ITWEM1

Kitchen Commander

Group 8 - SVAPS

4/6/2009

Kasper Sørensen - 06128

Jesper Thue Kristensen - 05230

Frederik Laulund - 06888

Christian Jensen - 06791

Indholdsfortegnelse

1 INTRODUKTION 3

2 SÅDAN VIRKER ET PLOGG 4

3 SÅDAN VIRKER SYSTEMET 4

4 USE CASES 5

4.1 AKTØRBESKRIVELSER 5
4.2 USE CASE 1: DETEKTER TÆNDT KOGEPLADE 5
4.3 USE CASE 2: DETEKTER BEVÆGELSE 6
4.4 USE CASE 3: AKTIVÉR ALARM 6
4.5 USE CASE 4: SLUK FOR KOGEPLADE 7

5 ARKITEKTURDESIGN 8

6 SOFTWARE DESIGN 9

7 HARDWARE DESIGN 11

8 ERFARINGER MED MICRO FRAMEWORK OG TAHOE II 13

9 BRUGERGRÆNSEFLADE 14

9.1 STATUS 14
9.2 LOG 14
9.3 ADVARSEL 15

10 KOMMUNIKATION 16

11 RESULTATER 17

12 KONKLUSION 17

13 REFERENCER 18

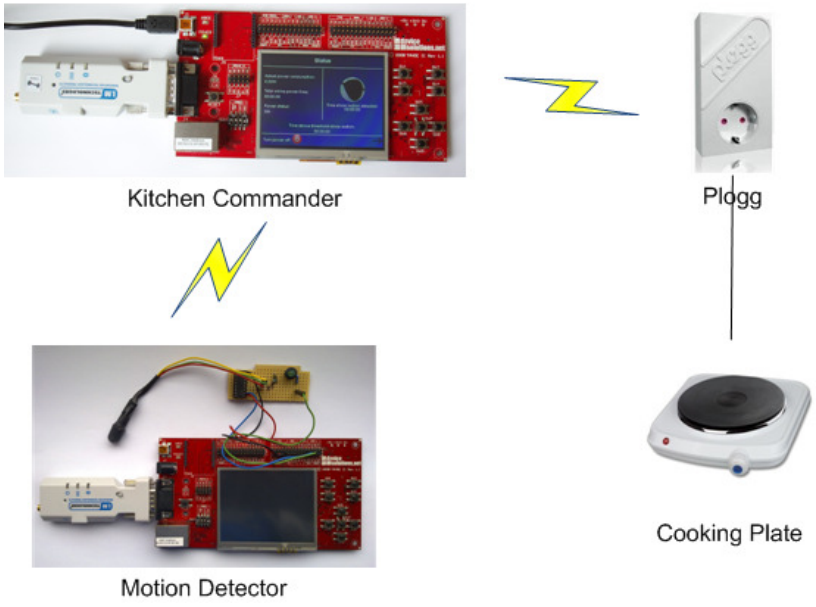
1 Introduktion

I forbindelse med faget ITWEM1 på Ingeniørhøjskolen i Århus er undertegnede blevet stillet opgaven at udarbejde et mini-projekt. Som krav skulle der inddrages et eller flere elementer fra faget, og vi har valgt at udvikle et system med .NET Micro Framework 3.0, der afvikles på et Tahoe II board. Systemets funktion skåret ind til benet er, at kunne identificere et inaktivt apparat og evt. slukke for det, logge relevante hændelser og vise dem samt strømforbruget på en skærm. Alle medlemmer i projektgruppen har også fulgt faget ITAPC1, hvori der også skulle udvikles et meget lignende system [1]. Derfor har der naturligvis været udflydende grænser for, hvornår der blev arbejdet på det ene og henholdsvis det andet projekt. Vi har valgt at tage udgangspunkt i samme case som i ITAPC1 kurset, som er kort beskrevet i det følgende afsnit.

Casens primære fokus er at afhjælpe brande, der opstår i køkkenet grundet en glemt kogeplade. Det er hovedsageligt med tanken på at hjælpe ældre, og desuden lette sundhedsvæsnet ved at give dem mulighed for at kontrollere systemets log og derved kunne spotte tidlige tegn på demens eller andre ændringer i den ældres opførsel. Det vil derved være muligt at rådgive den ældre og dennes familie, så de nødvendige foranstaltninger kan sættes i gang, inden en evt. ulykke skulle opstå.

