukana saapuu myös tieto, mikä pelikello ja minkä värinen joukkue on kyseessä. Nämä tiedot tulkitaan viestistä ja annetaan funktiolle, joka käsittelee tapahtuman.

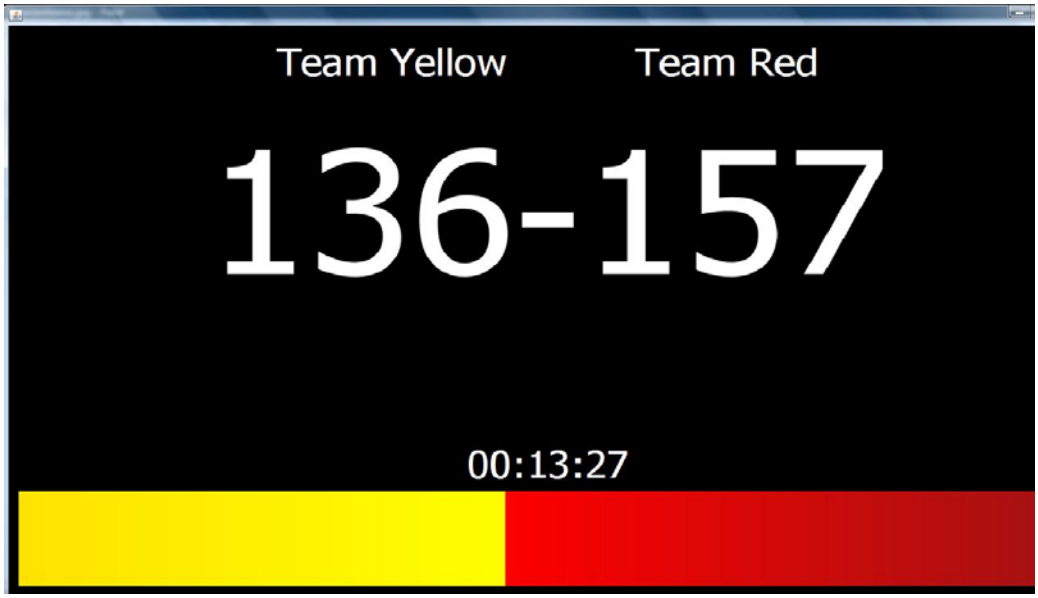
6.3. Pelitilanneikkuna

Pelitilanneikkuna ilmoittaa pelin pistetilanteen sekä ajan kulun. Ikkunan on tarkoitus antaa pelaajille informaatiota pelin pistetilanteesta, joten ikkunoissa on käytetty isoja fontteja sekä kaksiväristä pistepalkkia. Ikkunassa on myös jokaiselle pelikellolle oma pisteidenlasku sekä pistepalkki. Ylälaidassa olevat ajat kertovat kokonaisajan sekä kuinka kauan mahdollinen tauko on kestänyt. Kuvassa 9 nähdään pelitilanne-esimerkki.



Kuva 9. Pelitilanneikkuna

Ikkunasta on myös mahdollisuus aktivoida kokoruututila, joka on nähtävillä kuvassa 10. Tämä avaa uuden ikkunan, josta näkee todella isolla fontilla pistetilanteen sekä näyttää pistepalkin. Tämä ominaisuus haluttiin siksi, että pelaajien ei tarvitse tulla katsomaan pisteitä ihan tietokoneen ruudun viereltä nähdäkseen ne. Näkyvyyttä parantamaan väriteemana on valkoinen teksti mustalla pohjalla. Ikkuna on siirrettävissä, joten jos käytettävissä on esimerkiksi toinen näyttö, niin kokoruutuisen pisteikkunan pystyy siirtämään toiselle näytölle. Tämän ansiosta pääruudulla, joka yleisimmin on pieni kannettavan tietokoneen ruutu, voidaan pitää jotain toista ikkunaa auki.



