Projekt Rapport

Windows Embedded Mobile

Rasmus Larsen, Thomas Vogel & Harald Nielsen

GODKENDELSESFORMULAR

Ved underskrivelse af dette dokument accepteres det af begge parter, som værende projekt rapporten til det ønskede system.

Sted og dato:

_____

_____

Michael Alrøe

Sted og dato:

_____

_____

06701 Thomas Vogel

_____

06082 Rasmus Anegaard Larsen

_____

06505 Harald Kjær Nielsen

1. RESUME

The focus of this project is to create a user friendly program for remote controlling a model car [Modelbil]. The program(s) will be fully functional, having all the functions needed to create a Bluetooth connection between the remote control (a mobile phone) and the car (model car, Tahoe II [Tahoe] board and a Bluetooth device). The connection is used to send information about the user's choices about throttle and direction to the Tahoe II board which then adjusts the PWM signal of the motors accordingly.

The system is divided into two programs:

CarControl:

Runs on the Windows Compact Framework 3.5 [Compact] on a HTC Diamond [Diamond]. This program is designed with 2 tier layers, due to the lack of persistent data, there is only the GUI layer and the BL layer, the Franson Bluetools Library takes the place as the DAL layer. The GUI layer uses functionality in the BL layer to search for the Bluetooth device and its service, for connecting to that service and letting the user control the throttle and direction. The BL layer contains all the logic behind the user event. Such as establishing the Bluetooth connection, calculations behind the user's input of throttle and direction and finally transmitting those actions to the model car.

RemoteCar:

Runs on the Windows Micro Framework 3.0 [Micro] on a Tahoe II Board. This program is designed after a 3 tier layer with the GUI layer as a presentation layer only. It takes the role of a GUI that shows what PWM values that are sent from the mobile client. The BL layer handles the data that comes from the mobile client, and if it's a PWM value, sets it. The DAL Layer contains the logic that handles the Bluetooth connection and the PWM timing.

2. INDHOLDSFORTEGNELSE

Indhold

Figuroversigt

3. DESIGN

3.1.1 Baggrund for udvikling af programmet

Der ønskes udviklet et fjernstyringssystem til en modelbil. Fra en mobiltelefon skal man genem en grafiskbruggrænseflade kunne fjernstyre modelbilens fart samt retning. Dette skal ske via trådløs kommunikation i form af bluetooth.

3.1.2 Fokusområder

I og med der er tale om en første version af programmet, har fokus været på følgende områder under udviklingen for at på bedst mulig måde at tilfredsstille både kunde(red. Michael Alrøe) samt bruger.

- Fuld implementation af de essientelle funktionaliteter
- Brugervenlighed
- Simplicitet
- Programstrukturering med henblik på videreudvikling

3.1.3 Systemets formål

Det overordnede formål med RemoteSystemet er at brugeren nemt og intuitivt ved hjælp af sin mobiltelefon kan søge efter modelbilens bluetoothenhed for derefter at koble sig på dens service. Derefter kan brugeren via et simpelt kontrolinterface let styre modelbilens fart samt retning så længe brugeren (med mobiltelefonen) er inden for bluetooth rækkevidde.

Figur 1: Systemets indbyrdes kommunikation

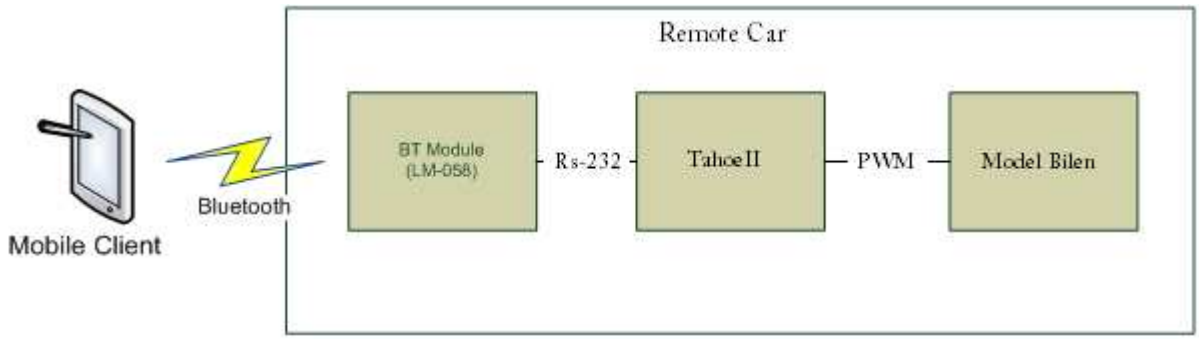
3.1.4 System introduktion

Vores system er bygget op af 2 applikationer.

Først og fremmest er der deployet en .NET Micro Framework løsning til et Tahoe II board. Til dette board er der tilkoblet en FireFly bluetooth enhed via serielporten. Ydermere er Tahoe II boardet koblet til modelbilens 2 servomotorer således at hastigheden kan justeres.

Derudover er der udviklet en .NET Compact Framework løsning til en hvilken somhelst mobiltelefon som understøtter Windows Mobile 6.0 eller over samt har bluetooth. I dette tilfælde en HTC Touch Diamond.

