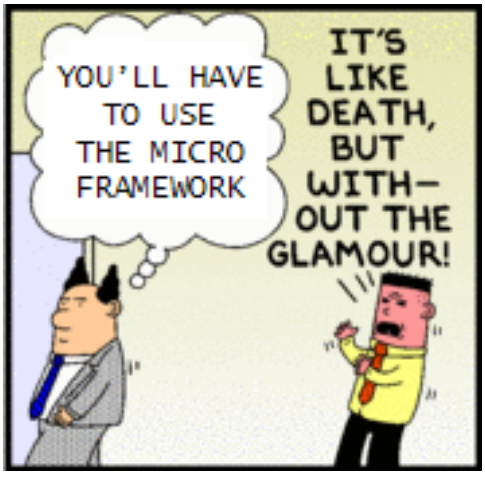
Rock Tahoe

ITWEM1 Projektrapport



Rock Tahoe

4. Juni 2009

Claus Andersen (20086025)
Ulrik Hjul Andersen (04289)

RESUME

Det følgende er et resume af det tekniske arbejde i forbindelse med afsluttende projekt i faget ITWEM1 ved IKT afdelingen på ingeniørhøjskolen i Århus.

Problemstilling / Formål

Ønsket med Rock Tahoe projektet er at udvikle en applikation med Microsofts .NET Micro framework til afvikling på DeviceSolutions development kit; Tahoe II.

Applikationen skal kunne hente lyd via netværk, spille musik/lyde ud fra accellerometer genkendelse, samt vise informationer som Ur, temperatur og lignende.

Resultater

Arbejdet med .NET micro frameworket viste sig noget mere besværligt og tidsslugende end forventet hvilket medførte at det meste af udviklingstiden gik med at deploye applikationen, opdatere firmwaren og SDKet og lignende. Dette har medført at det endelige resultat ikke på alle punkter opfylder de oprindelige krav til projektet.

Ulrik Hjul Andersen (04289)

Claus Andersen (20086025)

INDHOLDSFORTEGNELSE

RESUME **1**

INDHOLDSFORTEGNELSE **2**

1. INDLEDNING / PROJEKTBESKRIVELSE **3**

2. RESULTATET **4**

2.1. GENNEMGANG AF ROCK TAHOE 4

2.1.1. Graphical User Interface (GUI) *5*

2.1.2. Kommunikation med netradio server *7*

2.1.3. Afspilning af lyd *9*

2.1.4. Touch *9*

2.1.5. Temperaturføler *10*

2.1.6. Ur *10*

2.2. GENERELT OM .NET MICRO FRAMEWORK 10

3. KONKLUSION **12**

4. BILAG **13**

1. INDLEDNING / PROJEKTBESKRIVELSE

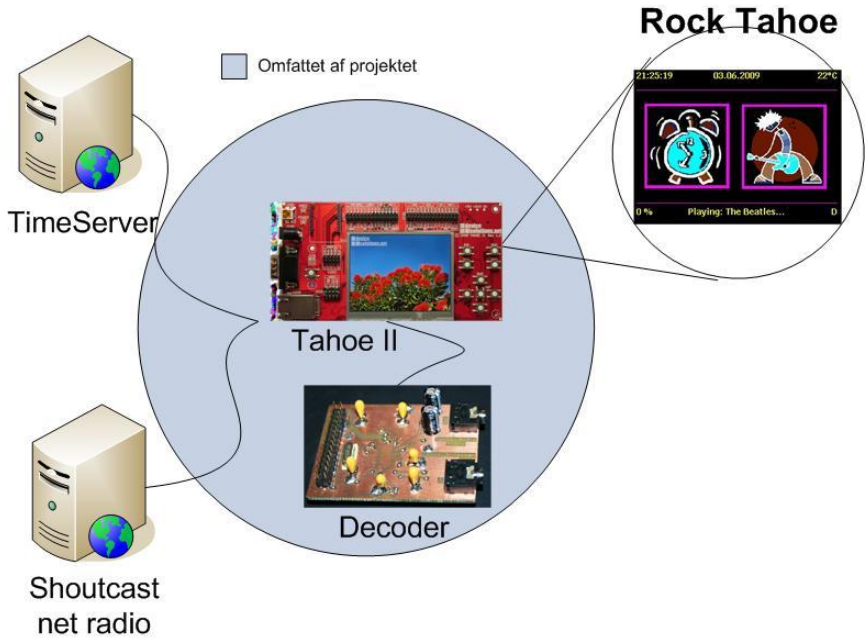
Rock Tahoe er en gadget, der tilbyder små funktionaliteter, som er smarte og sjove. Formålet med denne gadget er primært at undersøge og arbejde med de forskellige hardware områder, som Tahoe II development kit tilbyder til direkte brug med .NET Micro frameworket.

