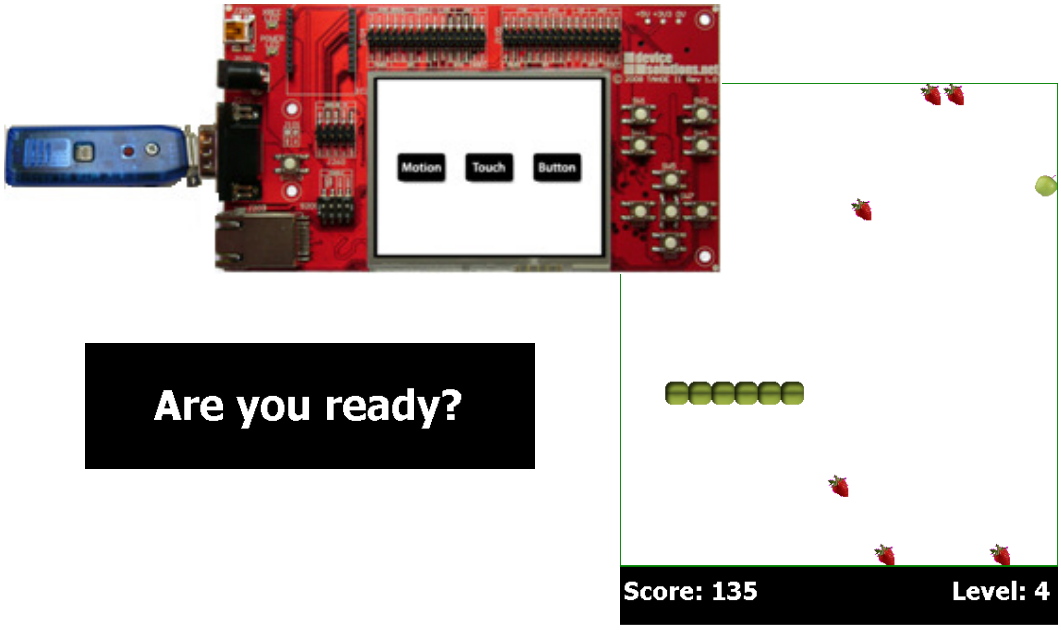


Remote m. Bluetooth Interface
Projekt Rapport

Hold: ITWEM1
Gruppe: Gruppe 11
Vejleder: Michael Alrøe

Projektdeltagere: Dennis Madsen [20060941]
Leni Guldbrandt Lausdahl [06721]
Martin Nordahl [06732]

Afleveringsdato: 4.6.2009

Godkendelsesformular

Sted og dato:

Ingeniørhøjskolen i Århus, den _____

Vejleder underskrift:

_____
Michael Alrøe

Deltager underskrift:

_____
Dennis Madsen 20060941

_____
Leni Guldbrandt Lausdahl 06721

_____
Martin Nordahl 06732

Indholdsfortegnelse

Figuroversigt

1 Indledning

Der ønskes udviklet en remote (fjernbetejning) med Bluetooth interface til synkronisering mellem et .NET Micro Framework kompatibelt hardware, samt en Windows Mobile client.

Eksempelvis er spilfunktionalitet i HTC Touch Diamond smartphone begrænset til kun at benytte touch samt accelerometer styring. Det blev vurderet at nogle brugere måske kunne ønske dette udvidet, hvorfra ideen bag udviklingen af en remote blev til.
Et parallel til dette projekt kunne være at en bruger benytter sin Sony PlayStation Portable (PSP) som remote til en Sony PlayStation 3 (PS3).

Til at teste denne remote, blev det besluttet at en testplatform skulle udvikles i form af et simpelt spil i stil med den populære klassiker "Snake" kendt fra Nokias mobiltelefoner.

Projektets fokusområde bliver derfor en blanding af samspillet mellem de to platforme, samt udviklingen af et Snake spil til en Windows Mobile client.

Dette arbejde vil blive beskrevet mere dybdegående i rapportens kommende afsnit. Hvis der ønskes et generelt overblik over systemets funktioner og virkemåde, kan brugervejledningen med fordel læses inden videre gennemlæsning af dokumentet. Denne brugervejledning er vedlagt som Appendiks 2.

I det efterfølgende afsnit vil læseren blive præsenteret for den overordnede strukturering af dette dokument.

Læsevejledning

Denne rapport er opdelt ved først at introducerer læseren for den overordnede system arkitektur, det være sig en high level beskrivelse af systemet enheder. Dette afsnit vil ligeledes fokuserer på at belyse de tekniske overvejelser, som danner grundlaget for systemets hardware.

Herefter introduceres læseren for systemets hardware og tilhørende grænseflader. Dette afsnit vil indeholde en beskrivelse af de enkelte hardware blokke og definerer hvilke interfaces/kommunikationsformer der anvendes mellem enhederne.

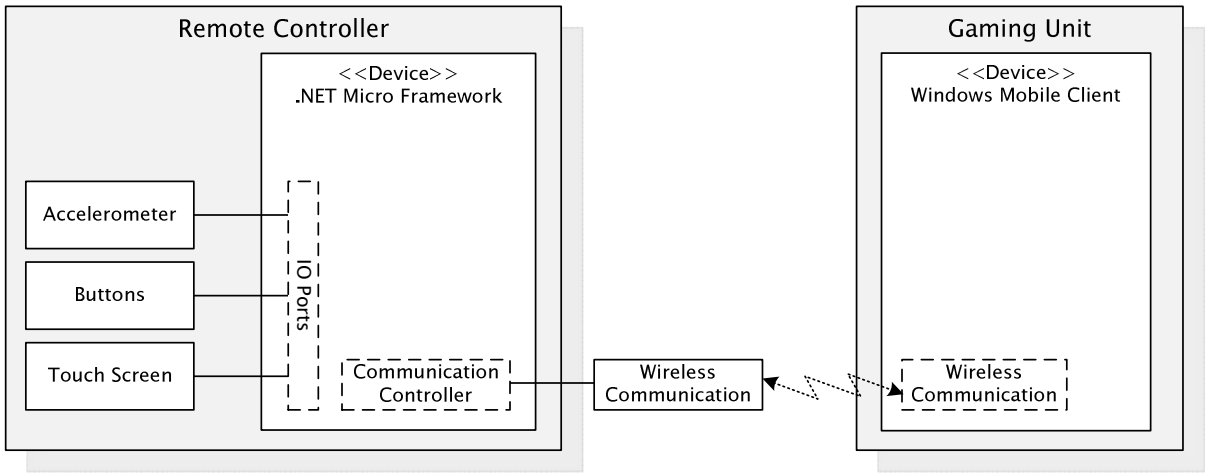
Software designet vil herefter blive præsenteret vha. UML-diagrammer samt med forklarende tekst til, hvorledes den valgte funktionalitet er implementeret.