Når applikationen er startet på telefonen kan brugeren ved hjælp af en grafisk brugergrænseflade på telefonen let tilslutte sig til Tahoe II boardets bluetooth service. Derefter åbnes et nyt vindue hvor i brugeren har mulighed for at ændre på hastigheden samt retningen på modelbilen. Disse ændringer sendes derefter ved hjælp af den trådløseforbindelse til modelbilens 2 motorere, så de kan imødekomme brugerens valg.



Figur 2: Oversigt af systemet

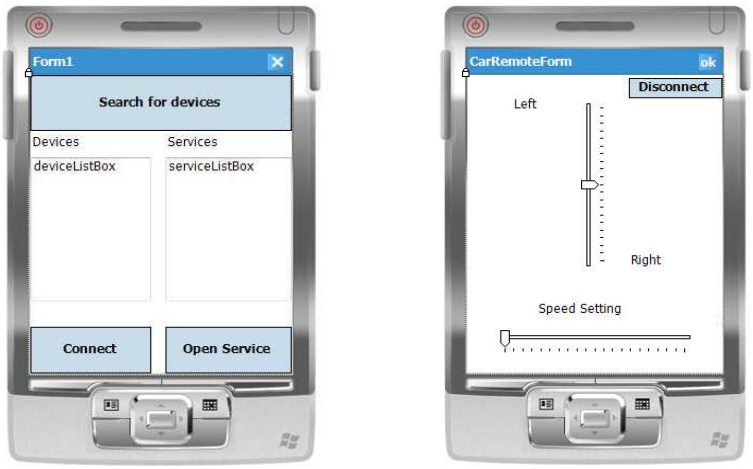
ControlCar's hovedfunktionaliteter er følgende:

- Søge efter bluetooth enhed.
  - Dette sker ved tryk på Search for devices knappen på figur 3's venstre billede.

- Opret forbindelse til Remote Car, over bluetooth.
  - Der forbindes til den ønskede bluetooth enhed ved at vælge den fra den venstre liste, herefter trykkes Connect. Derefter vælges den ønskede service fra den højre liste og der trykkes Open Service. Det åbner vinduet som ses på figur 3' højre billede.

- Styring af hastighed.
  - Her kan hastigheden styres ved at trække slideren i bunden fra den ene side til den anden

- Styring af retning.
  - Retningen kan kontrolleres ved at trække slideren i toppen op eller ned. Når slideren befinder sig i midten køres der ligeud. Bevæges den helt til toppen drejes mod venstre hvorimod hvis den bevæges til imod bunden drejes der til højre.

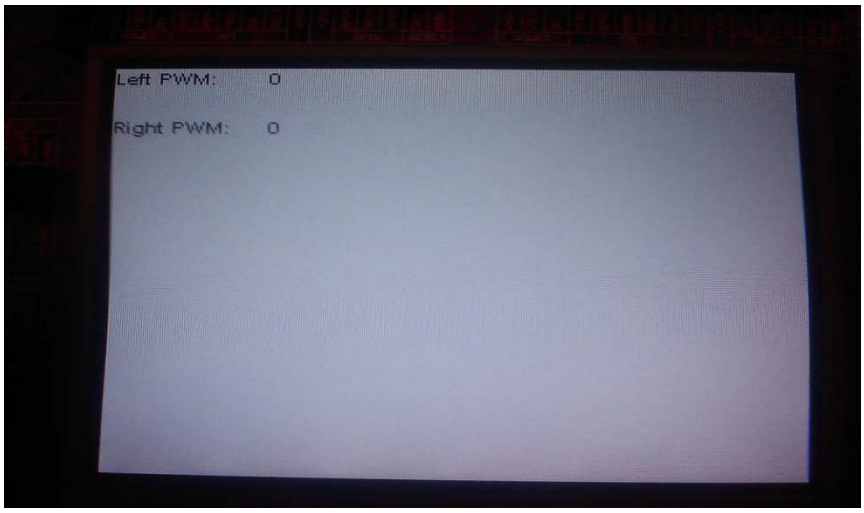


Figur 3: Oversigt over ControlCar applikationen

RemoteCar's hovedfunktionaliteter er følgende:

- Modtage forbindelsen af Klienten.
- Parse dataen der kommer fra Klienten.
- Styring af PWM signalet til Modelbilen.

- Udskrivning af PWM signaler til LCD display.



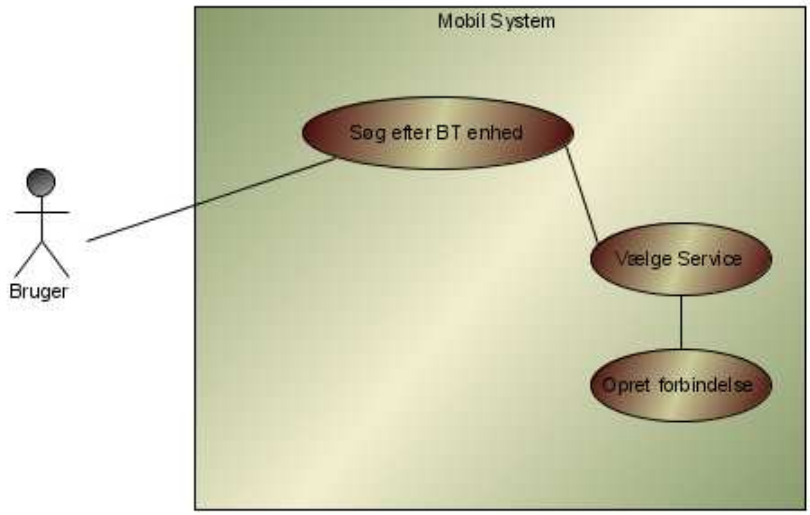
Figur 4: Nuværende PWM signaler udskrevet på Tahoe II boardet