De primære funktioner i Rock Tahoe er:

- Vækkeur/clock der henter lyd via internetradio eller over ethernet fra en server
- Musikgadget/Tahoe-rockblomst der spiller rytmer/lyde udfra accelerometer genkendelse.
- Infostand der viser ur/temperatur.

For at ovenstående funktioner skal fungere kommer vi til at arbejde med følgende;

- Afspilning af lyd
- Kommunikation over ethernet
- Accelerometer
- Touch
- Temperaturføler
- Ur (opdatering via NTP, el. anden timeserver)

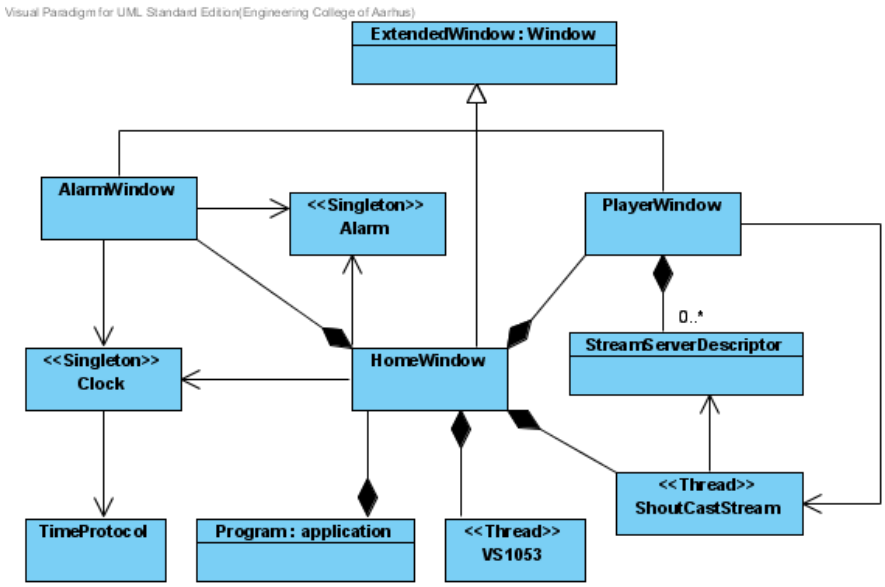


