Rapport

EasyRun – En løbers bedsteven

Anders Arnfast 06525, Martin Søberg 06551, Ken Falk 06504

09

1. INDHOLD

2. INTRODUKTION

I dette projekt skal der designes og udvikles et løbehjælpemiddel der kan vise følgende under et løb:

- Løbelængde.
- Nuværende hastighed.
- Gennemsnitlige hastighed.
- Energi forbrænding.

Have følgende funktionaliteter:

- Downloade en brugerprofil fra en desktop pc indeholdende navn, vægt og skridtlængde
- Kunne gemme hvert løb der er fortaget, dvs. løbet skal kunne persisteres.
- Gøre det muligt at løbe mod et tidligere løb som er gemt.
- Kunne uploade samt download gemte løb til en PC.

Til udvikling af disse bliver der benyttet:

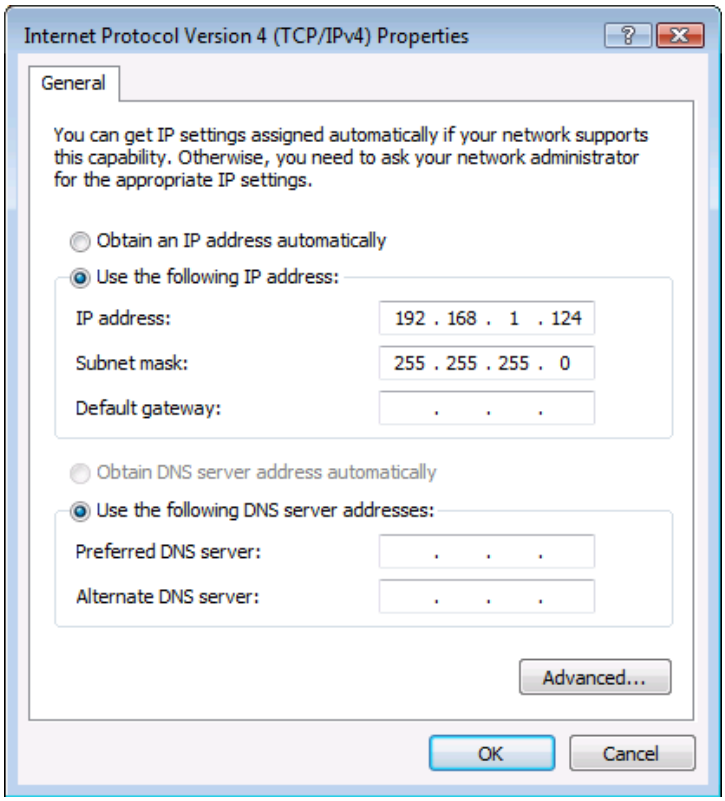
- Tahoe-II kort, til at installere applikationen på.
- Accelerometer , til at registrere antal skridt.
- Ethernet, til at kommunikere med PC med.
- Touch skærm, til brugergrænseflade.

Dette projekt er fungerer som afsluttende opgave i ITEWM faget og vil give et indblik i hvorledes der udvikles systemet til embedede enheder.