3.1.5 Bruger scenarioer

Et typisk bruger scenario vil se således ud, vist i diagrammer:

A) Find og Forbind

- Brugeren trykker på Search
- Brugeren vælger den korrekte bluetooth enhed og trykker på Connect
- Brugeren vælger den korrekte bluetooth service og trykker Open Service
    - En ny side åbnes på mobiltelefonen til kontrol af fart og retning

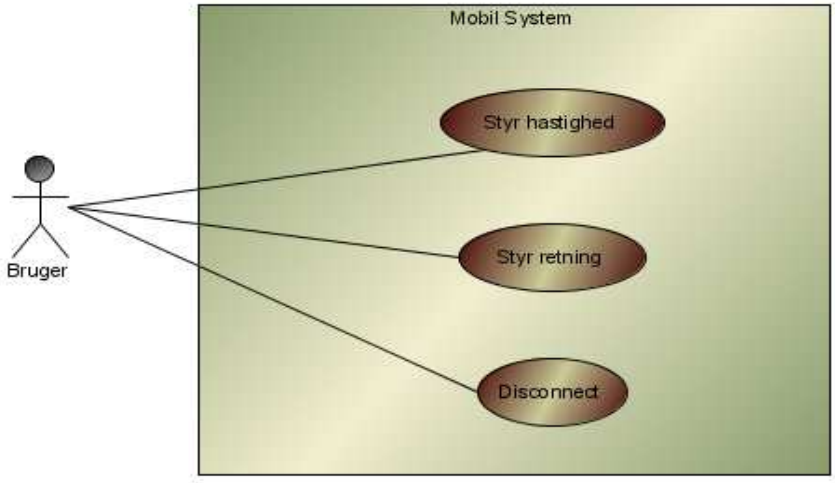


Figur 5: ControlCar bruger scenarioer del 1

For at styring af Remote Car skal være tilgængelig for brugeren skal scenario A være opfyldt:

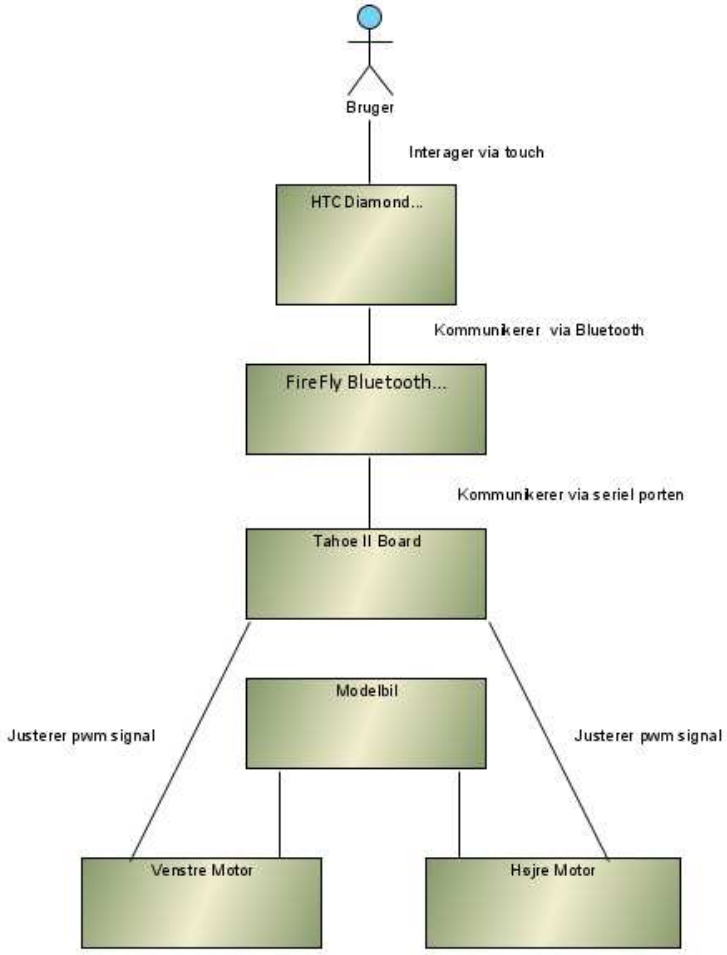
B) Styr fart og retning

- Brugeren trækker fart slideren, bilen justeres til den nye hastighed
- Brugeren trækker i retning slideren, bilen justeres til den nye retning
- Brugeren trykker på Disconnect – Der returneres til index siden