Figur 1 Systemoversigtsdiagram

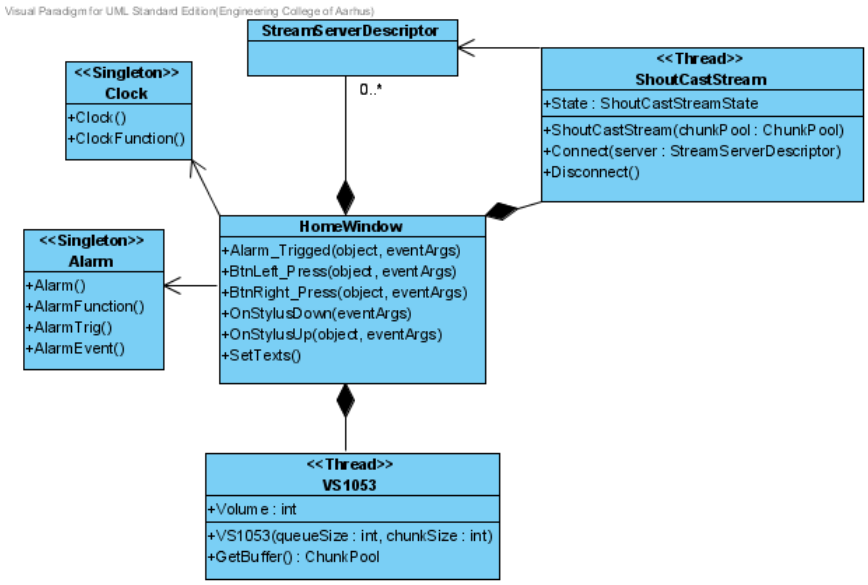
2. RESULTATET

2.1. Gennemgang af Rock Tahoe

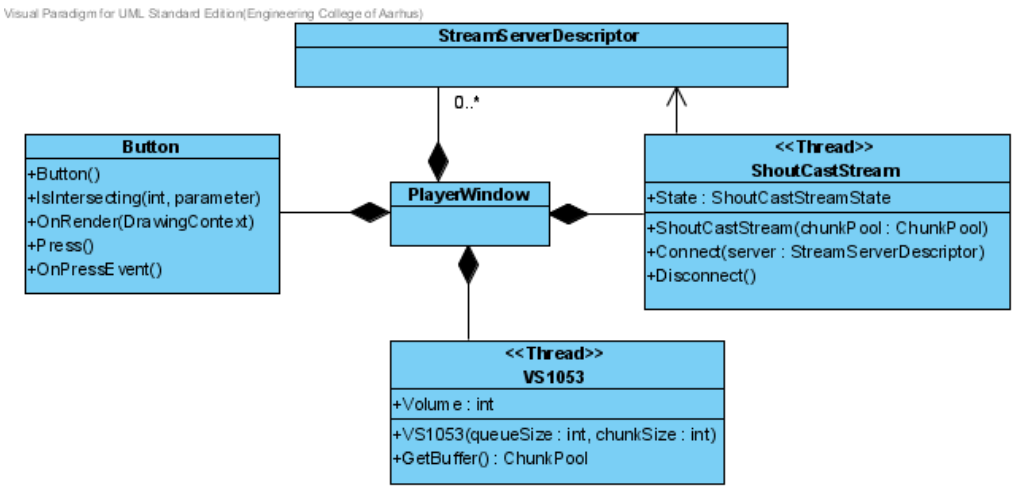
Følgende er et arkitektur view af Rock Tahoe. Først vises et samlet klasse diagram med de vigtigste klasser, og efterfølgende vises enkelte dele med flere detaljer. De mere detaljerede klassediagrammer viser områder fra overviewet, hvor yderligere detaljer er skønnet vigtige for forståelsen af det samlede system. Efter klassediagrammerne beskrives enkelte dele af projektet.