Kuva 10. Pelitilanneikkunan kokoruututila

Pelin alkaessa ohjelma luo pelitilanneikkunan vasempaan laitaan jokaiselle kellolle oman pistelaskurin. Pistelaskurit ovat nähtävissä kuvan 9 vasemmassa laidassa. Tähän kuuluu kummankin puolen pisteet kyseiseltä kellolta sekä pienikokoinen pistepalkki. Kun pelin kuluessa kellot vallataan, tukiasema lähettää viestit ohjelmalle vallatuista kelloista ilmoittaen ohjelmalle kellon ID:n sekä värin. Tämän mukaan omistajat asetetaan vektoriin, jossa pidetään kirjaa kunkin kellon senhetkisistä haltijoista. Kun ohjelman ajanlaskukello laskee aikaa, se samalla käy läpi jokaisen pelikellon omistajat ja lisää aikaa myös kellojen omistajille. Nämä ajat ovat säilöttynä kummankin joukkueen omissa taulukoissa, joissa ensimmäinen alkio vastaa ensimmäistä kelloa ja toinen alkio toista kelloa ja niin eteenpäin. Kellopeli, joka on tällä hetkellä ainoa pelimoodi, antaa joukkueelle pisteen jokaisesta kymmenestä sekunnista mitä joukkue on pitänyt kelloa hallussaan. Nämä pisteet lasketaan yhteen, josta saadaan pelin kokonaispisteet kummallekin joukkueelle.

```
public void updatepoints()
{
    int yp=0,rp=0; //tilapäiset kokonaispisteet keltaiselle ja punaiselle
    for(int i=0;i<not;i++)  //käydään läpi jokaisen kellon ajat (not=number of timers)
    {
        //ajat ovat millisekunneissa joten 1000=sekunti ja 10 sekuntia=piste joten jako 10000:lla
        yp+=Ytimertimes[i]/10000; //lisätään kokonaispisteisiin käsiteltävän kellon pisteet
        rp+=Rtimertimes[i]/10000;
        //funktio joka asettaa i pistepalkin värien suhteen
        TPP[i].setmid((int)(Ytimertimes[i]/10000),(int)(Rtimertimes[i]/10000));
```

```
        }
….
}
```

Jos pelin alussa on määritetty asetus, joka antaa joukkueille vielä lisäpisteet jokaisesta valtauksesta, lisätään nekin vielä loppusummaan.

```
if(GM1.GetPPC()!=0)//jos PPC(points per capture) on asetettu
{
    yp+=YTimesCaptured*GM1.GetPPC();
    rp+=RTimesCaptured*GM1.GetPPC();
}
```

Tämän jälkeen yp- sekä rp-arvoja voidaan käyttää moneen eri funktioon, jossa tarvitaan kokonaispisteitä.

Pistepalkki on joukkueiden väreillä muodostettu graafinen esitys pistetilanteesta. Värin osuus palkissa määräytyy joukkueiden pisteiden suhteesta toisiinsa. Jos esimerkiksi keltaisella joukkueella on 20 pistettä ja punaisella 60, niin keltaisen värin osuus palkissa on 25 % ja punaisen 75 %.

```
public void setmid(int yp, int rp)
{
    mid=(int)(((float)yp/(float)(rp+yp))*100);
    //mid=prosenttimäärä jonka keltaiset omistavat pisteistä 100-mid= punaisten määrä
    repaint();  //pistepalkki päivitetään
}
```

6.4. Karttaikkuna

Karttaikkunan tarkoitus on helpottaa pelin seurantaa. Se antaa paremman kuvan kellojen hallitsijoista pelikentällä. Karttaikkuna antaa käyttäjän asettaa kuvatiedoston ohjelmaan, jonka on tarkoitus olla kartta pelialueesta, jolla peli käydään. Tähän karttaan voidaan lisätä kellojen sijainnit, jotka aluksi esiintyvät vihreinä palloina. Nämä pallot muuttavat väriään sen mukaan, minkä värinen joukkue hallitsee kyseistä kelloa.