Figur 6: ControlCar bruger scenarioer del 2

3.1.6 System oversigt



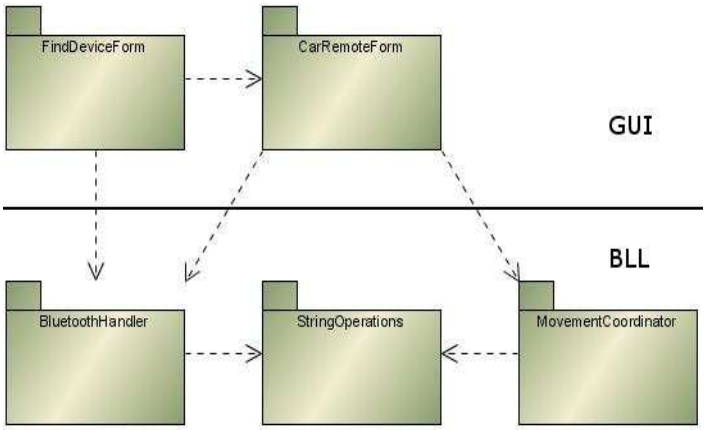
Figur 7: Domænemodel af systemet

Figur 6 viser domænemodellen, som er en illustrativ afbildning af systemets koncept. Domænemodellen er udarbejdet for at komme i gang med selve den designmæssige del af udviklingen, men primært, for at give overblik over systemet. Den er blevet brugt til at finde softwareklasser og til at give input til Use Case analyse.

Ordene som står imellem de forskellige domæner, forklarer aktiviteten mellem disse.

3.1.7 Lagdeling

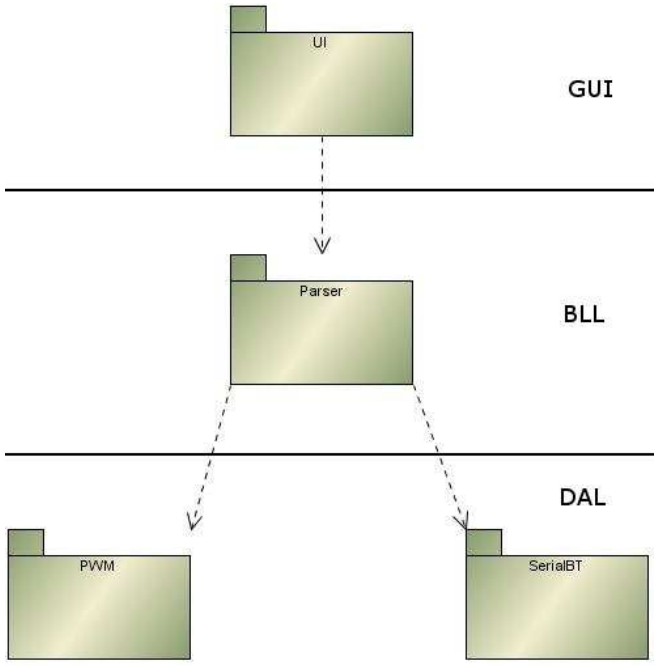
CarControl



Figur 8: Lagdelt diagram for CarControl

FindDeviceForm benytter sig af BluetoothHandler til at finde og oprette forbindelse til et Bluetooth device. Der kaldes kun ned til BluetoothHandleren. For at få informationer tilbage op til Formen, subscriber den til forskellige events i BluetoothHandleren.
FindDeviceForm kan åbne CarRemoteForm når der er blevet oprettet en Bluetooth forbindelse.
CarRemoteForm benytter MovementCoordinator til at udregne PWM signaler,
MovementCoordinator benytter BluetoothHandler til at sende PWM til en Bluetooth enhed.
Både BluetoothHandler og MovementCoordinator benytter StringOperations for at sørge for at teksten bliver konverteret rigtigt til noget man kan sende og tilbage igen.