Afslutningsvis vil resultaterne af udførte tests samt en beskrivelse af hvorledes projektet kunne forbedres, blive præsenteret.
Dette afsnit ligger op til konklusionen hvor der samles op på projektforløbet som en helhed samt de projektets fortræffeligheder.

2 Arkitektur Design

Dette afsnit vil fokuserer på at introducerer læseren for det overordnede system design, samt belyse de overvejelser og krav som er opstillede for de enkelte blokke.

Nedenfor, jf. Figur 1, vises sammenhængen mellem systemets forskellige enheder:



Figur 1. Skitsering af systemets overordnede arkitektur (High Level Design)

Systemet består altså af to hovedelementer:

- En .NET Micro Framework baseret fjernbetjening, hvorfra brugeren kan styre det udviklede spil.
- Samt en Windows Mobile baseret spilleenhed, hvorfra det udviklede spil afvikles.

2.1 Design overvejelser

Dette afsnit vil beskrive hvilke overvejelser der er gjort i forbindelse med valg af enheder, de opstillede krav samt opdelingen af systemets funktionalitet.

2.1.1 Spilenhed

Som tidligere nævnt ville en oplagt testkandidat til udvikling af Snake spillet være en HTC Touch Diamond smartphone.

Denne smartphone baserer sig på Windows Mobile Professional 6.1, samt at den var til rådighed gennem en af projektets deltagere.

2.1.2 Fjernbetjening

Den største overvejelse som skulle gøres for denne enhed, var udvælgelse af betjeningsmuligheder. Den mest simple form for styring har altid været, og vil altid være vha. trykknapper. Men for at give brugeren mere indlevelse i spillet, skulle der tages flere muligheder i brug.

Følgende ekstra betjeningsmuligheder findes og vil kunne anvendes:

- Gyroskop, hvor dennes kompensation for at opnå ligevægt benyttes som styringssignal.
- Accelerometer, hvor en transducer måler g-påvirkning (acceleration) i en given retning (x, y og z).
- Touch screen, hvor berøring af skærmens overflade oversættes til et koordinat.

Det blev besluttet at fjernbetjeningen skulle anvende .NET Micro Framework'et, hvorved et Tahoe-II development board kunne benyttes. Dette development board var allerede tilkoblet trykknapper,

accelerometer samt touch screen, hvorved brugeren selv vælger hvilken betjeningsform vedkommende ønsker at benytte.

De tre valgte betjeningsmuligheder var ligeledes et godt supplement til de eksisterende på spilenheden, da HTC Touch Diamond'en allerede havde muligheden for touch samt accelerometer styring. Herved ville brugeren få den fornemmelse, at fjernbetjeningen blot er en udvidelse til den eksisterende styring.

2.1.3 Wireless Kommunikation

Det blev valgt at implementerer kommunikationen mellem systemets enheder vha. en form for wireless. Dette var et ønske for at undgå kabelforbindelser og derved give brugeren mere frihed til at bevæge sig frit.

Af wireless kommunikations former, kunne følgende benyttes:

- IrDA, hvor infrarødt benyttes til at sende en form for seriel data.
- Bluetooth, hvor der skabes en parring mellem to enheder.

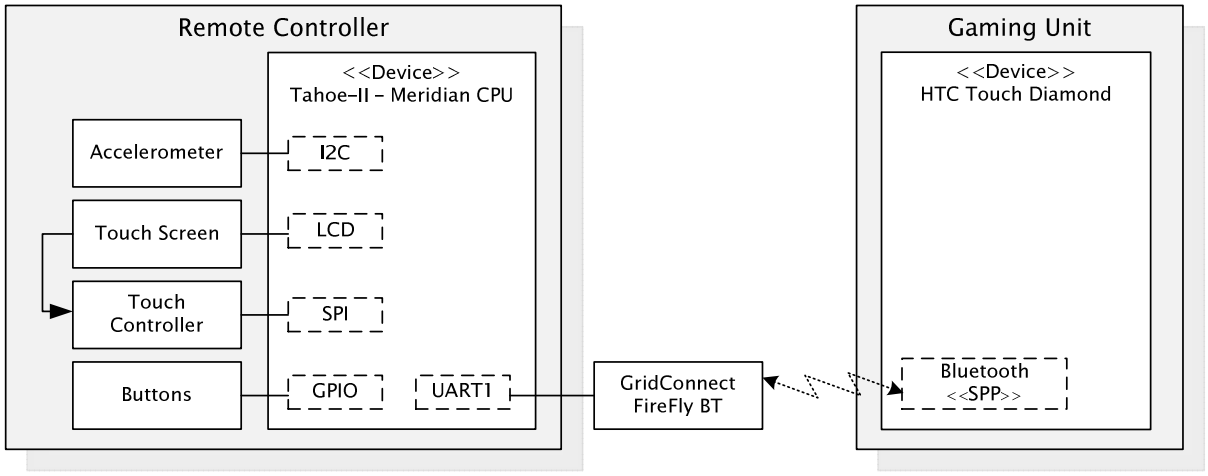
HTC Touch Diamond'en understøtter kun Bluetooth, hvorved IrDA ville kræve tilkobling af eksternt hardware på både spilenheden samt fjernbetjeningen. Dette ville stride mod ønsket om at undgå unødvendige kabelforbindelser.
Bluetooth blev derfor valgt som den mest egnede kommunikationsform mellem de to enheder. Dette ville blot kræve, at der på fjernbetjeningen skulle tilkobles en Bluetooth enhed.