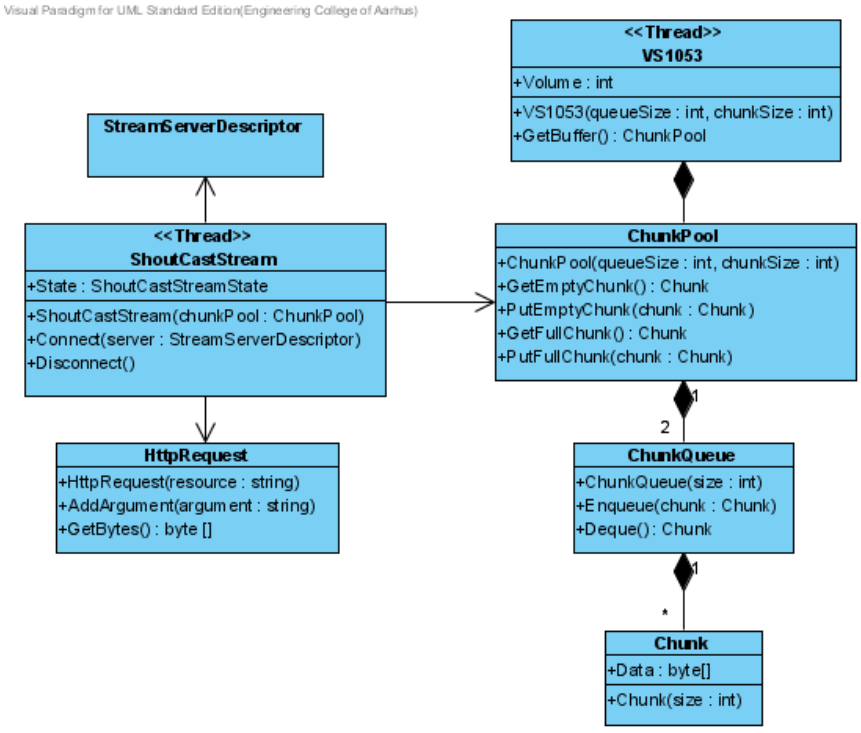
Figur 2 Klasse overview



Figur 3 HomeWindow View



Figur 4 PlayerWindow View



Figur 5 Playback View

2.1.1. Graphical User Interface (GUI)

Rock Tahoes GUI er opbygget med tre vinduer; et HomeWindow og to undervinduer; AlarmWindow og PlayerWindow. Alle vinduerne er implementeret med et StackPanel, der indeholder tre Panels

(TopPanel, MiddelPanel og BottomPanel). Alle tre Panels indeholder en række controller som childs; texts, bitmaps, buttons osv.

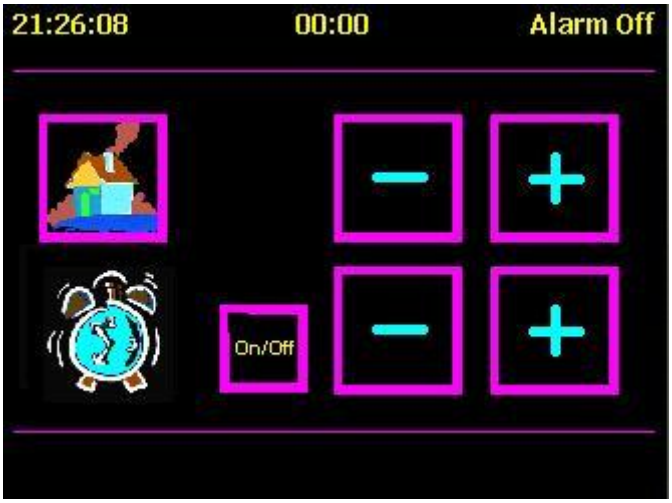
HomeWindow

HomeWindow viser de mest interessante oplysninger; Klokken, dato, temperatur, musikbuffer, musik der afspilles og state (Se 2.1.2). Desuden findes der to store knapper, som brugeren kan klikke på for at komme ind på de to undersider.

Figur 6 HomeWindow

AlarmWindow

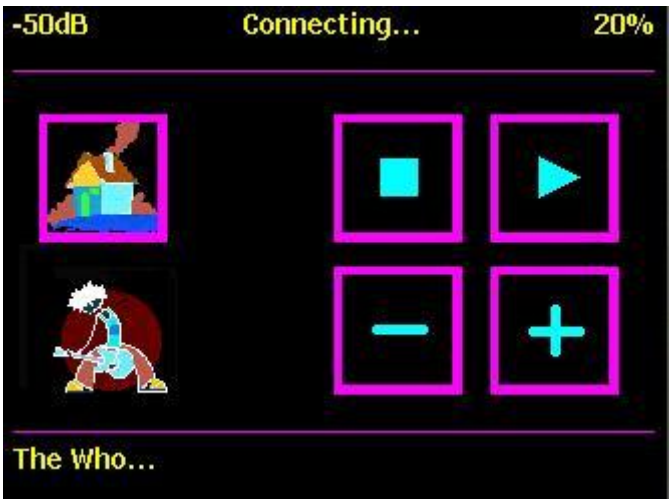
Alarmwindow viser alarmspecifikke oplysninger; alarm On/Off, tidspunkt og alarmtidspunkt. Der findes knapper til at stille alarmen med, samt en knap til at komme tilbage til hovedvinduet.



Figur 7 AlarmWindow

PlayerWindow

PlayerWindow viser Playerspecifikke oplysninger; Lydstyrke, State (Se 2.1.2), musikbuffer og hvilket nummer der spiller. Der findes knapper til start, stop, volume op, volume ned og til at komme tilbage til HomeWindow.