RemoteCar

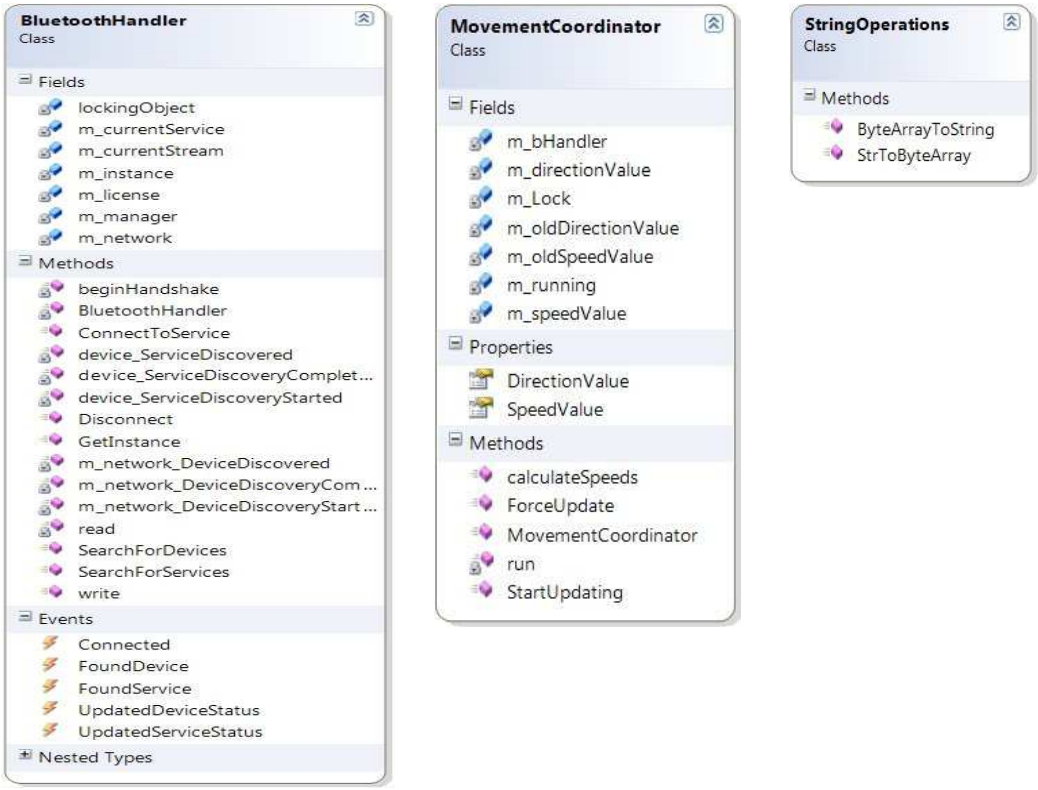


Figur 9: Lagdelt diagram for Remote Car

Klasserne i RemoteCar er blevet inddelt efter den klassiske 3 lags arkitektur, bestående af et User interface lag (UI), et applikation logik lag (BLL) samt et data lag (DAL). Ved at opbygge programmet efter denne struktur opnås en adskillelse af det grafiske interface og data.
UI laget viser de data der PWM objekterne er sat til. Parser klassen sørger for at parse det data som kommer fra SerialBT, og sætte disse værdier i PWM. PWM klassen sætter PWM signalet til de tilhørende pins. Det skal bemærkes at PWM er en aktiv klasse.

3.1.8 Klassebeskrivelser

3.1.8.1 CarControl klasser



Figur 10: Klasserne BluetoothHandler, MovementCoordinator og StringOperations

BluetoothHandler:

BluetoothHandler benytter sig af Franson bibliotekerne til at benytte bluetooth kommunikation på en Compact Framework enhed.

BluetoothHandler er en singleton klasse da der kun må være en enkelt instans, med mutual exclusion der har fat i selve bluetooth funktionaliteten.

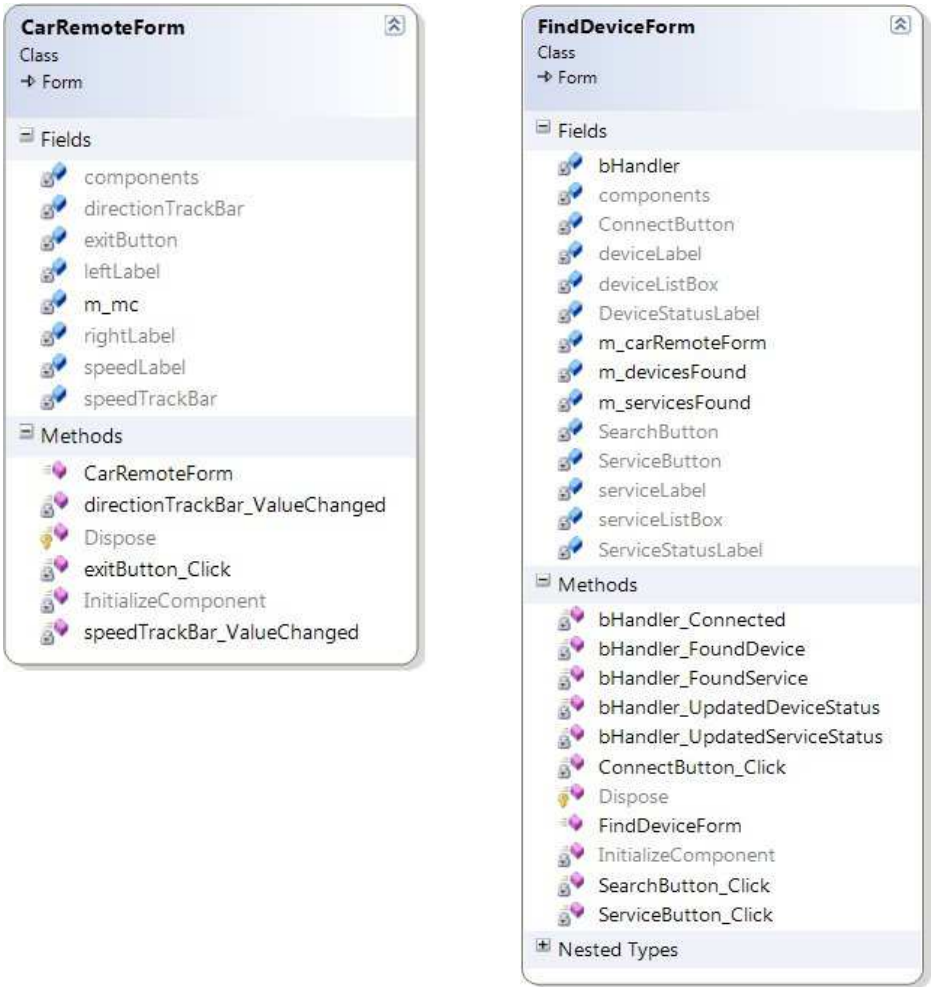
Da bluetooth forbindelser bliver kompliceret når man ikke benytter sig af pairing, er dette en stor klasse.