3 Hardware Design

Dette afsnit vil beskrive de valgte hardware dele samt klarligge opsætning af de interfaces der anvendes til kommunikation med de tilkoblede enheder.

Der vil blive lagt særligt vægt på en beskrivelse af interfacet mellem de tilkoblede enheder.

Nedenfor, jf. Figur 2, vises et blokdiagram for de væsentligste elementer/komponenter i systemet:



Figur 2. Blokdiagram for hardware designet

I de kommende underafsnit vil hardware designet af hovedblokken (Tahoe-II hardwaren) blive beskrevet.

3.1 Accelerometer

Tahoe-II development boardet har et indbygget accelerometer, som er tilkoblet Meridian CPU'ens I2C bus. Accelerometeret er af typen MMA7260QT fra Freescale og har en indbygget fire kanels analog til digital I2C konverter.

Denne fungerer ved at omdanne en acceleration til en spænding, hvorefter denne konverteres til et digitalt signal som kan aflæses over I2C bussen.
Der var tidligere opnået erfaring med dette accelerometer gennem en lab øvelse og det blev besluttet at benytte driveren herfra til aflæsning af data.

3.2 Buttons

Ni trykknapper på Tahoe-II kortet er tilkoblet GPIO's (Generel Purpose I/O) ben 1-9. Det ville derfor ikke være nødvendigt at tilkoble noget ekstern hardware for at benytte denne styringsform.

Det blev vurderet at en aflæsnings af tasterne med en frekvens på 10 Hz eller hvert 100ms ville mindske prel, samt sikre at tryk på knappen altid ville blive registeret.

3.3 Touch screen og controller

Selve skærmen styres af den indbyggede LCD controller på Meridian CPU, mens en ekstern touch kontroller tilkoblet SPI interfacet muliggør aflæsning af et koordinat, som fortæller hvor på skærmen brugeren har trykket.

3.4 Bluetooth interface

Der er på Tahoe-II development boardet ikke nogen indbyggede Bluetooth kontroller, og det var således nødvendigt at finde en ekstern enhed som kunne kommunikere med spilenheden.

Development boardet har mulighed for tilslutning af et Bluetooth interface på enten; USB, Ethernet samt UART'en. UART'en er tilkoblet en RS232 tranceiver, som gør det muligt at tilslutte denne direkte til en PC. Ved at betragte de tilgængelige muligheder blev det vurderet, at den letteste metode hvorved et Bluetooth interface kunne tilkobles Tahoe-II kortet, ville være gennem UART'en.

Derfor blev en FireFly fra firmaet GridConnect valgt til at implementerer interfacet. Enheden gør det muligt at transmitterer et serielt signal mellem to enheder over Bluetooth. Det kræves blot at den modtagne enhed enten ligeledes er tilkoblet en FireFly, eller at denne understøtter en Bluetooth profil kaldet SPP (Serial Port Profile).
HTC Touch Diamond'en understøtter SPP Bluetooth profilen, hvorved det for begge enheder blot ville kræve at implementerer kode til en seriel port driver, samt at få parret de to Bluetooth enheder.

FireFly'en benytter sig af et princip kaldet handshaking, hvor hosten afsender et 'Request to send' (RTS) signal til den tilkoblede enhed for efterfølgende at afvente svaret 'Clear to send' (CTS). Dette skal sikre at hosten ikke sender data til en enhed, som endnu ikke er klar til at modtage, hvorved der kan opstå data tab. For at omgå denne handshaking, er det muligt at forbinde en jumper internt på FireFly'en, således at RTS og CTS kortsluttes.
HTC'en skulle således have implementeret en funktionalitet, så den altid ville begynde at modtage det indkommende data fra serielporten.

FireFly'en havde mulighed for at kommunikere ved to hastigheder:

- 9600 baud
- Samt 115.200 baud

Baudraten er et udtryk for hvor mange pulser der sendes i sekundet over serielporten, og bruges til at synkroniserer hvornår et databit skal aflæses. Matcher de to enheders baudrate ikke, vil det modtagne data ikke kunne genskabes/genkendes.

For at gøre det muligt at afsende en fuld ASCII karakter (en byte) over serielporten, blev det besluttet at anvende følgende opsætning:

Navn	Type	Protokol	Bit	Baud rate	Paritet	Stop bit	Flow
Serial Communication	Digital/Seriel	UART	8	115.200	None	1	None

4 Software Design

Dette afsnit vil beskrive den mest kritiske funktionalitet for systemet, samt de overvejelser som er gjort i forbindelse med valg af design patterns.

Som tidligere nævnt, er systemet funktionalitet opdelt på to enheder:

- En fjernbetjening, navngivet *'BluetoothRemote'* (Tahoe-II, .NET Micro Framework)
- Samt en spilleenhed, navngivet *'RemoteSnake'* (Windows Mobile, .NET Compact Framework)

For at gøre kildekoden mere overskuelig samt genanvendelig, er koden blevet struktureret opdelt i forhold til et udvalgt design pattern. Der vil ikke være kildekode eksempler i dette afsnit, men beskrivende tekst, samt klasse- og sekvens-diagrammer for at højne den overordnede forståelse af systemet.