Projekterne benytter flere ens komponenter, men har hver især 2 forskellige kontrolenheder, som vi har kaldt Kitchen Commander (KC). ITAPC1 delen benytter en ASUS Eee Top [2], der er en touch PC, med så fysisk småt hardware, at det er indbygget i skærmrammen. Dette projekts Kitchen Commander implementeres på det velkendte Tahoe II board. Begge projekter deler samme Bluetooth elmåler, de såkaldte Ploggs [3], samt en Bluetooth bevægelsessensor, der også er implementeret på et Tahoe II board.

Denne rapport beskriver implementeringen af systemet, der ses på Figur 1**,** som inkluderer de Tahoe II baserede Kitchen Commander og Bluetooth bevægelsessensor og Plogg. I projektet udforskes og afprøves flere forskellige teknologier og discipliner med varierende succes, men vil dog alle blive beskrevet i dette dokument. Forudsætningen for at kunne udføre projektet er den afholdte undervisning i ITWEM1, men i den grad også tidligere erfaringer.



Figur 1. System oversigt

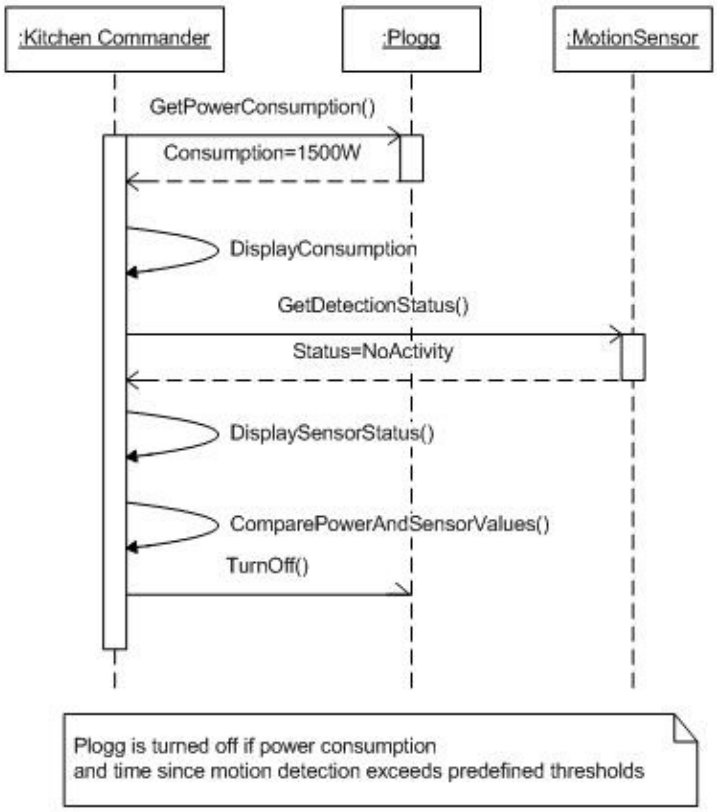
2 Sådan virker et Plogg

Figur 2. Plogg

Et Plogg, der ses på Figur 2, virker ved at det sættes i en normal stikkontakt og et apparat tilsluttes Plogg'et. I vores projekt er Plogg'ets hovedfunktion at overvåge og indsamle data om strømforbruget. Kommunikationen foregår via Bluetooth og afhængigt af hvilken kommando, man sender til Plogg'et, kan man bl.a. modtage data om aktuelt og historisk strømforbrug. Der kan også sendes en kommando til Plogg'et, så et tilsluttet apparat ikke får strøm.

3 Sådan virker systemet

Et apparat overvåges og kontrolleres af et Plogg. Plogg'et sender hele tiden via Bluetooth det aktuelle strømforbrug til Kitchen Commander, der viser det på skærmen. Hver gang bevægelses sensoren detekterer en bevægelse, sender den en besked til Kitchen Commanderen. Hvis der ikke har været bevægelse inden for et prædefineret tidsrum, gives der en visuel alarm. Reageres der ikke på denne, slukkes der for det tilsluttede apparat. Systemet logger de forskellige handlinger og inputs fra brugeren. På Figur 3 ses et tidligt udkast til et sekvensdiagram over systemets hovedfunktionalitet.



Figur 3. Tidligt sekvensdiagram

4 Use Cases

4.1 Aktørbeskrivelser

Kitchen Commander (KC), primær
SUD

Bruger, sekundær
Person der benytter SUD

Bevægelsessensor, primær
Sensor og Tahoe II kort

Plogg, sekundær
Intelligent strømstik

Kogeplade, sekundær
Kogeplade

4.2 Use Case 1: Detekter tændt kogeplade

Mål:
At detektere at kogepladen i fokus er tændt.