OPSÆTNING

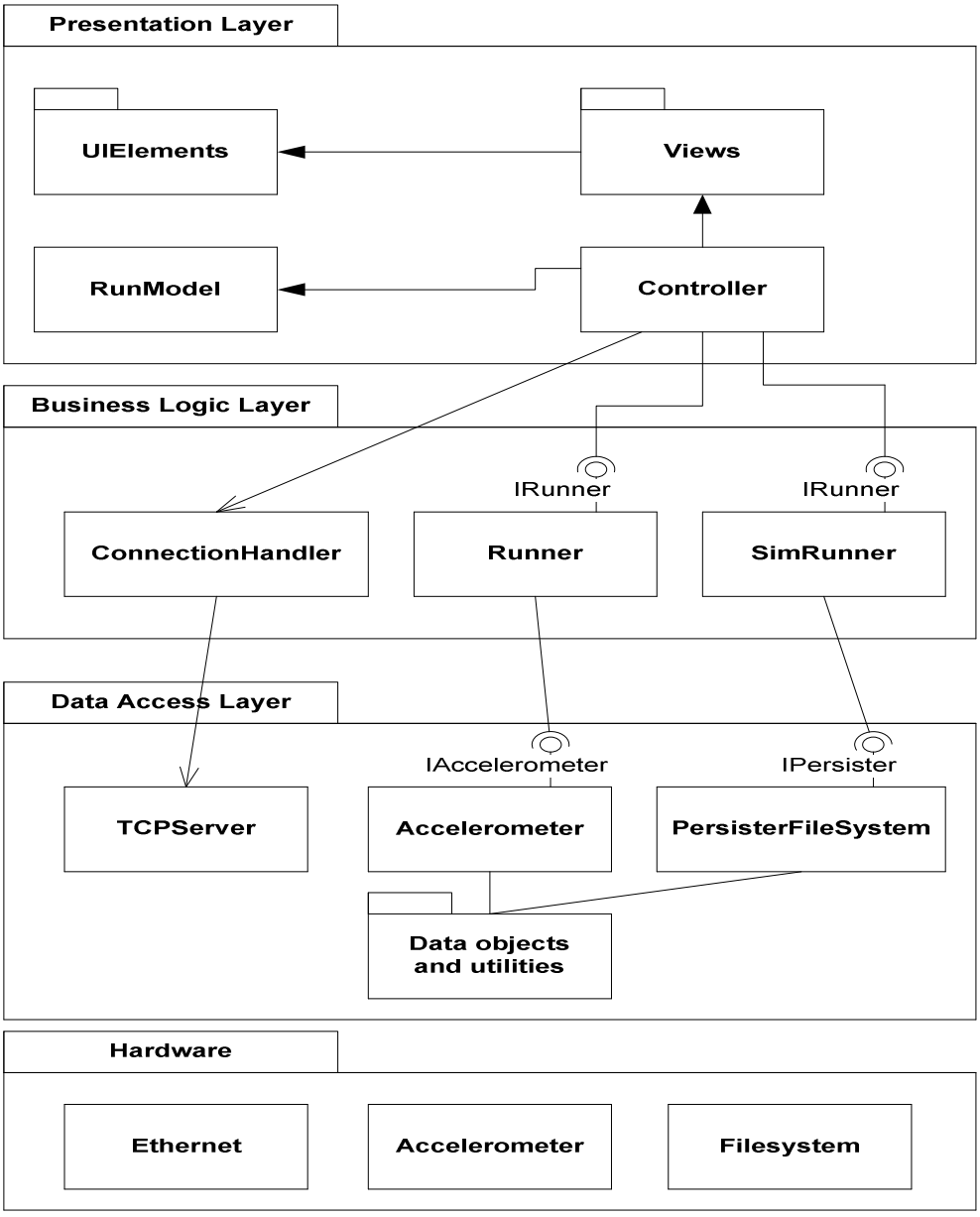
For at desktop applikationen kan kommunikere med EasyGo platformen over ethernet, skal desktop pc'en konfigureres med opsætningen på Figur 2.

Ip i desktop applikationen er sat til at connecte til 192.168.1.123, hvilket er Tahoe-II kortets ip. Hvis dette kræves ændre foregår det i TCPClient og string host="<<nye ip>>".



Figur 1 - Ip opsætning på desktop pc'en.