Kartan lisääminen ikkunaan tapahtuu perinteisellä tiedostoselaimella, joka hyväksyy ja näyttää pelkästään .jpg- ja .gif-päätteisiä tiedostoja. Kun tiedosto on valittu, se ilmestyy sille tarkoitetulle paneelille skaalautuen automaattisesti sopivan kokoiseksi. Normaalisti kuva levittäytyy sille määrätylle alueelle unohtaen kokonaan mittasuhteet. Halutessa kumminkin voidaan pitää mittasuhteet tallella, joten kuvasta ei tule venytetyn näköistä. Siksi

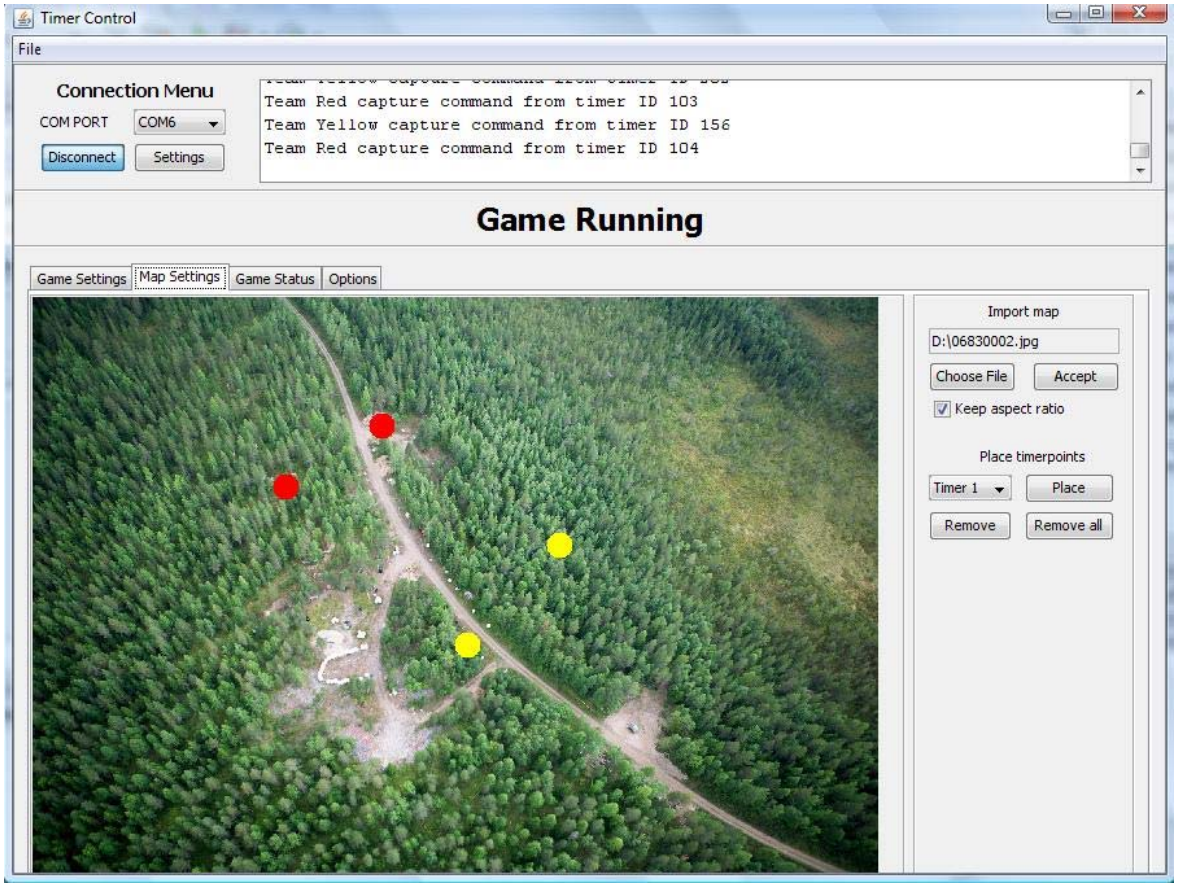
olisikin hyvä, että käytettävä karttakuva olisi mielellään kuvasuhteeltaan sopivan kokoinen ohjelmalle.