For at kunne finde en bil ud af mange enheder søger man først efter bluetooth devices. Efter disse er fundet kan man finde services på et device. For at sikre sig at det er en RemoteCar er der et

handshake bruges efter at en forbindelse er blevet oprettet til en service. Hvis der ikke bliver svare korrekt tilbage på handshaket vil forbindelsen blive lukket igen.
Udover at håndtere oprettelsen af forbindelsen, benyttes BluetoothHandler også til at skrive og læse fra Bluetooth forbindelsen.

StringOperations:
Dette er den mindste klasse, dens funktioner er statiske, så det er ikke nødvendigt at oprette en instans af den.
Funktionerne i denne klasse sørger for at encoding på tekst, når det bliver konverteret til et byte array til at sendes, eller til tekst fra et modtaget byte array, bliver korrekt fortolket.

MovementCoordinator:
Denne klasse står for at kører en tråd, når programmet er i remote delen. Værdier bliver sat af RemoteCar klassen hvor der så udregnes 2 PWM værdier der er et procentvis udtryk for hastigheden bilen skal kører.
Forskellen i de 2 PWM signaler bestemmer hvilken vej bilen drejer. Motoren med det højeste PWM signal vil kører hurtigere og dermed tvinge bilen til dreje i retning af det andet hjul.
Hvis indstillingerne for kørselen hele tiden bliver ændret vil der bliver sendt 4 gange i sekundet. Hvis der ikke sker nogle ændringer vil der alligevel blive sendt PWM data af sted hvert sekund, for at være sikker på at der bliver holdt liv i forbindelsen. Dette sker igennem instansen af BluetoothHandler



Figur 11: Klasserne CarRemoteForm samt FindDeviceForm

CarRemoteForm:

Denne Form er bygget op at 2 scrollbars og en disconnect knap.

Når værdien på en af scrollbarene bliver ændret, bliver den nye værdi sendt til MovementCoordinatoren der opdatere PWM signalet efter behov.

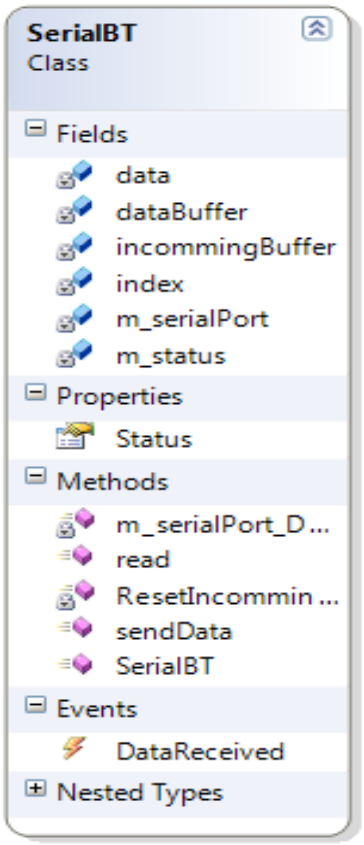
Disconnect stopper PWM signalet og lukker programmet

FindDevicesForm:

FindDevicesForm består af en search knap der benytter BluetoothHandler til at finde Bluetooth devices i nærheden.

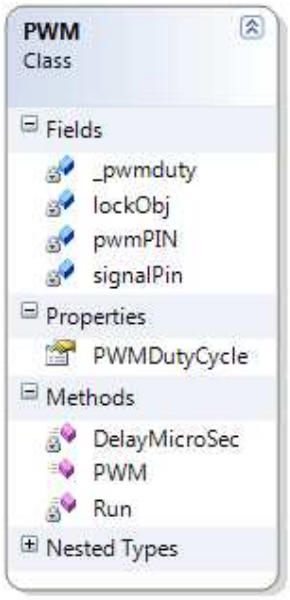
Efter devices er blevet fundet, er det muligt via connect knappen at finde services på selve devicet. Man kan derefter forbinde til servicen, hvis forbindelsen lykkedes vil der blive åbnet et CarRemote vindue.

3.1.8.2 RemoteCar klasser



Figur 12: Klasse diagram over SerialBT

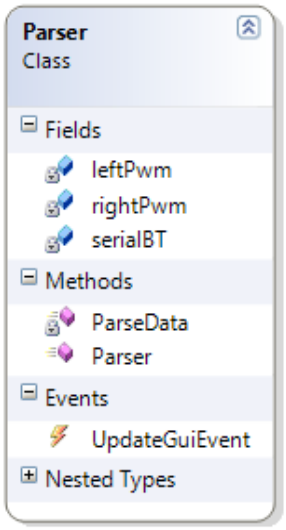
SerialBT sørger for al kommunikation over serialporten til bluetooth modulet. Som der ses er der et DataRecived event, som triggers når incommingBuffer er fyldt. Parseren fra BLL abonnere på denne event. På denne måde overholder vi den lagdelte struktur og vi spare CPU tid, da vi ellers skulle polle efter data i en tråd.