Initiering:
KC

Aktører og interessenter:
KC, bruger, Plogg og kogeplade

Startbetingelser:
Kogepladen er forbundet til lysnettet via Plogg.

Slutresultat ved succes:
KC har detekteret, at kogepladen er tændt.

Normalforløb:

1. Kogepladen tændes.
2. KC spørger Plogg om det aktuelle strømforbrug.
3. Hvis strømforbrug større end 0 er kogepladen tændt.
4. Det noteres af KC, at kogepladen er tændt.

4.3 Use Case 2: Detekter bevægelse

Mål:
At detektere bevægelse.

Initiering:
Bevægelsessensor

Aktører og interessenter:
Bevægelsessensor, KC og bruger

Startbetingelser:
Bevægelsessensoren er startet og kalibreret.

Slutresultat ved succes:
KC har modtaget en besked fra bevægelsessensoren, der angiver, at der er detekteret bevægelse.

Normalforløb:
1. Bevægelsessensorer detekterer bevægelse.
2. Bevægelsessensoren sender en besked via Bluetooth til KC.
3. KC nulstiller en timer, der tæller op så længe, der ikke er registeret bevægelse.

4.4 Use Case 3: Aktivér alarm

Mål:
At aktivere alarmen, der informerer om en tændt og ubenyttet kogeplade.

Initiering:
KC.

Aktører og interessenter:
KC, bevægelsessensor, Plogg og kogeplade.

Startbetingelser:
At Use Case 1 og Use Case 2 er forløbet uden undtagelse.

Slutresultat ved succes:
KC slår alarm.

Normalforløb:
1. KC konstaterer, at elforbruget overstiger en tærskel over en bestemt periode
2. KC konstaterer, at tiden siden sidste registrerede bevægelse er overskredet.
3. En visuel alarm startes af KC.

4.5 Use Case 4: Sluk for kogeplade

Mål:
At slukke den tændte, men ubenyttede kogeplade.

Initiering:
KC

Aktører og interessenter:
KC, Plogg og kogeplade

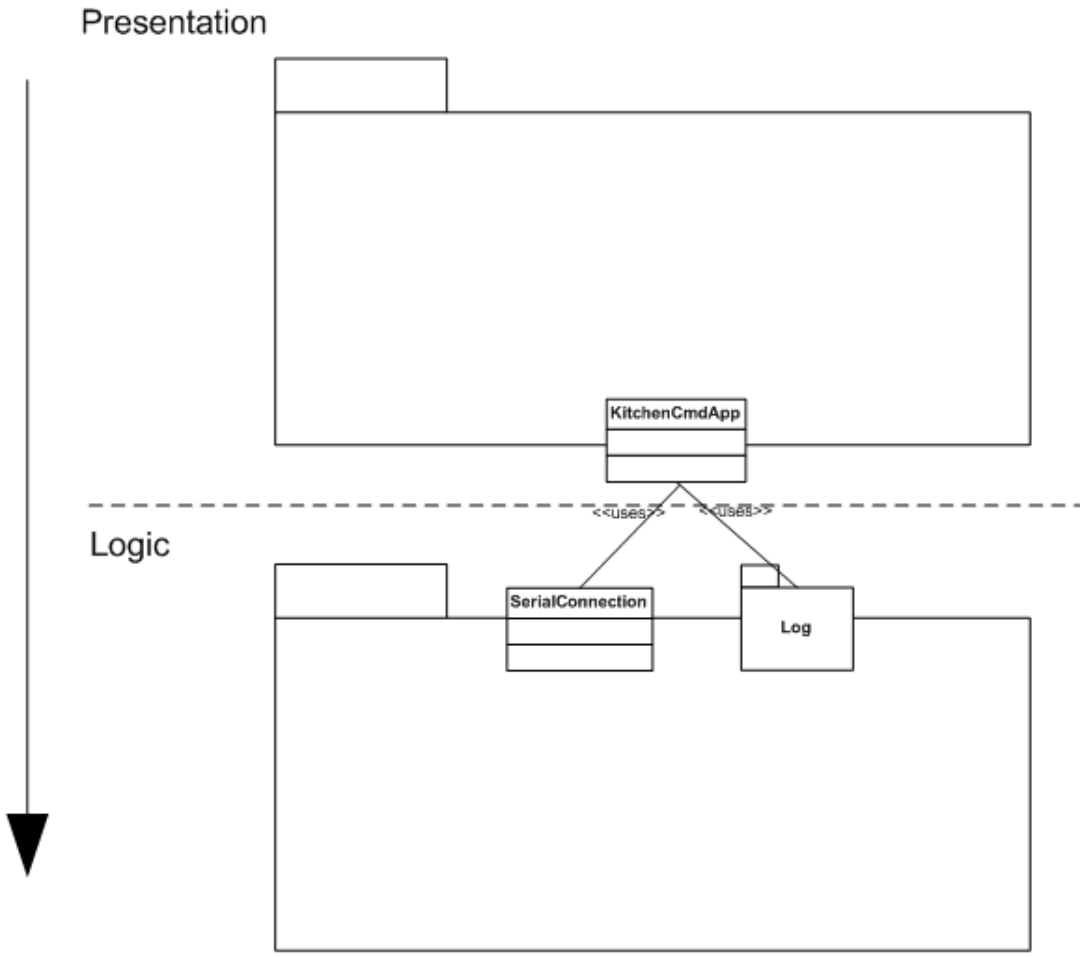
Startbetingelser:
At Use Case 1, Use Case 2 og Use Case 3 er forløbet uden udtagelser.