Kartan asettaminen paneelille ilman, että kuvasuhdetta halutaan säilyttää, tapahtuu seuraavasti.

```
Dimension d = this.getSize();
Insets i = this.getInsets();
if(!keepaspectratio)
      g.drawImage(image, i.left, i.top, d.width- i.left- i.right, d.height- i.top - i.bottom, this );
else
{
….//kun kuvasuhde halutaan säilyttää
}
```

Pelikellot, joita pelikentälle asetetaan, ovat valittavissa tiputusvalikosta, jossa on listattuna kaikki pelissä käytettävät pelikellot. Kun haluttu kello on valittu, niin ”Place”-napin painamisen jälkeen voidaan klikata haluamaansa kohtaan kartalle, johon haluaa kellon asettaa. Jos peli ei ole vielä käynnistynyt, kellot ilmestyvät vihreinä palloina. Vihreä tarkoittaa, että kello ei ole vielä kenenkään hallussa. Kun karttaan klikataan kelloa asetettaessa, koordinaatteja kartalla ei oteta talteen, vaan suhdeluku kuvaan nähden, johon se on asetettu. Tämä sen takia, koska kuvan koko pystyy muuttumaan ohjelman ikkunan kokoa muutettaessa. Suhdelukua käyttäen kellojen paikat pysyvät aina kohdallaan, kävi kuvan koolle mitä tahansa. Kuvassa 11 on mahdollista nähdä pelitilanne-esimerkki, jossa on käytössä 4 kelloa ja joista kumpikin joukkue hallitsee kahta kelloa.

```
public void setTimer(int x,int y,int i) //x ja y koordinaatit sekä  i=kellon numero
{
      Dimension d = this.getSize(); //paneelin mitat johon kuva on asetettu
      timersX[i]=(double)(x-TimerSize/2)/(double)d.width;  //suhde x suunta
      timersY[i]=(double)(y-TimerSize/2)/(double)d.height; //suhde y suunta
}
```