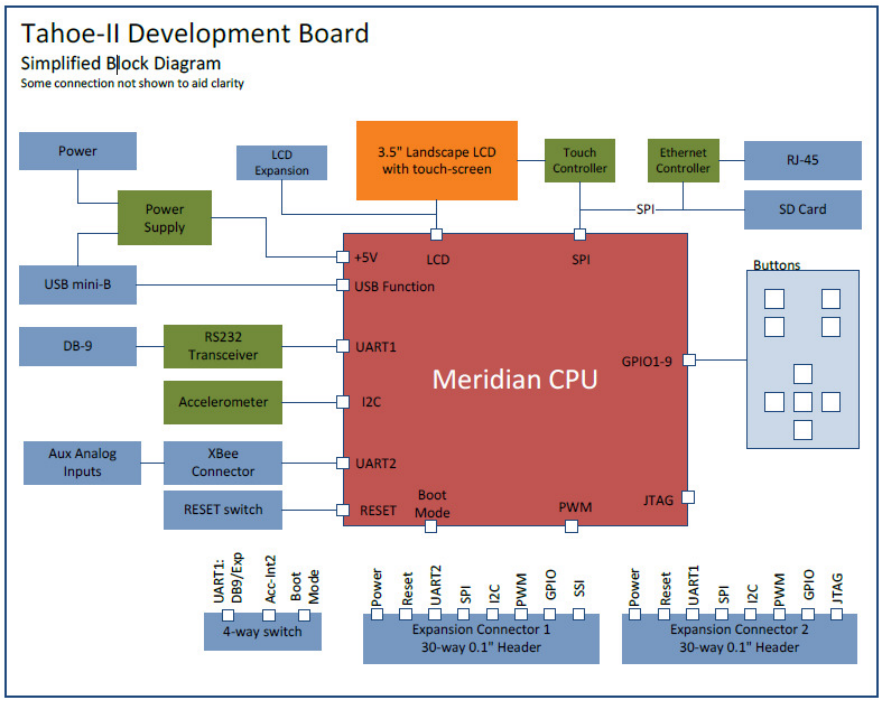
3. SYSTEM ARKITEKTURDESIGN



Figur 2 - Diagram over system arkitekturen

4. HARDWARE DESIGN

Til udvikling er der benyttet et Tahoe-II development board, som ses på figuren herunder.



Figur 3 - Hardware oversigt, Tahoe-II

Ethernet Controller & Touch Controller

På dette kort benyttes SPI bussen til kommunikation med Ethernet Controlleren, driveren til denne er wrappet i .Net Micro Frameworket under System.Net.Sockets.
SPI benyttes også til Touch Controlleren, som håndterer alle inputs på skærmen. Disse touch inputs er linket til applikationen igennem frameworket Microsoft.SPOT.Touch.Touch.Initialize(<<Applikation>>), hvilket resulterer i at alle touch inputs bliver sendt til funktionen OnStylusDown(StylusEventArgs e) som findes i UIElement.