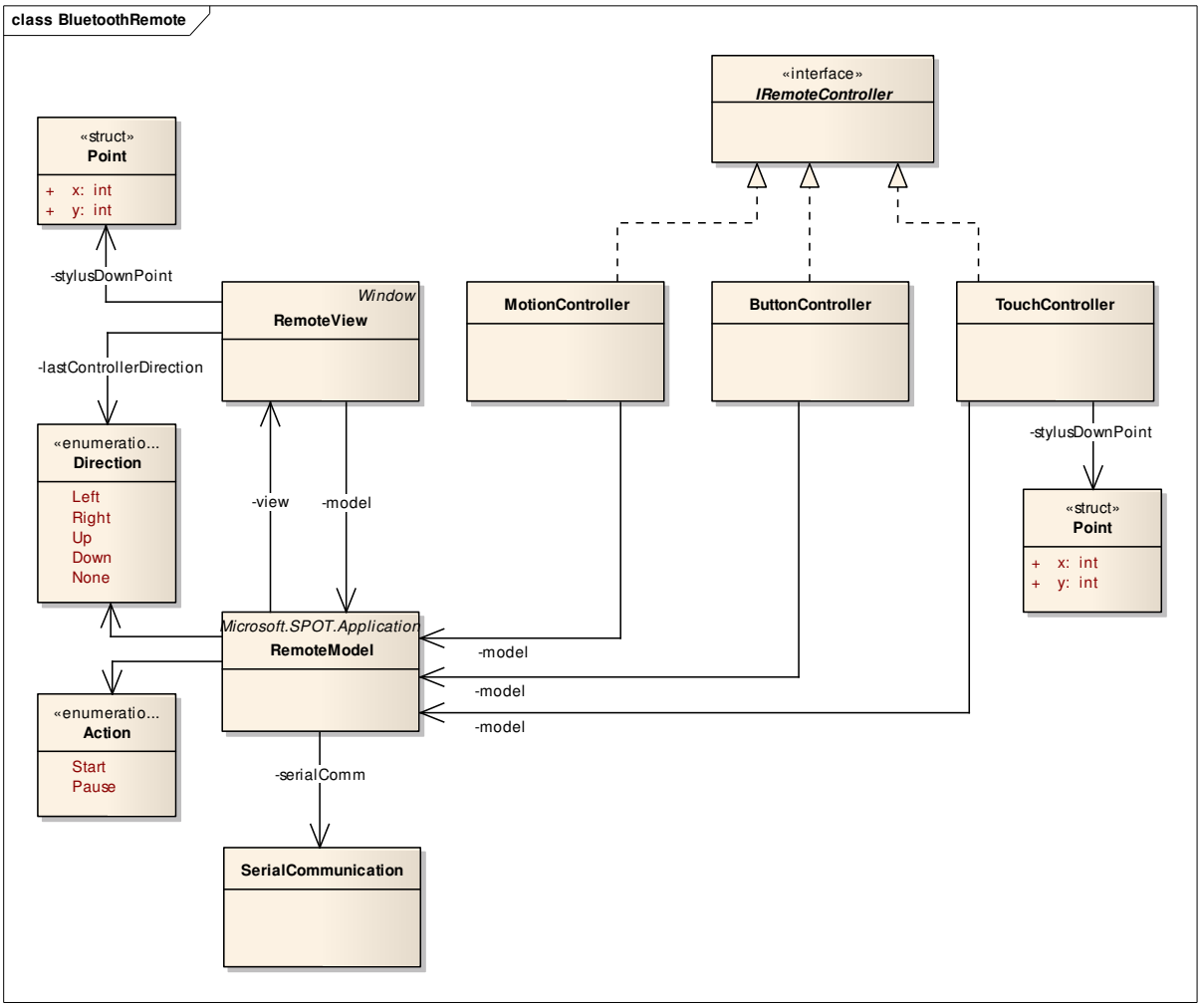
Der henvises i stedet til Appendiks 1 for den komplette kildekode.

4.1 Tahoe-II (.NET Micro Framework)

Dette afsnit vil beskrive software designet bag det udviklede *'BluetoothRemote'* program. Programmet er udviklet til .NET Micro Framework version 3.0, som er integreret i Meridian CPU'en på Tahoe-II development boardet.

For at sikre en objekt orienteret model opbygning, og derigennem gøre koden mere genanvendelig, blev det valgt at implementere denne del efter et MVC (Model-View-Controller) design pattern.

Nedenfor ses det endelige klassediagram for *'BluetoothRemote'* og viser arkitekturen bag det udviklede program, jf. Figur 3.

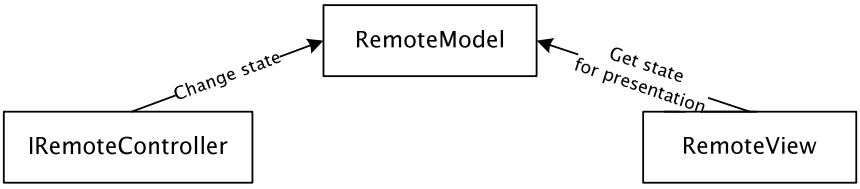


Figur 3. Klassediagram for .NET Micro Framework klassen 'Bluetooth Remote'

På de efterfølgende sider, vil der blive taget udgangspunkt i dennes associationer og udvalgte funktioner vil blive beskrevet yderligere. Men inden da, vil det valgte MVC design pattern først blive præsenteret.

Fordelen ved at benytte et MVC pattern til udviklingen af vores program, er muligheden for en fuldstændig afkobling mellem systemets business og præsentations lag. I MVC repræsenterer *Model* programmets tilstande. *View*'et er alt tilhørende den grafiske præsentation af tilstandene. Mens *Controller* benyttes til at ændrer tilstandene i *Model*.

Nedenfor ses en illustration af det implementerede MVC pattern, jf. Figur 4:



Figur 4. Det implementerede MVC design pattern

Alle grafiske elementer i programmet håndteres af klassen *RemoteView*, og tegnes på et *Image* påsat et *Panel* (begge fra Microsoft.SPOT.Presentation.Controls). Der er mulighed for at styre spillet vha. en af de tre udvalgte betjeningsmuligheder og *RemoteView* er grundet MVC pattern'et ligeglad med hvilken kontroller der er valgt. En af de tre kontrollere ændrer på data'en i *RemoteModel*, når brugeren udfører en godkendt handling, hvorefter *RemoteView*'et fremviser ændringen.