Kuva 11. Karttaikkuna ja esimerkki pelitilanteesta

6.5. Pelin lokit ja muut asetukset

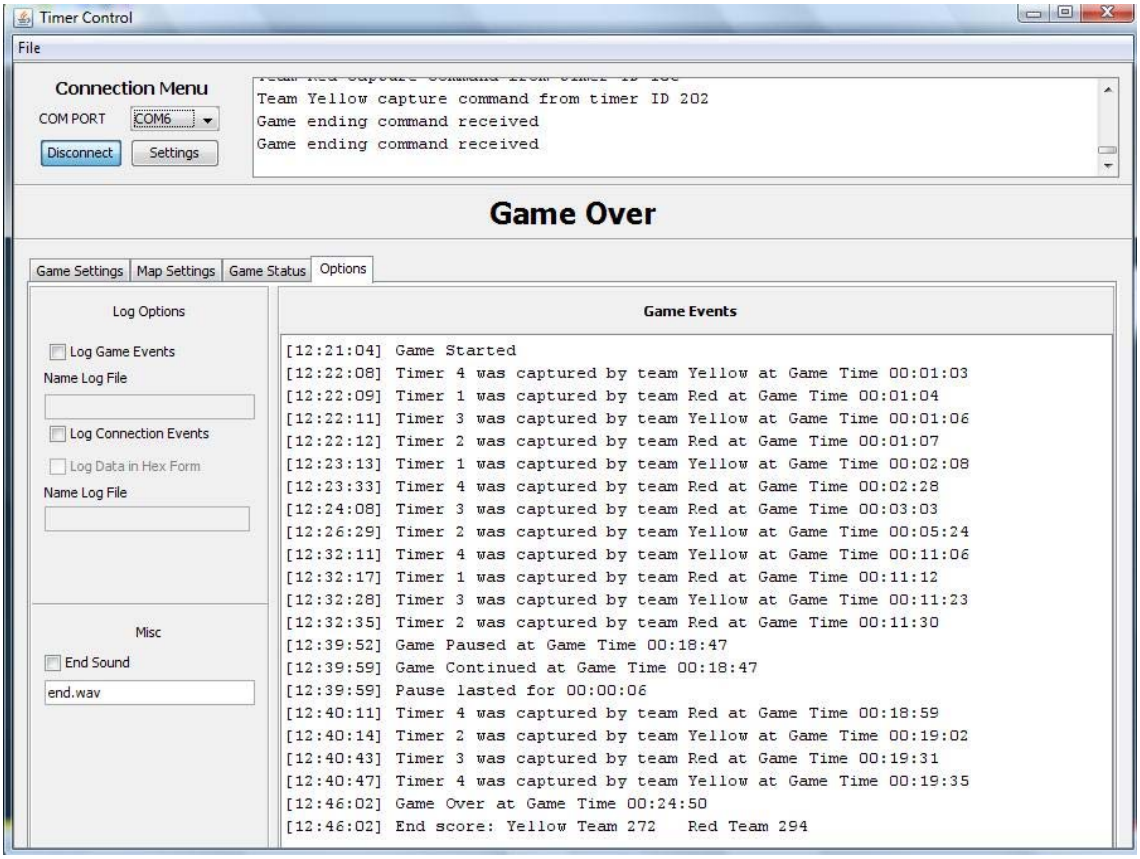
Ohjelmassa on mahdollista luoda pelin kulusta lokitiedostot. Lokitiedostoja on mahdollista luoda kaksi erilaista. Toinen kerää pelin tapahtumat ja toinen kerää sarjaliikenteen tapahtumat. Pelin tapahtumaloki kerää tekstitiedostoon pelin kulun ja sen lopputuloksen, kun taas sarjaliikenneloki kerää tekstitiedostoon jokaisen sarjaliikenteessä tapahtuneen tapahtuman. Jokaiseen tapahtumaan on lisätty aikaleima. Pelin tapahtumien lokitiedostoon kerääminen haluttiin siksi, että pelin kulkua voidaan myöhemmin tutkia ja se voidaan asettaa muidenkin nähtäväksi esimerkiksi foorumeille. Sarjaliikenteen lokimahdollisuus luotiin lähinnä testaustarkoituksiin. Pelin lokit voidaan asettaa päälle asetusvalikosta, josta niille voidaan myös tarvittaessa määrittää omat nimet. Jos tiedoston nimen määrittävä kohta jätetään tyhjäksi, niin kone asettaa tiedostolle automaattisesti nimen, joka olisi tapahtumalokin tapauksessa GameEvent ja jonka perään asetetaan aikaleima, joten tiedostot eivät voi saada samoja nimiä.

Tapahtumalokissa lokin alkuun tulee laatikko, jossa on pelin tietoja. Nämä tiedot ovat: päivämäärä, pelimoodi, pelaajien määrä(jos ne on ilmoitettu pelin alussa) sekä mahdolliset aika- ja pisterajat. Peli kerää lokiin aloituskomennon jälkeen kellojen valtaustapahtumat sekä mahdolliset pelin pysäytys- ja jatkotapahtumat sekä lopuksi lopetustapahtuman. Tapahtumalokiin kokeiltiin myös laittaa ilmoitukset aina, kun joukkue saisi pisteen, mutta tämä

teki lokitiedostosta ihan liian sekavan. Nämä tapahtumat pätevät kellopelimoodiin, joten mahdollisissa myöhemmin lisättävissä pelimoodeissa lokiin kerätään sille pelimoodille oleelliset asiat.