Figur 13: Klassediagram over PWM

PWM klassen sørger for timingen til PWM signalet og styringen af pin's. Der findes et bibliotek til dette i API'et til Tahoe II kortet. Men i det er der kun mulighed for at sætte et PWM signal til de to PWM pins. Den måde det er løst på er ved hjælp af et delay, som stopper tråden i de antal microsekunder der skulle til for at styre dutycyclen korrekt. For at opnå dette toggles GPIO pins manuelt.

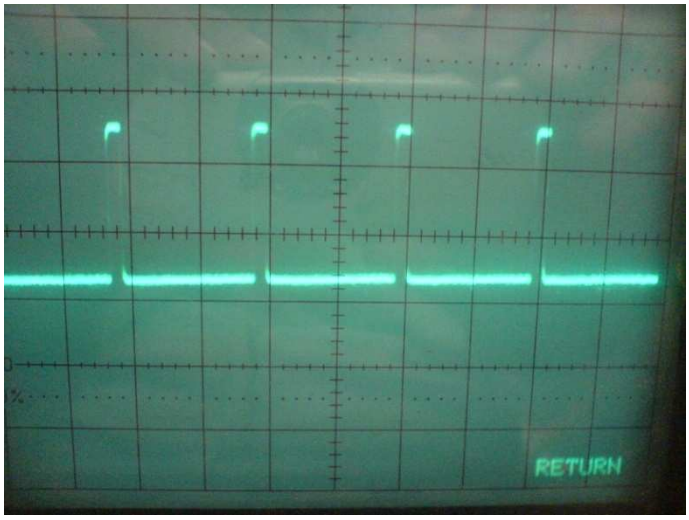
For at spare CPU tid bruges Thread.Sleep til "hele" millisekunder da måden at lave delay i microsekunder er meget CPU krævende.



Figur 14: Klassediagram for Parser

Parser er klassen hvor alt logikken for RemoteCar ligger. Den håndterer det data der kommer fra SerialBT og svare tilbage (Handshake) eller sætter PWM signalet. Her er der igen et event, UpdataGUIEvent. Dette event bliver abonneret på af UI, da der ellers skulle polles efter data. Der opnåes derved at UI bliver opdateret så snart en PWM værdi bliver sat. Dette er vigtigt, især da der kun er en 100mHz CPU som bliver belastet af PWM trådene.

4. KONKLUSION



Figur 15: PWM signal overvåget via oscilloskop

Fokus igennem forløbet har været at implementere så meget som mulig, og ud fra dette er alt der var planlagt blevet implementeret.
Dog er vi på grund af et uheldigt tab af modelbilen blevet nødsaget til kun at teste de dynamiske pwm signalerne rent visuelt, i stedet for at kunne se bilen fysisk ændre retning samt fart. Dog erfarer vi efter ved en testopstilling hvor Tahoe II boardet er tilsluttet et oscilloskop for at monitorere de udgående pwm signaler, at variablerne opdateres korrekt efter brugerinput. Det ses afbilledet på figur 15 hvor dutycycles ses hvert 20 000 mikrosekund / 50 Hertz.
En ting vi gerne ville have implementeret var mouseDown og mouseUp events på de 2 sliders i CarControl applikationen i stedet for valueChanged. Derved kunne der spares væk en del unødig datatrafik mellem de 2 applikationer. I stedet ville man kun få heartbeat overførelser minimum hvert sekund ved uændret tilstand eller når brugeren har sluppet den pågældende slider hvor han ønsker den placeret. Derved udgås alle mellem – events.
Igennem projekt forløbet er der blevet brugt en stor mængde tid på at få etableret en Bluetooth forbindelse imellem HTC Touch Diamonden og Tahoe II boardet via LM-058 Bluetooth til RS232 modulet. Når LM-058 modulet er tilsluttet en PC er der ingen problemer med at kommunikere over

den, men så snart at det blev tilsluttet Tahoe II boardet, fungerer kommunikationen kun i den ene retning. Dette har under hele forløbet været et kæmpe problem og været skyld i forsinkelser.
Først på den sidste dag, har vi lånt en Bluetooth enhed der fungere på næsten samme måde, en FireFly. Straks efter tilslutning og ændring af Serielport opsætning (Til at akkommodere en 9600 baudrate) har der ingen problemer været. Som resultat kan vi konkludere at LM-058 enheden enten er defekt eller ikke kan fungere på et Tahoe II board.
Af andre problemer kan det nævnes at det var meningen at RemoteCar programmet skulle benytte sig at keepalive PWM signalet til at finde ud af om den stadigt har Bluetooth forbindelse. Hvis der ikke kom et PWM signal inden for en kort tidsperiode, skulle bilen så antage at der ikke længere var forbindelse og stoppe bilen. Dette har vist sig ikke på nuværende tidspunkt at være muligt, da Tahoe II boardet ikke har et internt ur, og virkede tilfældigt hvilken data tid datatime.now returnerede.

4.1 Kvalitet

I dette afsnit beskrives hvorledes arkitekturen opfylder fokusområderne tidligere beskrevet.