Slutresultat ved succes:
KC har slukket for strømmen til kogepladen, via Plogg.

Normalforløb:
1. Visuel alarm startes af KC.
2. KC konstaterer, at brugeren ikke har reageret på alarmen
3. Der detekteres ingen bevægelse.
4. KC slukker for strømmen til kogepladen via Plogg.

5 Arkitekturdesign

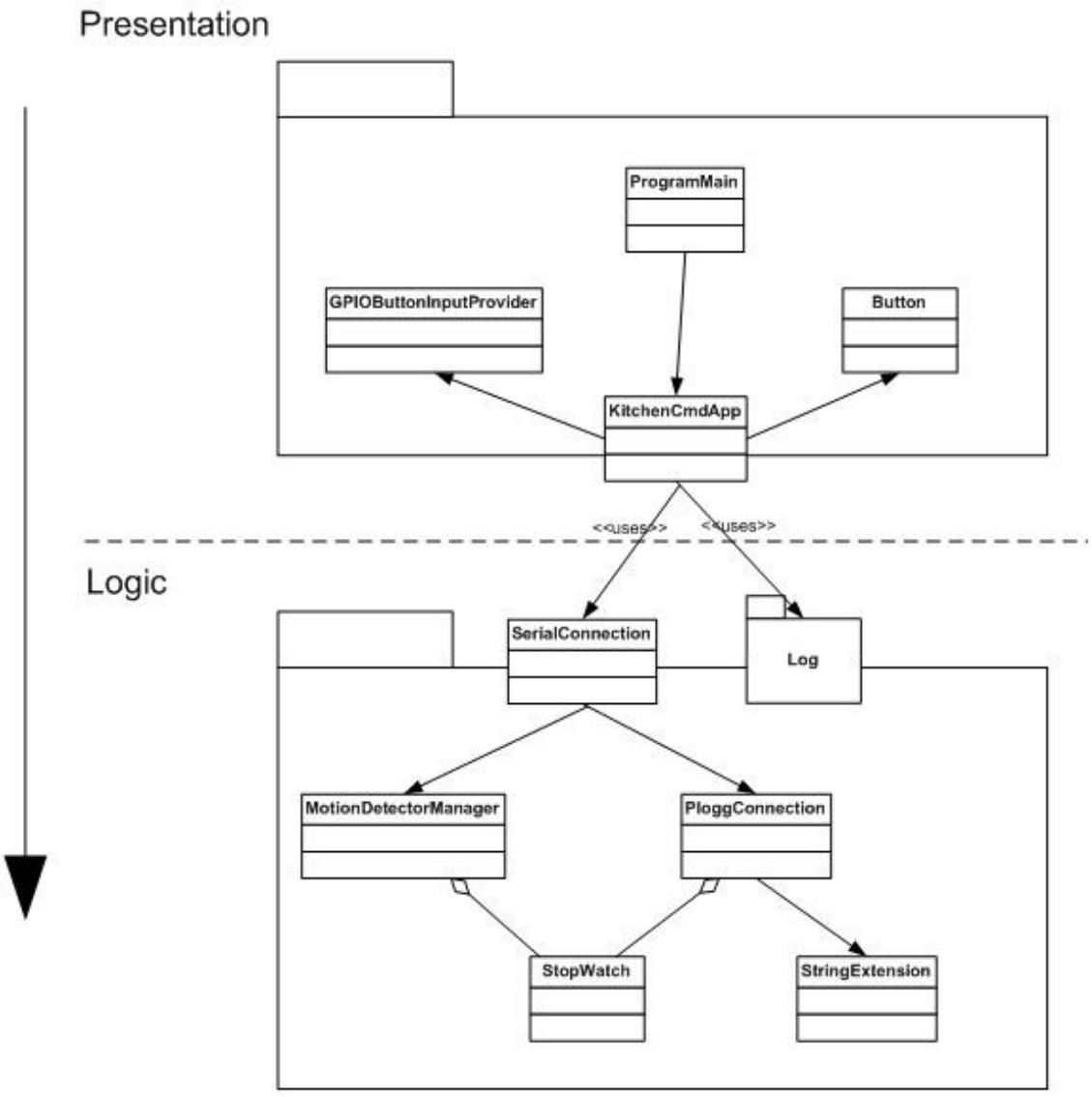
Den overordnede arkitektur for `KitchenCommander` ses på Figur 4. Arkitekturen er lagdelt med et præsentations-lag indeholdende al GUI logik og et Logisk-lag indeholdende den funktionelle logik. Selve kernen i programmet er klassen `KitchenCmdApp`, som også er af typen `Application`. Denne klasse kontrollerer programflowet bl.a. via en simpel statemachine til skift imellem visuelle paneler. Da klassen præsenterer programmet visuelt, kræver dette også adgang til loggen, samt logik for bevægelses-detektor og Plogg, der kører seriel forbindelse via Bluetooth. Dette opnås med adgang til logik-laget, som indeholder denne information.