Asetusvalikossa ei vielä tämänhetkisessä versiossa ole lokien lisäksi säädettävissä muuta kuin mahdollinen lopetusääni, jonka ohjelma soittaa, kun peli päättyy. Tämä ominaisuus on myös pisteidenlaskujärjestelmän kelloissa, mutta niiden ääni kuuluu vielä melko heikosti. Myös sovelluksen soittama ääni ei ole oikein optimaalinen, jos se soitetaan kannettavan kaiuttimista, mutta ainakin se antaa merkin, jonka voi kuulla lähietäisyydeltä. Äänen voi itse määrittää, kunhan se on .wav-muotoa. Ohjelman mukana tulee kumminkin yksinkertainen lopetusääni, joka on luotu saha-aallosta äänenkäsittelyohjelmalla.

Kuvassa 12 on asetusvalikko, jossa on myös nähtävissä pelin kulku, joka on hyvin samantapainen kuin itse luotavat lokitiedostot. Tapahtumat tulostuu tekstialueeseen ja ne ovat samaa muotoa kuin lokitiedostossa. Näin pelin kulkua voi vielä tutkia, kun peli on käynnissä.



Kuva 12. Asetusvalikko

7. TESTAUS

Ohjelman testaus oli suurilta osin aina yhden lisätyn osion testaamista kerrallaan. Kun testauksen jälkeen osio toimi ongelmitta, oli aika siirtyä seuraavaan osioon. Tällä tyylillä yleensä osio toimi pääpiirteittäin oikein, mutta mahdollisia virheitä saattoi löytyä seuraavan osion testissä. Ohjelman testaamisen kanssa lähes joka osiossa mukana täytyi olla pisteidenlaskujärjestelmä. Koska pisteidenlaskujärjestelmän ohjelmointi tapahtui samaan aikaan kuin tietokonesovelluksen, niin testauksessa oli mahdollista löytää samaan aikaan virheet molemmista koodeista. Kun kriittisimmät osiot olivat valmiit, koko järjestelmän testausta pystyttiin toteuttamaan. Tämä vaati paljon aikaa, sillä yksikin erä, joka järjestelmällä tullaan pelaamaan, voi kestää yli kolme tuntia ja on oltava varma, että tänä aikana ei ongelmia tule ohjelman eikä järjestelmän puolella. Suurimmat osat ongelmista yleensä ilmaantuivatkin pisteidenlaskujärjestelmän puolella.

Ohjelman testauttaminen toisella osapuolella oli lähes mahdotonta, sillä pisteidenlaskujärjestelmän pelikelloja ei ollut käytössä kuin viisi kappaletta ja testaaminen vaatisi mielellään kolme pelikelloa, joista yksi ohjelmoidaan tukiasemaksi. Koko järjestelmän testausta olisi pitänyt tehdä huomattavasti enemmän ja sitä olisi pitänyt tehdä samassa ympäristössä, missä sitä tullaan käyttämään. On suuri ero radiosignaaleja käyttävälle laitteelle testauksessa, käytetäänkö sitä pöydällä koneen ääressä, missä kaikki ovat lähellä toisiaan vai metsässä, jossa jokainen tulee olemaan melkein sadan metrin päässä toisistaan.

Kuva 13. Kuva sovelluksesta pelikäytössä

Järjestelmä kävi läpi yhden testauksen, joka toteutettiin ihan oikeassa pelitilanteessa. Tätä ennen järjestelmä testattiin kyseisellä pelialueella päivää ennen, jolloin todettiin, että viestien kantavuudessa oli ongelmia. Viestit eivät tahtoneet kantaa tukiasemalta kelloille, joten pelikelloja jouduttiin liikuttelemaan bunkkerien sisällä, mihin ne oli tarkoitettu. Pienen liikuttelun ja testaamisen myötä viestit tukiasemalta kantoivat pelikelloille. Testaus tapahtui lähettämällä valtausviestiä pelikellolta tukiasemalle. Tätä viestiä lähetettiin useampaan otteeseen jokaiselta pelikellolta. Usean testauksen jälkeen tyydyttiin tulokseen, sekä otettiin muistiin kellojen paikat, jotta viestit myös seuraavana päivänä kuuluisivat tukiasemalle.