Der blev oprettet et interface navngivet *IRemoteController* mellem de tre kontroller klasser og *RemoteModel*; *ButtonController*, *MotionController* samt *TouchController*. Alle kontrollere ændrer altså på det samme data i RemoteModel, hvorved det er nemt senere at tilføje ekstra kontrollere til programmet.

Der blev i RemoteModel oprettet to metoder som tager sig af kommunikationen med spilenheden; *HandleControllerDirection* samt *HandleControllerAction*. Hver metode kaldes gennem kontroller klasserne, og bruges til at sende data ud på serielporten gennem klassen *SerialCommunication*. Til at håndterer retning og status, blev der oprettet to public Enums; *Direction* og *Action*, hvortil de hver fik tilknyttet en Property; *ControllerDirection* samt *ControllerAction*.
Instanser af de oprettet Enums kunne derved benyttes af alle kontrollere samt *RemoteView*, for at gøre programmet mere struktureret og læsbart.

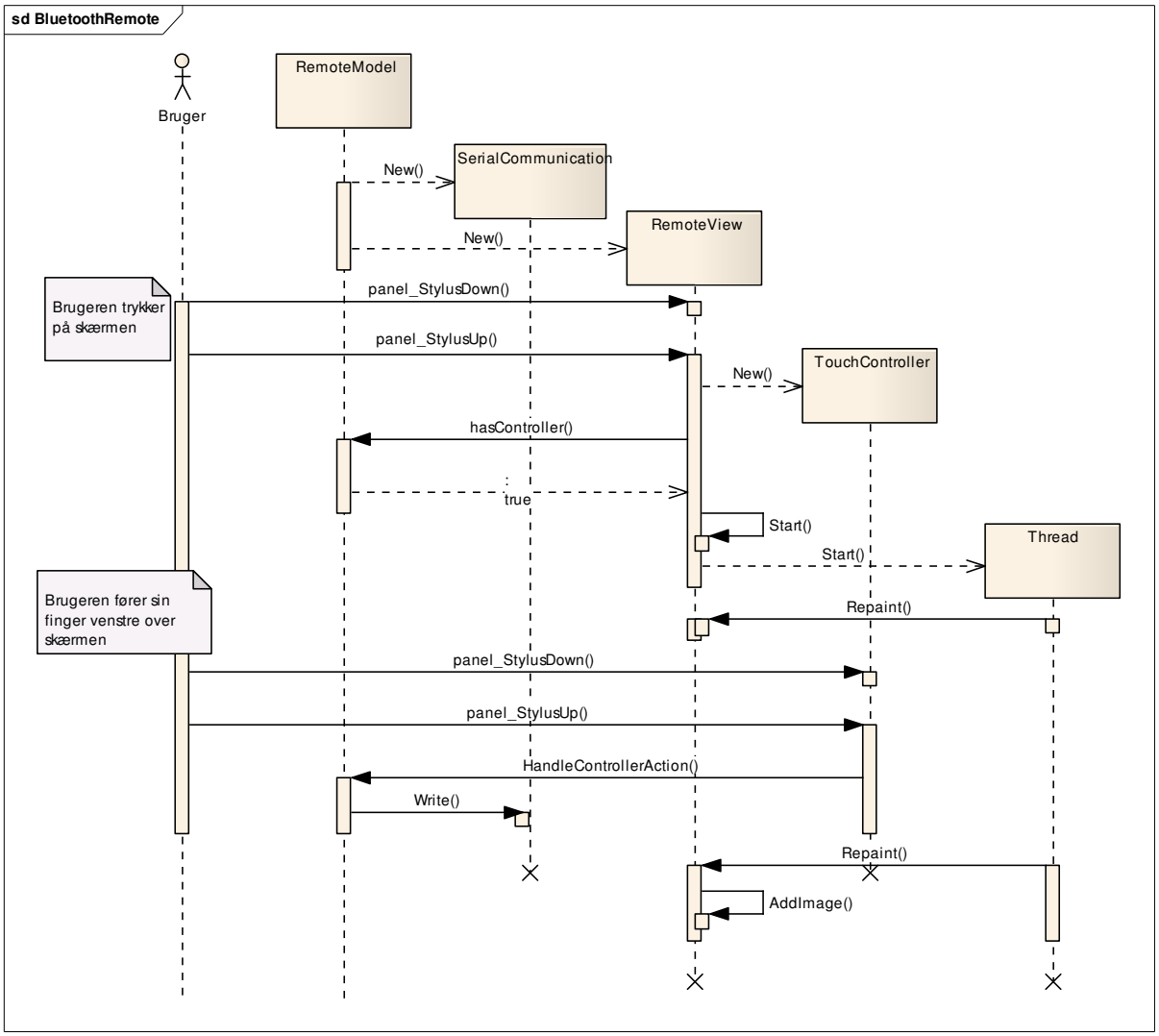
RemoteView'et er opsat til at køre i en tråd, hvor skærmen opdateres med en frekvens på 50Hz. Dermed skulle det ikke være muligt for brugeren at bemærke, at en instruktion ikke er blevet opfanget. Hver gang tråden gennemløbes, kontrolleres det hvorvidt retningen har ændret sig siden sidst. Er dette tilfældet, opdateres skærmen med den nye retning.

Når programmet opstartes, er der imidlertid ikke valgt nogen form for styring, og brugeren præsenteres derfor med muligheden for at udvælge en betjeningsform, jf. Figur 5.

Figur 5. Opstarts skærm på fjernbetejningen

Det blev besluttet at brugeren skulle udvælge den ønskede styring vha. touch funktionaliteten. Hvorved den ønskede styring registreres via et stylus-event i *RemoteView*. Herefter oprettes en af de tre kontroller klasser gennem *IRemoteController* interfacet.
For at undgå at registrerer yderligere stylus-events fra *RemoteView* klassen nedlægges dette event, hvorved brugeren kun har mulighed for styring gennem den valgte kontroller.