SD Kort

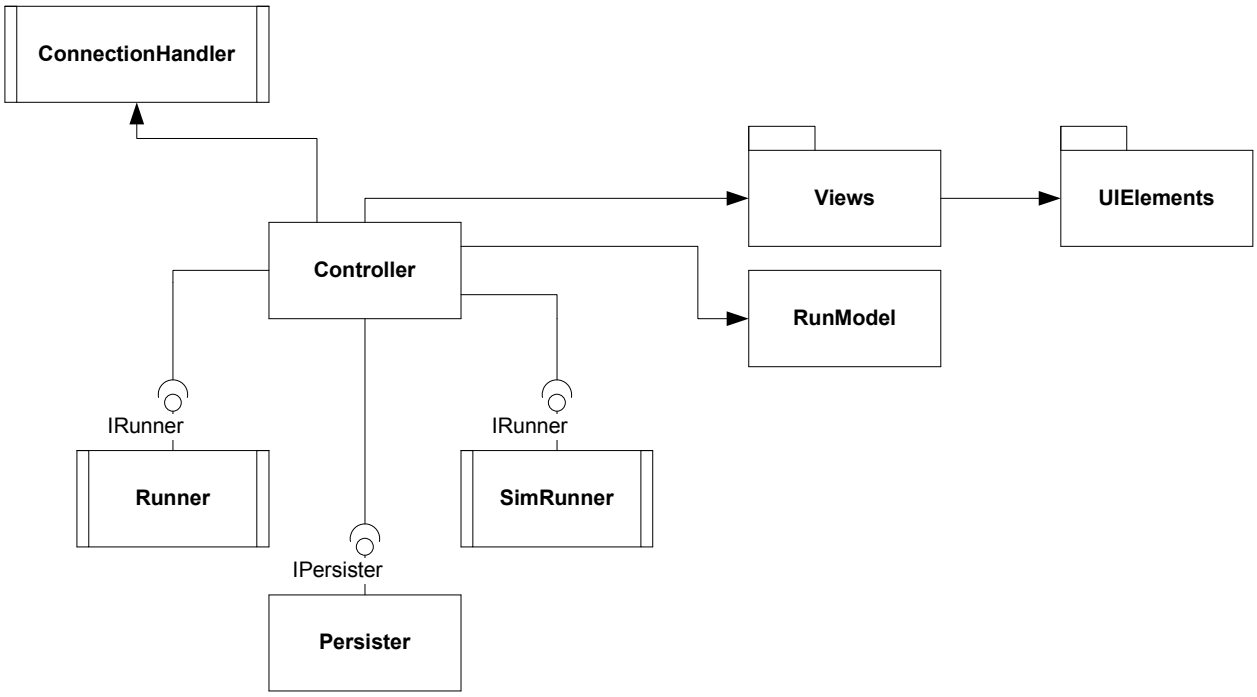
SPI bussen benyttes også til at tilgå SD kortet. På dette kort er der placeret en fil system. Dette kan tilgås gennem klassen System.IO.Directory og der derefter benyttes almindelige Streams til at placere data på kortet.

Accelerometer

IC2 bussen benyttes til at kommunikere med accelerometret på Tahoo-II kortet. Device Solutions har udgivet en driver til MMA7455 accelerometret. Da der i dette projekt skulle registreres skridt, ville det være optimalt at modtage interrupts fra accelerometret. Databladet for MMA7455 blev gennemgået, og det viste sig at det netop var muligt at sætte MMA7455 op til at give interrupts, hvis accelerationen på en af x-y-z akserne oversteg en konfigurérbar grænseværdi. Dette var en optimal løsning, da der således ikke skulle konstant skulle polles på accelerometret, men at problemet blev løst på hardware niveau, og der kunne modtages et interrupt, når der var foretaget en interessant acceleration(et skridt). Der blev skrevet en klasse, som udvidede mulighederne i driveren fra Device Solutions, således at der kunne modtages interrupts fra accelerometret. Det viste sig desværre, at der ikke blev modtaget interrupts fra accelerometret, og polling løsningen måtte implementeres i stedet. Dette var lidt ærgeligt, da databladet for MMA7455 lagde op til en sådan løsning, samt der ifølge Tahoo-II specifikationen var forbindelse fra cpu'ens GPIO14 pin til accelerometerets interruptpin (INT1). Det kunne dog tyde på at det var et generelt problem, da MMA7455 driveren fra Device Solutions heller ikke understøtter interrupt funktionaliterne beskrevet i databladet for MMA7455 accelerometret.