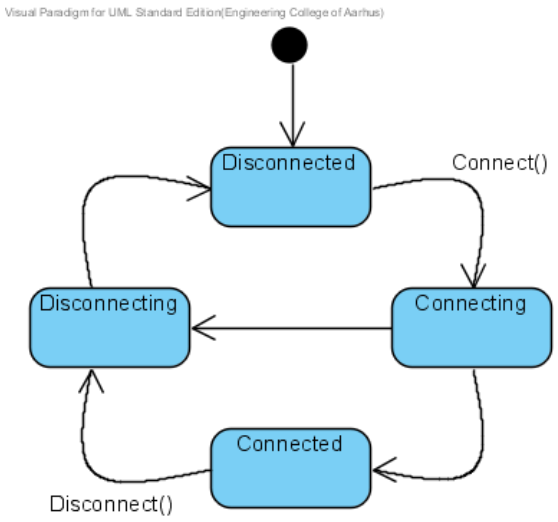
Figur 8 PlayerWindow

Button

.NET Micro frameworket tilbyder ikke knapper i form af controls og derfor er der lavet en klasse til oprettelse af knapper i Windows Forms stil. Denne klasse arver fra Control og har derfor Controls egenskaber. Den har desuden ekstra egenskaber og en OnRender override metode, som gør at den kan placeres et bestemt sted, at den kan have en bestemt udformning, at den kan have en bestemt tekst etc. Desuden indeholder en Button en IsIntersecting metode, som bruges til at spørge button om den ligger indenfor et bestemt område. Dette bruges når der fanges en StylusDown event, til at spørge knappen om trykket ligger indenfor det område som knappen dækker. Når en knap er Intersected raiser den et event, som bruges på samme måde som et Button_click event i Windows Forms.

2.1.2. Kommunikation med netradio server

Kommunikationen med netradioserveren består i oprettelse og nedlæggelse af forbindelsen til denne samt den egentlige real time streaming af MP3 audio data. Denne funktionalitet håndteres internt i ShoutCastStream klassen af en tråd der realiserer tilstandsmaskinen herunder.



Figur 9 State diagram

Skift i mellem disse tilstande styres til dels af brugeren af klassen og til dels af tråden selv.

For at oprette en forbindelse til en radioserver kaldes klassens Connect metode med et StreamServerDescriptor objekt der indeholder de informationer der er nødvendige for at oprette en forbindelse. Dette skifter tilstandsmaskinens tilstand til Connecting og den interne tråd begynder herefter at oprette forbindelsen. Lykkedes dette, skiftes til tilstanden Connected hvorved tråden begynder at fylde data i dens buffer.
Lykkedes det ikke at oprette forbindelsen skiftes til tilstanden Disconnecting hvorved alle interne data nulstilles. Når der er blevet ryddet op efter forbindelsen/forbindelsesforsøget skiftes tilbage til tilstanden Disconnected.

For at afbryde en aktiv forbindelse kaldes Disconnect metoden hvilket medfører et skifte til tilstanden Disconnecting, som beskrevet ovenfor.

Når der skiftes tilstand i ShoutCastStream rapporteres dette tilbage til anvenderen via et event hvilket benyttes til at opdatere skærmen.
Ud over forbindelsens øjeblikkelige tilstand er det også muligt at se meta information om den aktuelle audio stream. Dette er blandt andet sangtitel, bitrate og radiostationens navn som rapporteret fra serveren.

Som det ses ud fra klassens navn understøttes kun ShoutCast radiokanal af hvilke der er lang række. Protokollen for disse ligner HTTP 1.0/1.1 med et par ShoutCast specifikke tilføjelser. Som ved HTTP sendes der en request pakke når TCP forbindelsen til serveren er oprettet. Serveren sender herefter et statussvar og hvis alt gik godt sendes herefter en uendelig strøm af MP3 frames. Hvis det i requested blev specificeret at meta info ønskes, sendes der med faste intervaller information om sangtitel og andet

relevant metainfo.
Når klienten ikke længere ønsker at modtage data nedlægger den blot TCP forbindelsen.