Nedenfor, jf. Figur 6, vises et sekvensdiagram for selve opstartsfasen af programmet. Dette diagram gør sig gældende ved valg af touch kontrolleren:



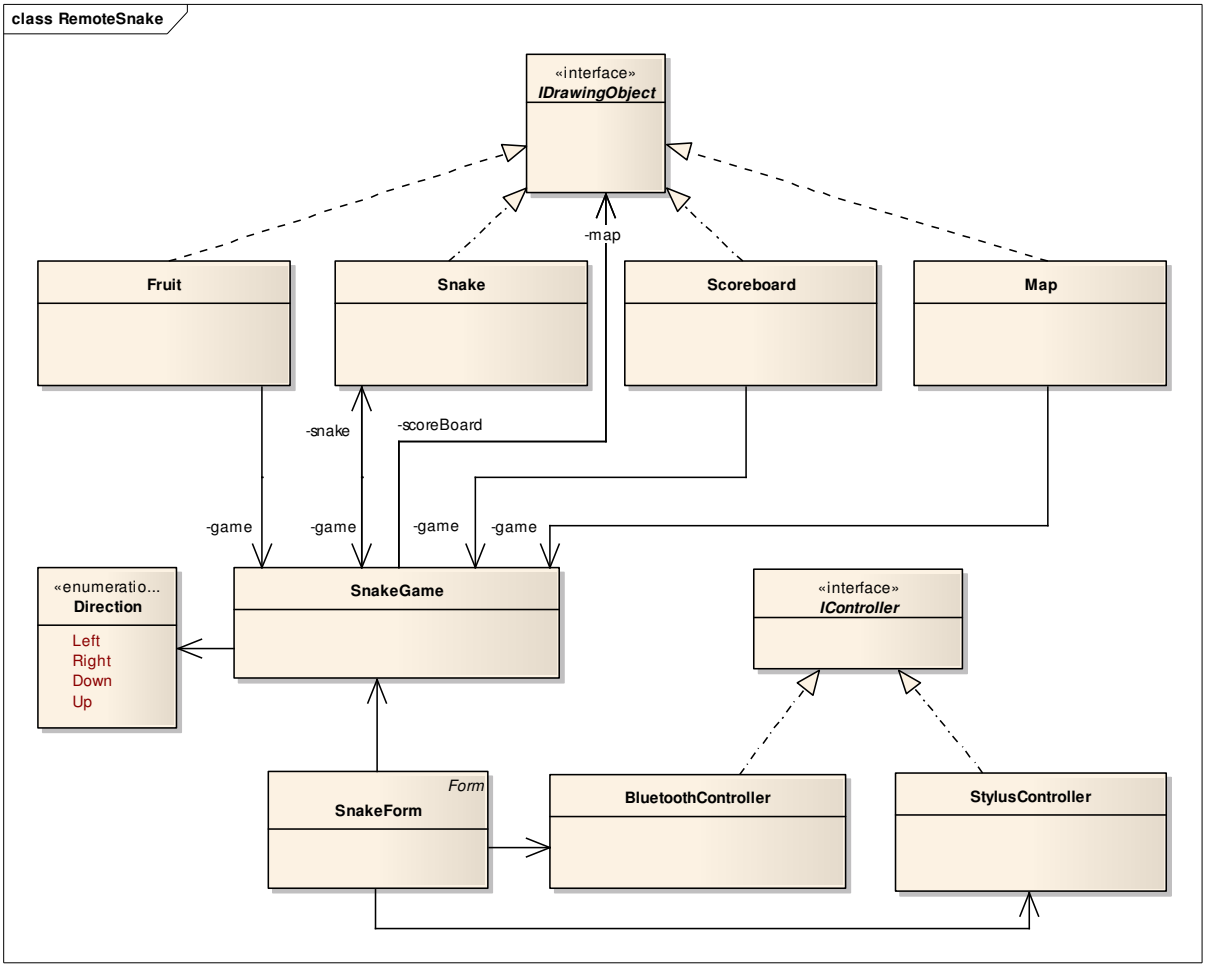
Figur 6. Sekvensdiagram for opstart og udvælgelse af touch kontroller

4.2 Windows Mobile (.NET Compact Framework)

Denne del vil beskrive software arkitekturen bag det udviklede *RemoteSnake* program. *RemoteSnake* er udviklet til Windows Mobile i .NET Compact Framework'et og er blevet testet på en HTC Touch Diamond.

Den overordnede ide med dette program er, at det skal udvikles som en separeret del fra .NET Micro Framework'et, som understøttede fjernstyring. Som nævnt i de foregående afsnit, foregår styringen via et Tahoe-II development board, hvortil et program er udviklet som kommunikerer og styrer *RemoteSnake* ved et wireless Bluetooth interface.

Nedenfor ses det endelige klassediagram for *'RemoteSnake'* og viser arkitekturen bag det udviklede program, jf. Figur 7.



Figur 7. Klassediagram for .NET Compact Framework klassen 'RemoteSnake'

Alle grafiske elementer tegnes på et *Graphics* objekt (System.Drawings) i *SnakeForm* der er implementeret som en *Form* (System.Windows.Forms). I denne *SnakeForm* initialiseres et nyt *SnakeGame* objekt som styrer spillet. Denne holder bl.a. styr på sværhedsgrad, hastighed og point.

Klassediagrammet viser interfacet *IDrawingObject*, som er de elementer der skal vises på *Graphics* objektet i vores *SnakeForm*.

Fire klasser implementerer dette interface, henholdsvis; *Map*, *Snake*, *Fruit* og *Scoreboard, jf.* Figur 8 *for en illustration af disse placeringer på skærmen*.