5. SOFTWARE DESIGN

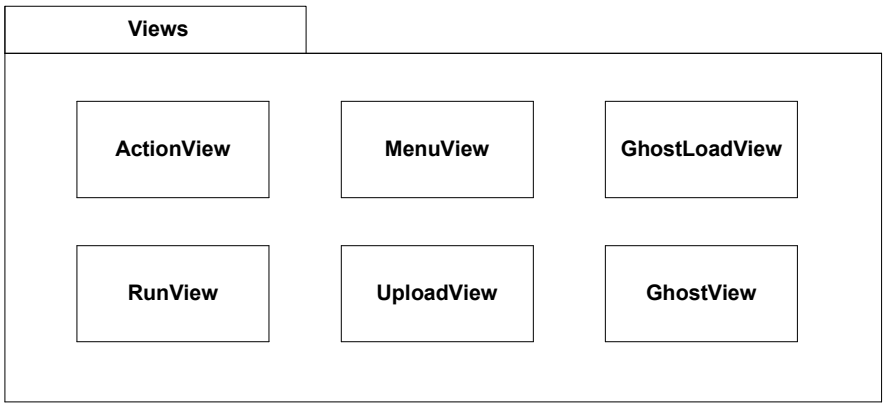
PRESENTATION LAYER



Controller

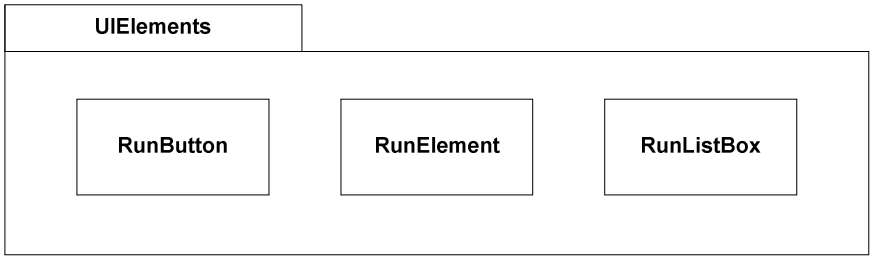
Presentation layer er bygget op omkring MVC (Model View Controller) princippet. Controlleren har mulighed for at styre hvilket views der skal vises, samt modellere RunModellen. Controlleren får input fra Runner klassen, der giver de værdier der opsamles fra den nuværende løbe tur. SimRunner arbejder på tidligere gemte løbeture og bliver brugt når skal lave et Ghost Løb (man løber imod en tidligere gemt løb). Controlleren har også forbindelse til at persister sådan er i stand til at hente og gemme løb. Controlleren hoster også ConnectionHandler, der styrer alle connectins over ethernet til systemet.

Views



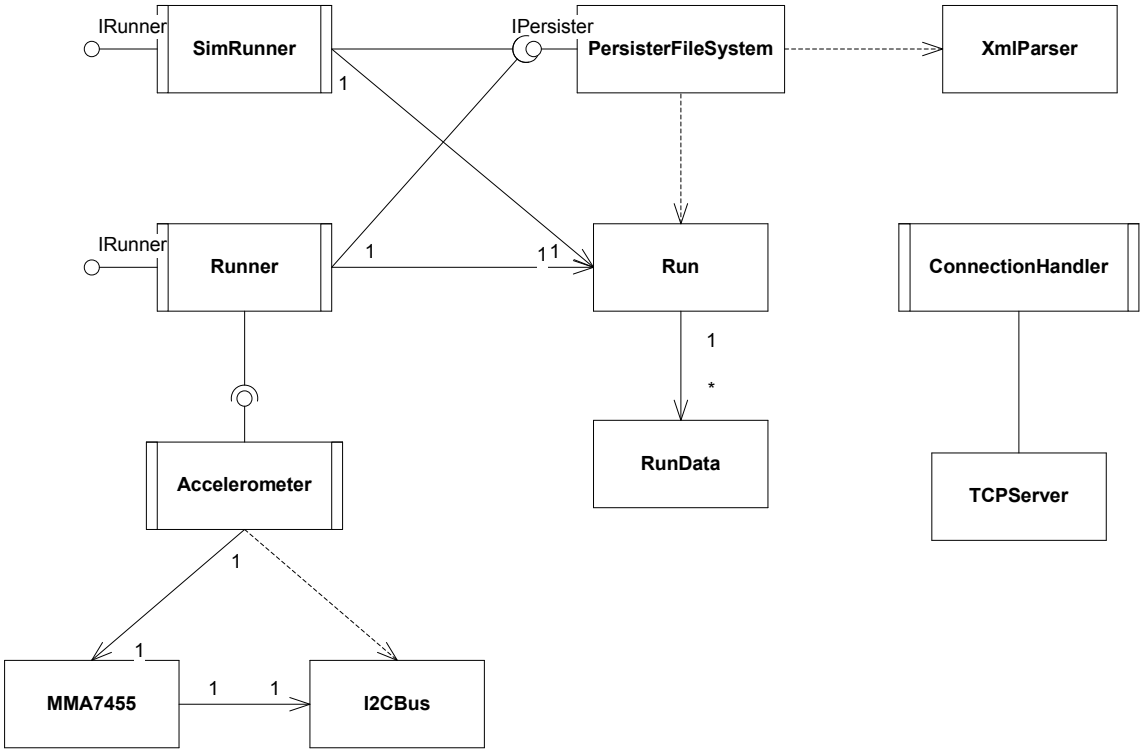
Der er 6 forskellige views i systemet. MenuView og Action view bliver brugt i til at håndtere genellelle inputs fra brugeren. MenuView bliver altid vist, samt ActionView kun bliver vist under normal og ghost run. Dette er gjort for at ikke skulle lave input panel i alle views, og i stedet genbruge noget kode. RunView repræsentere det view man ser når man løber normalt og GhostView repræsentere det view når man løber imod en tidligere tid. GhostLoadView er det view hvor man loader et tidligere løb. UploadViewet bruges når der en ekstern klient der vil tilgår vores løbe device.