2.1.3. Afspilning af lyd

Dekodning og afspilning af MP3 data fortages af en ekstern MP3 dekoder kreds[1]. Driveren for denne kreds sørger for initialisering af selve kredsen og den efterfølgende overførsel af datablokke fra applikationens buffer til kredsen.
Kommunikationen til dekoder kredsen sker via SPI bussen og kredsen optræder her som to SPI enheder da den indeholder to kanaler til henholdsvis kontrol og data.
Gennem kontrolkanalen, eller Serial Control Interface (SCI), initialiseres kredsen, volumen justeres og andre parametre kan indstilles og udlæses. Via datakanalen (Serial Data Interface – SDI) sendes audio data der skal afspilles.
De to kanaler er implementeret ved at oprette et SPI objekt og to SPI konfigurationer. Som standard er SDI konfigurationen valgt og forbundet med SPI bussen men når der sendes kontrolkommandoer skiftes midlertidigt til SCI konfigurationen.

VS1053 driveren indeholder en buffer hvori MP3 fra netradioen indsættes. Denne buffer (kaldet ChunkPool) er implementeret som to køer af henholdsvis tomme og fyldte datablokke (disse køer benævnes free og used). Ved opstart oprettes et antal chunks som indsættes i free køen. Når der efterfølgende modtages data fra serveren placeres denne data i en chunk fra free køen og denne chunk lægges i used køen. Dette fortages af ShoutCastStream tråden. VS1053 klassens tråd forbruger herefter chunks fra used køen og returnerer disse til free køen.
Når used køen bliver tom skifter afspilningstråden til ventetilstanden hvor der ventes på at mindst 70 % af alle chunks er placeret i used køen. Dette giver en prebuffering effekt der udglatter eventuelle hakker i datastrømmen.

2.1.4. Touch

Der er i .NET Micro frameworket en touch driver, som blot skal initialiseres for at fungere på en applikation.

```
Microsoft.SPOT.Touch.Touch.Initialize(Program);
```

Initialiseringmetoden tager en IEventListener som parameter, hvilket Program er, da den arver fra Application. Dette giver en udfordring når window skiftes, da touch stadig vil være initialiseret til det

[1] Den anvendte dekoder kreds er VS1053. Se databladet for denne for yderligere information.

vindue der sættes til MainWindow. Denne problematik er løst i ExtendedWindow ved at vinduer der arver fra denne bliver sat til MainWindow når de oprettes og ParentWindow gemmes og sættes til MainWindow igen når det nye vindue lukkes. På den måde følger touchfunktionaliteten det vindue der vises på skærmen. Metoden giver dog kun mulighed for at åbne vinduer i ét niveau under MainWindow, men det har været nok i vores applikation.

2.1.5. Temperaturføler

DeviceSolutions leverer en driver med Tahoe boardet, som gør det enkelt at læse en temperatur fra Tsc2046 temperatur føleren med følgende funktion:

```
DeviceSolutions.SPOT.Hardware.TahoeII.Tsc2046.ReadTemperature();
```

Temperaturføleren er dog ret ustabil og svinger op til 3°.

2.1.6. Ur

For at brugeren af systemet ikke manuelt skal indstille klokken anvendes Time protokollen til at indstille uret.

Timeprotokol

Time protokollen er en implementation af RFC868 der beskriver en simpel request/response model hvor klienten sender en tom pakke til serveren over enten TCP eller UDP. Serveren svarer herefter med en 32 bit værdi der repræsenterer antal sekunder siden klokken 00.00 den første januar år 1900.

Tidssynkronisering

Uret er implementeret så det kører i sin egen tråd og bruger en timer til at lægge sekunder til tiden. Dette bliver hurtigt upræcist og derfor opdaterer tråden til den eksakte tid fra timeserveren hvert minut. Dette minimerer requests over netværk og gør at tiden altid er opdateret. Uret er implementeret som en Singleton så der altid kun findes ét ur og ét klokkeslæt, og det er det samme klokkeslæt som alle klasser kan få fat i.

Alarm

Alarmen er implementeret så den kører i sin egen tråd. Den tjekker hvert sekund, om alarmen er sat, og om alarm-tidspunktet er det samme som urets tidspunkt. Hvis det er tilfældet raises et AlarmEvent. Alarmen er implementeret som en Singleton, så der altid kun findes ét alarmtidspunkt og det er det samme tidspunkt, som alle klasser kan få fat i.

2.2. Generelt om .NET Micro framework

Det er igennem arbejdet med .NET Micro blevet klart at dette framework er nyt og langt fra færdigudviklet. Der er i gruppen blevet talt om mange gange, hvilke mangler og fejl der er ved frameworket og om, hvordan Microsoft kan slippe af sted med at frigive noget, der i vores øjne slet ikke

lever op til det ”rigtige” .NET framework. Vi har nedenfor samlet en liste over fejl og mangler, som vi har erfaret under udviklingen af Rock Tahoe.