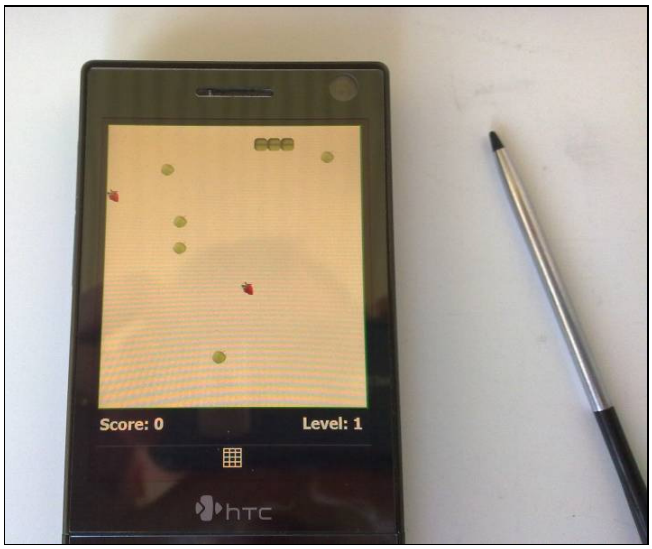
- *Map* er den bane som spillet foregår i
- *Snake* er den orm som bevæger sig rundt på banen
- *Fruit* er en samling af de frugter som kommer frem på banen og som ormen skal spise for at få point
- Samt *Scoreboard*, som er den pointtavle der vises i bunden af skærmen; hvorfra resultat og sværhedsgrad kan ses.

SnakeForm har en indbygget timer, som står for at repainte alle *IDrawingObject*'s med en passende frekvens. Denne frekvens er sat til 1000 divideret med spillets aktuelle hastighed, som er angivet i en property i SnakeGame kaldes Speed. Denne initialiseres til 1 og hvert 30. sekund ligges en til.
Dermed er frekvensen en funktion af hastigheden, hvilket gradvist får *Snaken* til at bevæge sig hurtigere.
Længden af *Snaken* er lig:

$$\frac{Fruits}{3}+1$$

hvor, fruits er antallet af opsamlede frugter.

SnakeForm starter desuden to *IController*, som sørger for styre Snaken. Henholdsvis; *BluetoothController* og *StylusController*.
StylusController implementerer styring på enhedens egen touch-skærm (hvilket normalvis gøres med en stylus-pen), mens *BluetoothController* kommunikerer med vores .NET Micro Framework som tidligere omtalt. Med denne opbygning er det nemt at tilføje yderligere kontrollere, hvis det senere skulle blive nødvendigt.



Figur 8. Afvikling af RemoteSnake på HTC Touch Diamond (Windows Mobile Professional 6.1)

Den instans af *SnakeForm* som forsøger at initialiserer *BluetoothController* klassen, vil ikke kunne initialiseres hvis vores FireFly Bluetooth adapter er udenfor rækkevidde. Da det var et ønske at spillet stadig kunne afvikles, selvom der ikke kan opnås forbindelse til fjernbetjeningen, er initialiseringen af *BluetoothController* indsat i en *Try-Catch* rutine. Dermed kan programmet blot styres med en stylus-pen på enhedens egen touch-skærm, hvis ikke fjernbetjeningen kan tilkobles.

4.3 Distribueret Kommunikation

Som tidligere beskrevet er formålet med dette projekt, at udvikle et system bestående af to programmer, der afvikles på hver deres platform.

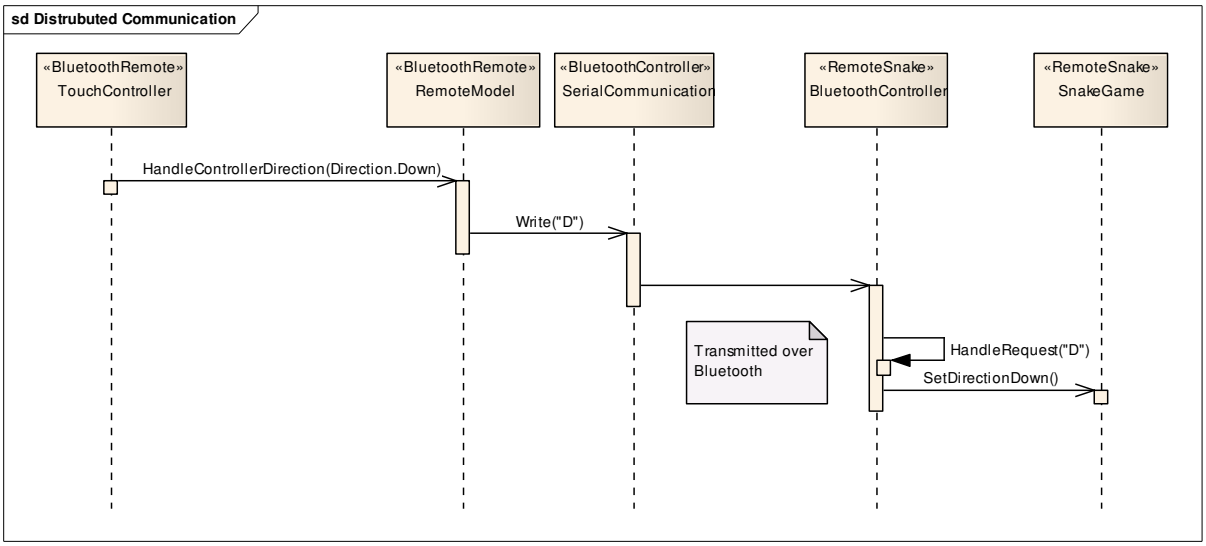
Kommunikationen mellem de to platforme foregår over et Bluetooth interface. Programmet *BluetoothRemote* har som tidligere nævnt mulighed for udnyttelse af tre type kontrollere; touch, motion og buttons.
Hver af disse kontrollere benytter *HandleControllerDirection* og *HandleControllerAction* metoderne i *RemoteModel*. Disse udsender via *SerialCommunications* driveren instruktion om retning/action til spilenheden.
I *SerialCommunication* constructoren oprettes en instans af SerialPort (System.IO.Ports) på "COM1". Handle-metoderne kalder *SerialCommunication.Write* med en string som input. Denne string konverteres først til et byte array hvortil der tilføjes et carriage return samt new line. Efterfølgende kaldes SerialPort.Write med det konverterede byte array som input.