UIElements



Da Micro Frameworket ikke stiller mange UI elementer til rådighed, har det været nødvendigt at lave nogen selv. RunButton fungere som en button, i en simpel udgave, som vi kender fra .NET Frameworket. RunElement viser løbe status, og får en RunModel injectet,som den arbejder op imod. RunListBox er en simpelListbox der bliver brugt når man skal være hvilken tidligere rute man vil løbe imod

BUSINESS LAYER & DATA ACCES LAYER



Figur 4 - Klasse diagram for BLL og DAL

Accelerometer

Denne klasse implementerer IAccelerometer interfacet. Denne klasse afkobler systemet fra MMA7455 driveren tilgængelig fra Device Solutions hjemmeside. Dette betyder, at der kunne benyttes en anden driver i systemet, og at kun ville være nødvendigt udskifte Data Access Laget, men de øvrige lag kunne genbruges uden ændringer, så længe at IAccelerometer interfacet stadig implementeres. Accelerometer klassen har ansvaret for at kommunikerer med accelerometer hardware, og afgive events, når der er registreret et skridt.

Runner

Udfører de beregninger der skal til for at beregne.

- Løbelængde.
- Nuværende hastighed.
- Gennemsnitlige hastighed.
- Energi forbrænding.

Dette gøres ved brug af events fra Accelerometeret og informationer fra Hvert 2. sekund sender Timeren et event med de nyeste beregninger til brugergrænsefladen.

SimRunner

Denne klasse benytter et forud lavet løb som lægger gemt på SD Kortet. Når SimRunner kører opdaterer den brugergrænsefladen med RunData objekter ud fra deres timestamps

PersisterFileSystem

Denne klasse kan persistere et Run på filsystemet, kan hente navnet på alle de filer som ligger i EasyRun mappen og hente en et Run objekt fra filsystemet igennem XMLParser.
Arver fra IPersister således at andre persisterings metoder kan laves. Eksempelvis benytter klassen Persister en anden persisterings metode, men persistering på igennem fil systemet blev foretrukket.