Itse pelipäivänä järjestelmä vielä nopeasti testattiin ennen pelien aloitusta ja kuuluvuudet vaikuttivat olevan todella hyvät. Peli käynnistettiin kannettavalta tietokoneelta, joka oli asetettu pelialueen ulkopuolella olevaan telttaan. Tämä tilanne on nähtävissä kuvassa 13. Käytössä oli myös kannettavaan lisätty isompi näyttö, johon pystyttiin asettamaan pelitilanneikkunan kokoruututila. Tämän ansiosta pystyttiin pitämään kannettavan ruudulla karttaikkunaa. Ensimmäinen erä sujuikin lähes ongelmitta ja pelaajat kävivät tarkistamassa pelin tilannetta useaan otteeseen teltasta. Järjestelmä sai hyvää palautetta pelaajilta pelin aikana. Toinen erä näytti, miten vähäinen testaus voi kostautua. Pelin alusta asti tuli jo yhden pelikellon kanssa yhteysongelmia ja vähän myöhemmin itse tukiasema lopetti toimimasta, joten ohjelma ei saanut enää uusia tietoja pelistä. Tämä ei kuitenkaan haitannut pisteidenlaskua, sillä niin kauan kuin pelikellot pyörivät, ne laskevat aikaa sitä hallitsevalle joukkueelle. Pelin loputtua kun jokaisen kellon ajat otettiin talteen, voitiin laskea jokaisen joukkueen pisteet käsin. Tämä testaus oikeassa pelitilanteessa antoi todella hyvän kuvan, mitä järjestelmässä ja sovelluksessa on vielä jatkokehitettävää.

8. JATKOKEHITYS

Koska pisteidenlaskujärjestelmästä on aikomus tehdä kaupallinen, vaatii se vielä huomattavaa jatkokehitystä. Ensimmäiseksi tulee tietenkin korjata ongelmat, joita löydettiin testissä laitteistosta ja sen ohjelmoinnista. Tämän lisäksi on jo valmiiksi suunnitteilla järjestelmän ja ohjelmiston välisen viestiprotokollan uusiminen sekä tapa, millä tavalla järjestelmä tunnistaa alueella olevat pelikellot. Nämä muutokset vaikuttavat myös sovellukseen, jonka tulee kehittyä järjestelmän mukana.

Isoimmat muutokset, mitä sovellus tuotteen kehittyessään saa, ovat pelimoodien mahdollinen lisääminen. Kellopeli on vain yksi monista pelimoodeista, joita järjestelmällä on mahdollista toteuttaa. Uusien pelimoodien lisääminen saattaa tuoda sovellukseen hyvinkin isoja muutoksia, sillä jokaisessa osiossa ei ole laajennusvaraa otettu vielä kovin hyvin huomioon. Laajennusvaraa pitäisi olla sen verran, että lisäykset on mahdollista toteuttaa kirjoittamatta sovellusta kokonaan uusiksi. Myös paintballia on mahdollista pelata kolmella eri joukkueella. Tämä laajennusvara on ollut hyvin pienellä huomiolla sovellusta ohjelmoitaessa. Yhden joukkueen lisäys toisi myös suuria muutoksia sovellukseen, mutta sen ei pitäisi myöskään olla mahdoton asia.