Figur 4. Overordnet arkitektur

6 Software design

Figur 5 viser klassediagrammet for KitchenCommander, hvor fokus på lagdeling er mindre fremherskende. Hvis vi starter fra toppen, så er `ProgramMain` programmets entry point, hvor selve applikationen startes fra. Som tidligere nævnt er `KitchenCmdApp` selve applikationen og derfor den store kontrol-klasse. Denne benytter klassen `GPIOButtonInputProvider`, som er autogenereret i .NET MF og muliggør input fra fysiske knapper. Endvidere anvendes `Button` klassen, som er lånt fra en øvelse i WEM kurset. Denne klasse udgør det visuelle look af en knap og anvendes i advarselsdialogen.



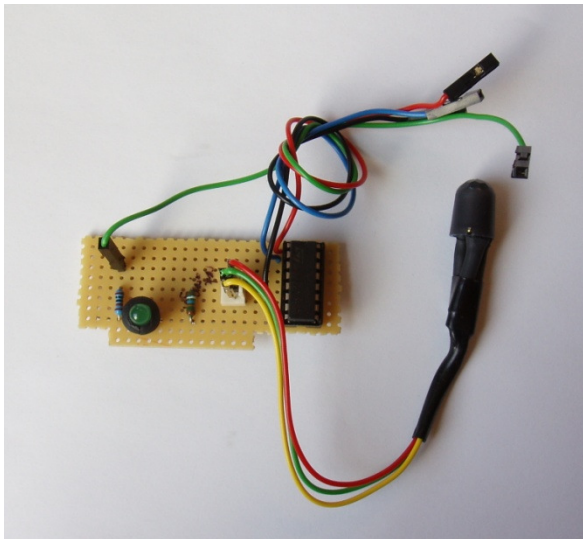
Figur 5. Klassediagram

`KitchenCmdApp` kalder ned i facadeklassen `SerialConnection`, som giver adgang til events fra Plogg og MotionDetector klasserne samt reel data (e.g. StopWatch). Endvidere foregår størstedelen af logningen i `KitchenCmdApp`, så denne anvender også log funktionaliteten fra det underliggende lag. Loggen gemmes på Tahoe II's indbyggede flash hukommelse, og da denne har en begrænset størrelse på 4 MB, har loggen en indbygget begrænsning, som pt. er sat til 50 log beskeder. Når dette tal nås, slettes den ældste halvdel af loggen. Da MF har begrænset indbygget funktionalitet, har vi været nødsaget til at lave vores egen udgave af klassen `StopWatch` samt diverse string