XMLParser

Denne klasse har metoder til at håndtere xml serialization . Da .Net Micro Frameworket ikke håndterer direkte xml serialization er dette gjort igennem manuel kodning.

string RunToXml(Run run)

Denne metode kan serializere et Run objekt til en string. Denne string er bygget op efter xml standarden. Eksempel på en returneret string ses herunder.

```
<savedrun>
   <run runnername="Ketil" starttime="0">
      <rundata avaragespeed="3.5" burnage="0" currentspeed="3.5" totallength="0" totaltime="0"/>
      <rundata avaragespeed="3.5" burnage="0" currentspeed="3.5" totallength="7" totaltime="20000000"/>
      <rundata avaragespeed="3.5" burnage="0" currentspeed="3.5" totallength="14" totaltime="40000000"/>
   </run>
</savedrun>
```

Run FromXmlFile(string filename)

Denne metode kan læse en fil, med indehold som set ovenstående, og konvertere det til et Run objekt.

ConnectionHandler

I denne bliver alle klient forbindelserne håndteret. Klienterne kan lave et request om fire forskellige ting. Få vist hvilke løb er tilstede, downloade et bestemt løb, uploade et bestemt løb, downloade en brugerprofil filen eller uploade en brugerprofil.

TCPServer

Tcp serveren venter på at der forbinder en klient og returnerer den socket som klienten er forbundet på til ConnectionHandler.

I denne benyttes der af vigtige klasser:

- System.Net.Sockets.socket til at vente på klient forbindelser og holde klient info.
- System.SPOT.Net.NetworkInfo til at hente IP på kortet.
- System.Net.Sockets.ProtocolType.TCP til at angive at det er en tcp forbindelse der bliver oprettet, dette giver en form for datasikring.

6. RESULTATER

Der er blevet udviklet en prototype af EasyRun på Tahoo-II udviklingskit'et. EasyRun er i stand til at registrere skridt ved hjælp af accelerometret på Tahoo-II kortet. Der er endvidere udviklet en tynd desktop applikation, som gør det muligt at loade brugerprofiler ned på EasyRun platformen gennem et ethernetkabel. Brugerprofilen består af brugerens navn, vægt og skridtlængde. Denne information benytter EasyRun til at udregne hastighed, forbrænding og distance for et løb. Det er endvidere muligt at up- og downloade et gemt løb til en desktop pc via ethernetkabel. Herefter er det muligt at løbe "mod" et gemt ghost løb. Det er udviklet en brugergrænsefalde, som viser relevant løbedata. Brugergrænsefalden benytter touchskærmen til at modtage inputs fra brugeren.

7. KONKLUSION

EasyRun er blevet udviklet på Tahoo-II udviklingsplatformen. Et færdigt produkt vil naturligvis have en meget anderledes og mere praktisk formfaktor. Der vil endvidere skulle udvikles en energikilde, formodentlig i form af en genopladeligt batteri. .NET Micro FrameWork er et meget sparsomt framework i forhold til .NET til desktop pc'er, men de mest basale funktionaliteter er dog til rådighed, gennem samme API som det fulde .NET framework. Derved kan meget kode portes direkte fra desktop applikationer. Så Microsoft har ret i at springet fra .NET ned til at kode .NET Micro ikke er så stort, og det kan gøres .NET udviklingserfaring. Der hvor Micro Framework adskiller sig mest fra .NET, er den nære kobling til harware enheder. I den forbindelse introduceres der også en del nye biblioteker. Det har også vist sig, at langt de største problemer har bestået i at kommunikere med hardwaren. Da der f.eks. skulle kommunikeres med et SD kort, for at persistere data, kunne dette SD kort ikke tilgås. Det viste sig, at være fordi kortet skulle formateres med som FAT16, og med en clusterstørrelse på 512bytes. Der var også problemer med at aktivere interrupts på accelerometret, og det var nødvendigt at polle værdier fra det i stedet.

Det kan konkluderes, at Microsoft med .NET Miro Framework har formået at gøre programmering til en embeded chip enkel for udviklere med erfaring i .NET platformen. Det er muligt at portere meget kode fra det fulde .NET til .NET Micro med en lille indlærings indsats. Det kræves dog en større indlæring, så snart, at der skal kommunikeres med hardware, da der herigennem introduceres nye og ukendte biblioteker i forhold til .NET framwork'et.