Funktionalitet:

Systemet opfylder de system specificerede krav som der var opstillet for projektet. Der er fuld understøttelse for trådløs kobling mellem de 2 applikationer hvorved der kan sendes bruger input hvert 250 millisekund ved bruger aktivtet eller hvert 1000 millisekund som passiv forbindelsebekræftelse. Ud fra brugerens ønske om hastighed samt retning som udtrykkes vhj. de 2 sliders beregnes der en dutycycle til hver servomotor, som efterfølgende sendes via bluetooth til Tahoe II boardet der skulle sende det videre til modelbilen. Der er dog yderligere valgt at udskrive pwm variablerne på Tahoe II boardets display.

Brugervenlighed:

CarControl er simpelt opbygget med 3 knapper og 2 listboxe som gør det let for brugeren at tilkoble sig den bluetooth enhed samt service som RemoteCar applikationen udbyder. Det kræver dog at brugeren kender navnet på enheden (Den udbyder kun 1 service). Herefter er selve styringen tilgængelig for brugeren via 2 sliding bars som styrer henholdvis fart og retning. For at lette at finde punktet hvor bilen kører ligeud er der lavet et lille interval omkring det neutralepunkt hvor slideren automatisk ”snapper” ind i neutral.

Udvidelsesmuligheder:

Vi har bestræbt os på at lavet arkitekturen således at GUI & BLL lagene er så adskilt fra hinanden som muligt. Sådan at informationerne kun eksisterer i BLL laget og at GUI laget udelukkende står for det visuelle. Derved er der skabet et grundlag, for at senere videreudvikling eller ændringer ville kunne foregå på det visuelle lag uden at påvirke funktionaliteten i BLL laget. I BLL laget kan man desuden tilføje ny funktionalitet uden det påvirker interface laget.

5. TEKNISKE INFORMATIONER

5.1 Implementeringssprog og værktøjer

Projektet er implementeret i C# [WF] og udviklet i Visual Studio 2008.

De 2 forskellige løsninger henholdvis CarControl (HTC Diamond) og RemoteCar (Tahoe II) er udviklet på forskellige framework som understøtter det pågældende embeddede enhed:

CarControl: Windows Compact Framework 3.5

RemoteCar: Windows Micro Framework 3.0

5.2 Oversættelses-software

Programmet til .NET Microframework bliver afviklet uden styresystem, og bliver dermed kompileret sammen men funktionalitet til at afvikle programmet på den embedded enhed.

5.3 Installation

CarControl programmet skal overføres til HTC Diamonden via et USB-Mini kabel.

RemoteCar programmet skal deployes til Tahoe II boardet via et USB-Mini kabel.

5.4 Kørsel

5.4.1 Kørsels-hardware

For at systemet skal kunne afvikles som tiltænkt, skal følgende opstilling være til stede:

- En HTC Diamond Touch hvor CarControl er installeret på.
- Et Tahoe II Board med seriel tilsluttet FireFly Bluetooth enhed hvor RemoteCar er installeret på.
- Strømforsyning til FireFly enheden
- En modelbil med 2 hjul med hver deres servomotorer.
- Strømedninger fra hver af de 2 motorere til Tahoe II Boardet.

5.4.2 Kørsels-software

Windows Mobile 6.0 Professional på mobil enheden, ellers kan CarController applikationen ikke eksekveres.

6. DEFINITIONER, ORDLISTE OG REFERENCER

6.1.1 Definitioner

RemoteSystem

- En samlet betegnelse for hele projektet

CarControl

- Windows Forms programmet som kører på HTC Touch Diamonden via .NET Compact Frameworket 3.5

RemoteCar

- Windows Forms programmet som kører på Tahoe II Boardet via .NET Micro Frameworket 3.0

Modelbil

- Bilen der skal fjernstyres. Bilen består af 2 hjul samt hvor deres tilhørende servomotor. Ydermere forefindes Tahoe boardet med dets bluetooth enhed på bilen.

6.1.2 Ordliste

HTC Touch Diamond [Diamond]

- En Pocket PC med Windows Mobile 6.1

Tahoe II Development Board [Tahoe]

- En udviklingsplatform med et 3.5” LCD touch display. Leveres med .Net Micro Frameworket samt en række andre features

.NET Micro Framework 3.5 [Micro]

- En .NET[.NET] platform for enheder med stærkt begrænsede rescourcer. Support for udvikling i C# samt en GUI framework baseret løst på Windows Presentation Foundation [WPF]

.NET Compact Framework 3.0 [Compact]

- En .NET platform som er designet til at køre mobile og embeddede enheder som PDAs, mobiltelefoner etc. En nedskaleret version af det originale .NET framework.

6.1.3 Referencer

Litteratur:

[Larman]: Larman, Craig: Applying UML and Patterns 3Ed, Massachusetts USA 2007

Websites:

[Tahoe]: http://devicesolutions.net/Products/TahoeII.aspx

[Diamond]: http://en.wikipedia.org/wiki/HTC_Touch_Diamond

[.NET Micro Framework]: http://en.wikipedia.org/wiki/.NET_Micro_Framework

[.NET Compact Framework]: http://en.wikipedia.org/wiki/.NET_Compact_Framework

[.NET]: http://en.wikipedia.org/wiki/.NET_Framework

[WPF]: http://en.wikipedia.org/wiki/Windows_Presentation_Foundation