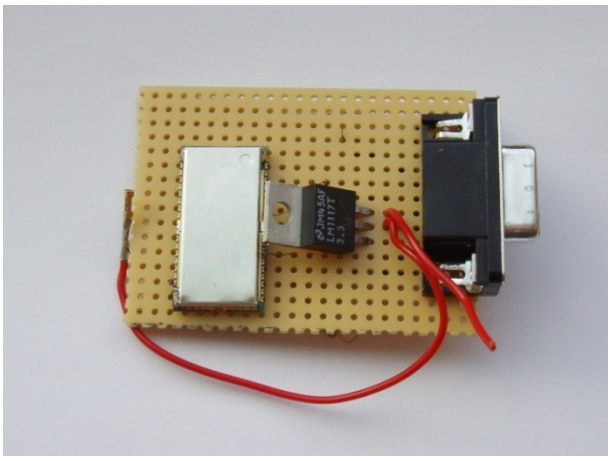
operationer repræsenteret ved klasserne `StopWatch` og `StringExtension`.
`MotionDetectorManager` klassen lytter på en COM port via Bluetooth modulet, og hvis der modtages en kommando, der indikerer en bevægelse, sendes der event ud til alle tilmeldte lyttere (læs: GUI). Et stopur kører løbende og nulstilles hver gang en bevægelse modtages – dette stopurs værdier vises også i GUI'en.
`PloggConnection` klassen har sin egen tråd, som løbende sender kommandoer til Plogg'et via Bluetooth. Klassen modtager også informationer fra Plogg'et om elforbruget. Den kan endvidere tænde og slukke for et forbundet Plogg. Denne klasse har også sit eget stopur, som kører så lang tid, der er strøm på Plogg'et. Endvidere eksisterer endnu et stopur, som kører så længe strømforbruget er over en hvis tærskel, og der ingen bevægelse er detekteret. Begge disses stopurs-værdier vises også i GUI'en. Da Tahoe II ikke har indbygget Real Time Clock (RTC), har vi måttet finde en anden løsning. Heldigvis kunne vi fra Plogg'et modtage korrekt dato og klokkeslæt til vores log beskeder, så `PloggConnection` sørger også for at parse dato samt klokkeslæt og derefter loade dem "ind i" Tahoe II's lokale tid.

7 Hardware design

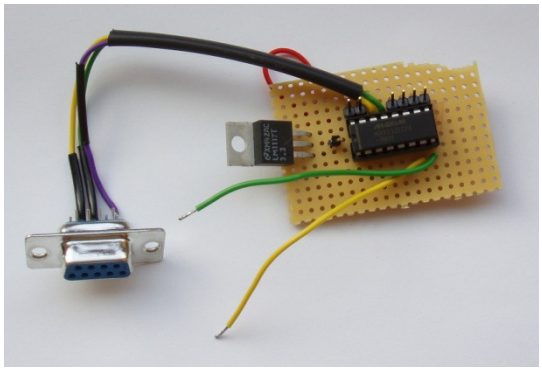
Udviklingen af vores Bluetooth-bevægelsessensor har affødt en masse frustrationer. Selve sensoren er en AMN31111, og virker ved at sende 2,8 V på dens output pin ved bevægelse [4]. Da Tahoe-II IO pins er aktiv lave, skal signalet fra sensoren inverteres. Til dette formål benyttes en inverter, HCF4049UBE. [5] Der er yderligere tilføjet en LED, der lyser, når der detekteres bevægelse. Den har som sådan ikke den store funktion, men har været meget anvendelig under udviklingen. Den er forbundet til en IO pin og styres fra softwaren. Bemærk: Ved power up skal sensoren have op til 30 sekunder, før den er stabil i dens output. Udvikling af denne hardware-komponent var forholdsvis smertefri og ses samlet på Figur 6. Helt anderledes var det med Bluetooth-delen. Vi havde indkøbt 2 forskellige moduler, så der var lidt forskelligt at prøve med. Det første modul, vi prøvede kræfter med, var Rayson BTM-222 [6]. Det blev valgt, da vi havde hørt, at det var blevet brugt i Tahoe-II-sammenhænge, og vi forventede derfor, at der var inspiration at finde på bl.a. internettet.