- Når man tegner en linje kan man vælge tykkelse, men den bliver altid 1 pixel bred.
- Tegnes et rektangel med runde hjørner bliver tykkelsen af linjen altid 1 pixel.
- Når man bruger touch på Tahoen, og StylusDown eventet bliver kaldt tre gange, så genereres et musehøjreklikker på PCen.
- En gang imellem er Tahoen sløv til at opfange stylusDown events.
- En gang imellem forsvinder markøren.
- Asynchronous Programming Model er forsvundet fra Socket (ingen BeginX/EndX metoder)
- StringBuilder mangler hvilket virker underligt da det er en god mulighed for at optimere.
- Når man starter et nyt MF projekt i Visual Studio hedder det ”Windows application” selvom MF ikke har noget med Windows at gøre.
- Generics mangler hvilket er en relativ stor mangel om end det sparer en del ressourcer.
- String.Split() er reduceret - der mangler StringSplitOptions.
- String.StartsWith() findes ikke.
- Der findes ingen GUI editor og man er tvunget til at bruge en handicappet udgave af WPF.
- InvokeRequired findes ikke på et vindue.
- Ofte kan debuggeren ikke initialiseres og forbindes til target men target starter alligevel.
- Ind i mellem skal man resette og/eller trække USB stikket ud nogle gange for at kunne deploye.
- Beskeden "The program '[x] Micro Framework application: Managed' has exited with code 0 (0x0)." blev modtaget mens applikationen kørte fint videre.
- Standard fonten i en Windows Application er underlig - nogle tegn står helt op ad hinanden andre har et mellemrum.
- Thread har ingen "IsBackground" property.
- VS crasher en gang i mellem når man forsøger at køre en applikation på target.
- Kodeeksemplerne viser dårlig programmeringsstil og/eller viser ikke ”den rigtige måde” at gøre bestemt ting på. Eksempler: brug af ”goto”, rester af testkode, kald af Receive på en lukket Socket.
- DHCP klienten initialiseres inden debuggeren startes og er så langsom at VS giver timeout inden en IP adresse tildeles.
- Det er ikke muligt at indstille en proxy i MF udgaven af .net (WebProxy i rigtig .net).
- Tråde kan ikke navngives hvilket besværliggør debugging.
- Enums .ToString() giver en int i MF mod normalt teksten.