Sovelluksen käyttöönotto tulee tehdä helpoksi sen käyttäjälle, joten tämän eteen tulee myös tehdä muutoksia sovellusta ajatellen. Jotkin valikot voivat vielä olla hiukan sekavia, mutta näihin on tulossa muutoksia. Myös pienimuotoinen manuaali tehdään jossain vaiheessa. Koska ohjelman sarjaliikenne tarvitsee toimiakseen Java Communications API -lisäosan, on nämä tiedostot myös tarjottava sovelluksen mukana. Ettei käyttäjän itse tarvitsisi asentaa lisäosaa, jonka mukana ei myöskään tullut minkäänlaista asennustiedostoa, on sellainen luultavasti tehtävä itse. Tiedostot on asetettava Javan eri kansioihin ja tämä olisi joillekin käyttäjille liian vaativa toimenpide. Asennustiedosto hoitaisi tämän automaattisesti ja samainen asennustiedosto tekisi myös sovellukselle mahdolliset pikakuvakkeet ja asentaisi sen haluttuun polkuun.

Järjestelmä tulee myös varmasti saamaan vielä paljon kehitysideoita itse paintball-pelaajilta tulevien testien aikana. Näihin ehdotuksiin ollaan varmasti valmiita vastaamaan, kun ne ovat vain toteutettavissa sekä nähdään tarpeellisiksi.

9. YHTEENVETO

Tämän opinnäytetyön tuloksena syntyi ohjelmiston ensimmäinen versio, jota voidaan käyttää tämänhetkisen pisteidenlaskujärjestelmän kanssa. Ohjelmisto tarvitsee vielä pientä viimeistelyä ja se kehittyy sitä mukaa kun pisteidenlaskujärjestelmä kehittyy.

Itse sovellus ei aiheuttanut suuria ongelmatilanteita missään vaiheessa, vaan suurin osa ongelmista ilmaantui itse pisteidenlaskujärjestelmän ohjelmoinnin kanssa. Ennen työn aloitusta isoimpana mahdollisena ongelmana pidettiin tietoliikennettä pisteidenlaskujärjestelmän ja sovelluksen välillä. Tämä osoittautuikin lopulta onnistuvan hyvin helposti, eikä vienyt pitkääkään aikaa, kun tietoliikennettä päästiin testaamaan.

Nähtäväksi vielä jää, kuinka hyvin sovellus tulee suoriutuu mahdollisista isoista lisäyksistä. Olisi hyvä, että koko ohjelmaa ei tarvitsisi kirjoittaa täysin uusiksi ainakaan ihan heti. Tämä versio tulee kumminkin pärjää vielä hyvin siinä tarkoituksessa, mihin pisteidenlaskujärjestelmä alunperin rakennettiin. Järjestelmän laajennusmahdollisuudet ovat niin suuret, että on mahdotonta tässä vaiheessa vielä ottaa huomioon kaikkia mahdollisia asioita, joista sovelluksen pitäisi tulla selviytyä tulevaisuudessa.

Itse opinnäytetyö antoi lisää kokemusta ohjelmoinnista ja toi myös mukanaan täysin uusia asioita. NetBeans-ohjelmointiympäristö teki monet asiat helpoksi, enkä ole vielä ehtinyt tutusta lähellekään kaikkiin sen tarjoamiin asioihin. Hieman häiritsevää ehkä oli, että joutui samaan aikaan ohjelmoimaan kahdella eri kielellä, joista toinen tuntui niin paljon helpommalta ja toinen taas toisinaan todella työläältä.

10. LÄHDELUETTELO

/1/ A Brief History of NetBeans, [WWW-dokumentti], <http://www.netbeans.org/about/history.html> 12.12.2009.

/2/ Kuinka paintball sai alkunsa, [WWW-dokumentti], <http://www.paintball.fi/page.php?id=194> 12.12.2009.

/3/ Mitä on paintball, [WWW-dokumentti], <http://splatweb.net/artikkelit/1/mita-on-paintball?.html> 12.12.2009.

/4/ Wireless USB Adapter 868 MHz AMB8460, [WWW-dokumentti], <http://amber-wireless.de/71-1-AMB8460.html> 16.12.2009.

/5/ Yleiskuvas Javasta, [WWW-dokumentti], <http://cs.joensuu.fi/~vouti/tjdoku/JAVA/luku1.html#yleiskuvausjavasta> 12.12.2009.