Figur 6: Motion sensor

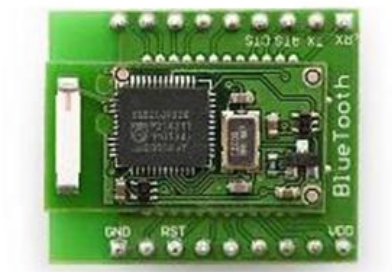


Figur 7: BTM-222 print



Figur 8: TTL-RS-232-adaptor

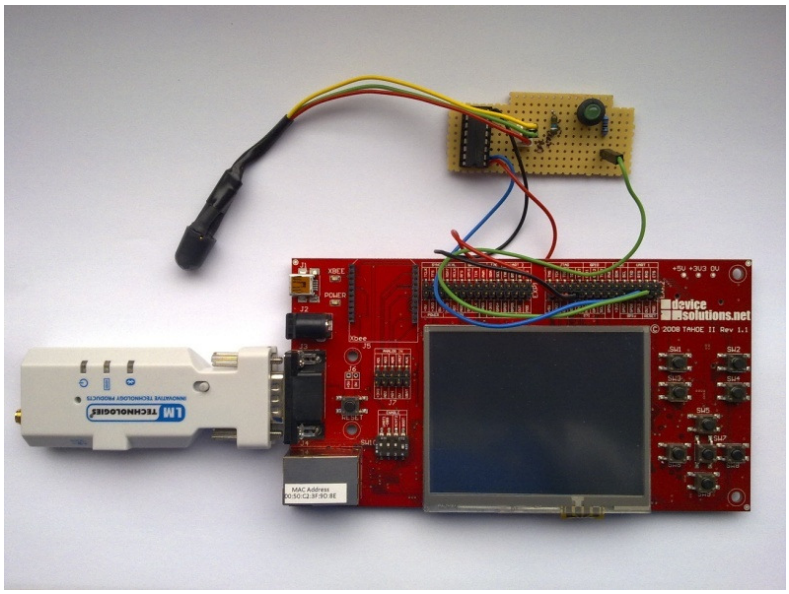
Første udfordring meldte sig ved modulets ankomst, da det var en SMD-udgave, vi fik leveret. Efter rådslagning med Torben Jensen fra elektronik værkstedet, blev der loddet tynde ledninger på de pins, vi skulle bruge. Som vist på Figur 7 blev modulet herefter sat på et hulprint sammen med et D-sub9 stik. Først forsøgte vi at forsyne enheden med strøm fra D-sub9-stikket. Det blev dog hurtigt klart, at der skulle en 3,3V spændingsregulator med i kampen, førend der skete noget. Modulet skulle nu kunne styres med AT-kommandoer, og det var her vi strandede. Det eneste vi fik ud af modulet var "OK" når den fik "AT", alt andet virkede ikke. Heldigvis havde vi bestilt et andet modul nemlig Sparkfun WRL-08461, der ses på Figur 9 [7]. Dette var i DIP-udgave, så det burde være lige til at bruge. Det reagerede overhovedet ikke på kommandoerne. Enten returnerede det intet, eller også en masse tilfældige værdier.



Figur 9. Bluetooth modul WRL-08461

Vi forsøgte at fremstille en TTl-RS-232-adaptor. På Figur 8 ses en tidlig revision af denne komponent. Vi fik dog ikke dette til at fungere.

Tiden var knap, og vi spurgte vores lærere til råds, og de foreslog en kommerciel RS-232-Bluetooth-adapter LM-058 [8]. Dette modul blev afprøvet og efter kort tid kunne vi modtage data fra et Plogg. Tahoe II ses på Figur 10, hvor både LM-058 og bevægelses-sensoren er monteret. Set i bakspejlet var det knap så smart at købe SMD, i det der jo unægteligt ligger et større arbejde frem for en DIP udgave. At ingen af dem alligevel ikke kom til at virke er en anden sag. Det skal dog siges at vi nu ved mere om hardware komponenter (og lodning)end før.