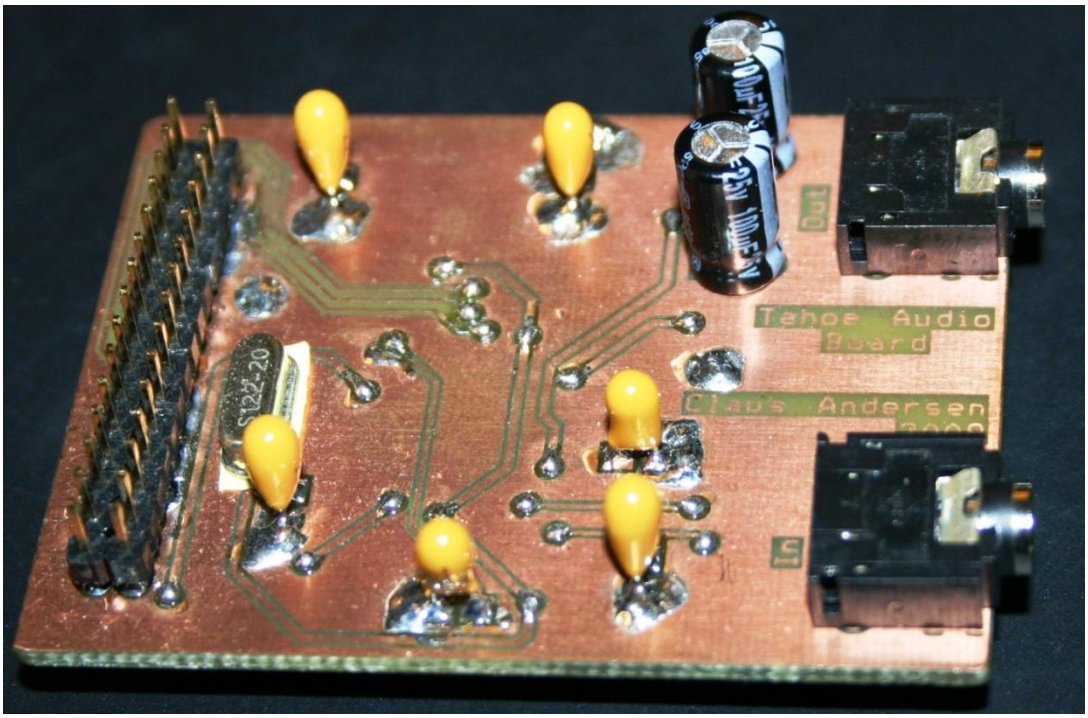
3. KONKLUSION

Der er lykkedes at designe en applikation til Microsoft .NET Micro frameworket, der lever op til en del af kravene i den oprindelige problemformulering. Det er ikke fuldt ud lykkedes at implementere lyddelen, da der er store problemer med hastigheden på SPI bussen. Det formodes at fejlen ligger i Device Solutions portering af .NET Micro framworket til i.MXS processoren.

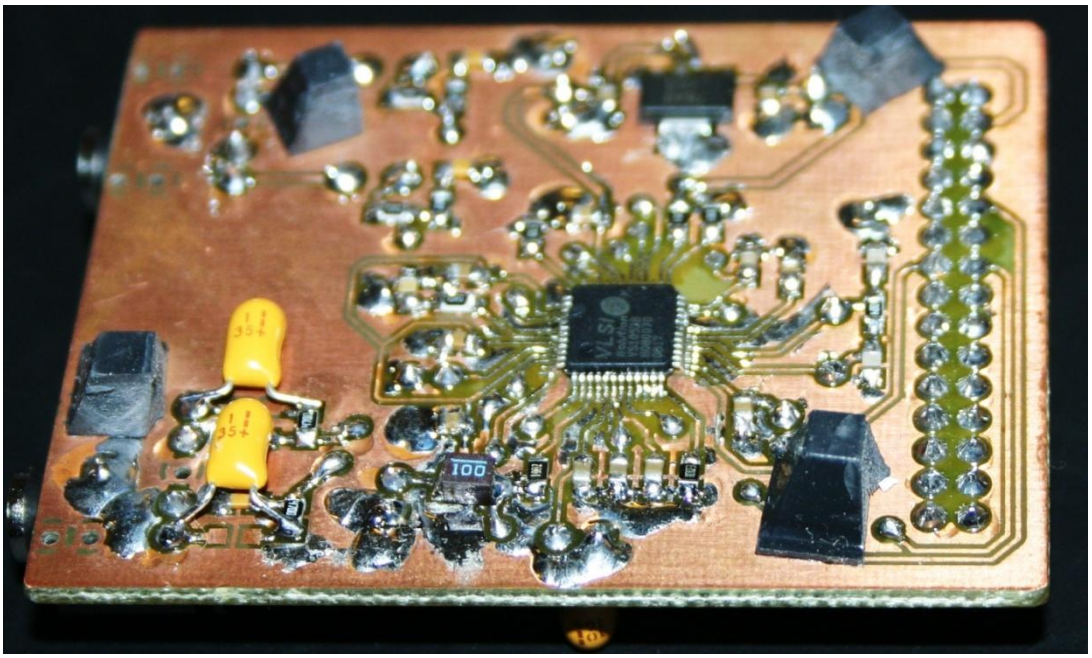
C# har en plads i softrealtime indlejrede systemer, men det konstateres at .NET Micro frameworket langt fra er klar til brug og at Microsofts PR-maskine arbejder hurtigere end deres udviklingsafdeling.

Under arbejdet med Rock Tahoen er det blevet klart at hvis alle funktionaliteter i .NET Micro frameworket og Tahoe Firmwaren virkede som forventet, ville det være et lækkert værktøj til udvikling af softrealtime indlejrede systemer.

4. Bilag



Figur 10 Mp3 lydmodul - Overside



Figur 11 Mp3 lydmodul – Underside