Som programmet er implementeret nu, består input strengen kun af en enkelt karakter, og der afsendes en af følgende seks mulige karakterer:

- Left (L)
- Right (R)
- Up (U)
- Down (D)
- Start (S)
- Pause (P)

I constructoren til BluetoothController klassen på Windows Mobile enheden, oprettes ligeledes en instans af SerialPort (System.IO.Ports) på "COM1".
Efterfølgende oprettes en eventhandler som registrerer hvornår der modtages data på serielporten (*SerialDataReceivedEventHandler)*, dette event kalder metoden *SerialPort_DataReceived*.
Så længe der ikke modtages et newline på serielporten, bliver karakteren tilføjet et input array. Når et newline modtages bliver en delegate invoked med det omtalte input array. Denne delegate afvikler metoden *HandleRequest*, som vha. en switch behandler de input-karakterer der understøttes af programmet. Hver af disse kaldes ned i *SnakeGame*, hvor den ønskede operation udføres.

Nedenfor, jf. Figur 9, vises et sekvensdiagram for det tilfælde, hvor en ny *ControllerDirection* kommunikeres mellem de to programmer:



Figur 9. Sekvensdiagram for 'Down' event fra touch kontroller

TouchControlleren på *BluetoothRemote* programmet registrer at Snakens retning ønskes ændret til nedad. Denne instruktion sendes til *RemoteModel*, hvorefter den sendes videre til *RemoteSnake* som tidligere beskrevet.
Denne har som tidligere beskrevet sin *HandleRequestDelegate*, som sender signalet videre, for til sidst at kalde *SetDirectionDown* metoden i *SnakeGame* klassen. Når signalet registreres, vil Snaken bevæge sig nedad næste gang der udføres et *Repaint* på Graphics.

5 Resultater

Dette afsnit vil samle op på resultatet af udførte tests, samt belyse nogle af de forbedringsforslag deltagerne ser for systemet.

Systemets to platforme blev testet separat ved at benytte en Hyper Terminal til at kontrollere den korrekte respons. Altså blev det kontrolleret hvorvidt fjernbetjeningen udsendte de korrekte ASCII karakterer, mens det blev kontrolleret hvorvidt spillet tolkede de modtaget ASCII karakterer korrekt.

Efter udviklingen af den første prototype afprøvede vi systemet på to testpersoner; En med, samt en uden teknisk baggrund. Personerne blev instrueret i hvilke funktioner der lå i systemet, samt hvordan de forskellige styringer til Snake spillet fungerede. I den forbindelse blev det valgt at udfærdige en brugervejledning for systemet.
Generelt var testpersonerne enige, og var godt tilfredse med brugervenligheden, både på det visuelle design af Snake spillet samt præsentationen af valgt retning på fjernbetjeningen. Derudover havde de følgende kommentere til systemet:

- Der manglede en funktionalitet til at sætte spillet på pause.
- Hvis touch og motion styring blev valgt, kunne man få Snaken til at bevæge sig skævt, ved eksempelvis at tilte fjernbetjeningen imod højre og nedad på samme tid.

Dette blev efterfølgende løst ved at implementerer en pause funktion hvis styring gennem buttons blev valgt. Derimod blev det vurderet at stride imod princippet, hvis touch og motion ligeledes kunne sættes på pause ved at benytte samme tast, hvorfor en funktionalitet herfor ikke blev implementeret.
Til at korrigerer imod at Snaken kunne gå skævt, blev der implementeret et tjek for i hvilken retning enten touch bevægelsen eller tiltningen af boardet var størst. Derved ville det blive denne retning som blev valgt.

5.1 Forslag til forbedringer

At implementere en funktionalitet, som tillader at spillet sættes på pause når touch eller motion styring vælges. Dette kunne eksempelvis være en "knap" på skærmen i tilfælde af touch er valgt, og tolkning af et ryst i z-retning for motion styring.

Mulighed for at gå tilbage til hovedmenuen og vælge en anden form for styring.

Implementering af en funktionalitet som ville gøre det muligt at afspille en lyd når frugter indsamles, samt i det tilfælde at brugeren får beskeden Game Over.

6 Konklusion

Der er igennem rapporten blevet redegjort for opbygningen af systemet, både hvad angår hardware og software.

Til hardware interfacet blev en FireFly Bluetooth adapter valgt som kommunikation mellem .NET Micro Framework'et og .NET Compact Framework'et. Til softwaren blev styring af en FireFly adapter samt et Snake spil udviklet.
Dette spil udmærker sig ved at kunne styres både på enhedens egen touch-skærm, samt med den udviklede fjernbetjening.

Fra start var det interessant at undersøge, hvorvidt kommunikationen imellem de to enheder foregik hurtigt nok, således at spillet reagerede fornuftigt. Styringen blev implementeret således at spillet reagerer med det samme, både når fjernbetjeningen samt enhedens egen touch-skærm benyttes.
Til BluetoothRemote programmet er der brugt model-view-controller design pattern, hvilket gør det meget fleksibelt. Bl.a. betyder dette, at yderligere kontrollere nemt kan tilføjes. Ydermere er der lavet en menu ved opstart af Tahoe-II kortet, hvorfra man kan afgøre hvilken kontroller man ønsker at benytte, henholdsvis; knapper, touch eller motion.

Generelt er der udviklet et fornuftigt system, som opfylder de gennem rapporten fremstillede krav – endda med yderligere funktionalitet. Bl.a. var det fra start ikke tiltænkte med en menu til udvælgelse af kontroller.

7 Appendiks

Appendiks er opdelt i to kategorier

- Vedlagt rapporten (samt CD)
- Vedlagt på CD

Vedlagt rapporten

Appendiks 1 – Kildekode for BluetoothRemote samt RemoteSnake
Appendiks 2 – Brugervejledning til Remote m. Bluetooth Interface

Vedlagt på CD

Appendiks 3 – FireFly BT Userguide