Figur 10. Tahoe II med bluetooth og motion sensor

8 Erfaringer med Micro Framework og Tahoe II

Da MF er en nedbarberet udgave af det fulde framework, som vi normalt bruger, har det krævet lidt tilvænning at anvende MF og dets funktionalitet. Specielt på den grafiske del har der været en del udfordringer, da man bl.a. ikke har en grafisk designer i MF. Generelt er tingene lettere i det fulde framework, da tingene er mere "pakket ind". Fx kræver en simpel opdatering af en tekst, at man manuelt fjerner denne og erstatter den med en ny tekst, hvor man i det fulde framework, har tekstblokke, der let kan opdateres ved at ændre en 'Text' property. Da vi ønskede at benytte touch-skærmen på Tahoe II, krævede dette også nogle tilføjelser/ændringer i koden, som tog en del tid at gennemskue. Det hele handlede om at få touch fokus på paneler og knapper, hvilket ikke var så ligetil. Man kunne foranlediges til at tro, at det var muligt at initialisere applikationen med kaldet. `Touch.Initialize(KitchenCommander)`, og derved ville touch så virke for hele applikationen, men dette var ikke tilfældet. For at få touch fokus på et panel, skal panelet fjernes og tilføjes igen for at kunne modtage inputs. Dermed kan panelet få touch fokus, men dette indebærer ikke, at kontrollerne på panelet kan modtage touch inputs. For at kunne opnå dette, skal kontrollerne tilføjes til panelet, lige efter de hver især er erklæret. Altså virker touch på TahoeII og MF en smule ulogisk opbygget, som forhåbentligt bliver bedre i en fremtidig opdatering af MF eller Tahoe II. Endvidere har vi savnet flere detaljer om exceptions, da man ofte kun får smidt et exception-navn og et stacktrace i hovedet, hvilket ikke altid er tilstrækkeligt med hjælp.

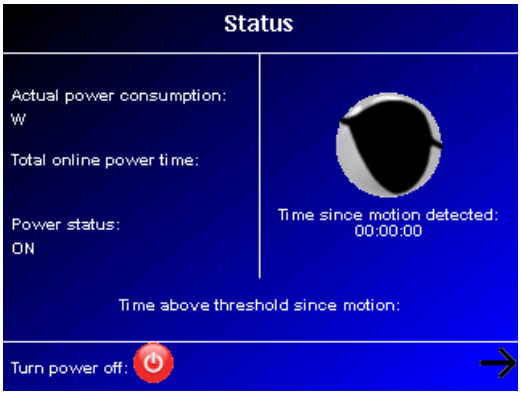
Ud over mærkværdigheder i GUI design, har resten virket relativt logisk. Vi har flere gange manglet funktioner og klasser, hvilket har krævet, at vi har måttet lave vores egne. Dette var vi dog også forberedte på, da frameworket som sagt har mindre indbygget funktionalitet, end vi er vant til. Når den manglende funktionalitet er implementeret, er MF faktisk et behageligt framework at arbejde i, da man kan anvende det, stort set som man plejer. Intellisense er dog savnet, da det er en feature, som højner skrivehastigheden. Dertil skal det dog siges, at Microsoft har formået at integrere MF i VS2008 på intuitiv vis, så det er let at debugge og køre ens applikation på en enhed uden de store problemer. Generelt er det rart at kunne tilgå hardware fra et højt abstraktionsniveau.

9 Brugergrænseflade

Brugergrænsefladen på Kitchen Commanderen er opbygget af flere paneler, der kan navigeres imellem med enten touch eller Tahoe II kortets knapper. Et hovedpanel, som ikke er visuelt, indeholder de 3 paneler, der udgør menuen til Kitchen Commanderen.

9.1 Status

Første panel, der ses på Figur 11, er statuspanelet og det primære panel, som man ofte ville anvende ved brug. Dette panel viser status på det overordnede system og består af følgende paneler:



Figur 11. Status panel

- *Power:* I skærmens venstre side vises information om strømforbruget. Her vises nuværende strømforbrug, hvor lang tid kontakten har været tændt og om kontakten er tændt/slukket.
- *Motion:* Status på bevægelse vises i skærmens højre side. Her kan tiden siden sidste bevægelse aflæses og en synlig indikation gives, når der detekteres bevægelse (Klappen for øjet fjernes og timer nulstilles).
- *Total activity:* Under de to ovenstående paneler vises den samlede tid, der er gået siden sidste bevægelse og hvor strømforbruget har været over en fastsat tærskel. Altså den tid, der er "vigtigst" at bide mærke i.
- *Control:* Dette panel indeholder to knapper, som brugeren kan interagere med. Knappen til venstre kan bruges til at tænde og slukke for strømmen, og skifter tilstand ved tryk. Resten af GUI'en ændres først når strømmen reelt er ændret. Dette gøres for at sikre, at man ikke går med en falsk tryghed om, at strømmen fx er slukket. Til højre er en navigeringsknap, som brugeren kan benytte for at navigere til næste skærmmenu.

9.2 Log

Næste panel er log-panelet, som der vist på Figur 12**,** og viser en oversigt over loggede beskeder fra systemet. Der kan navigeres op og ned via navigationsknapperne eller tilbage til status-panelet med knappen til venstre. Der logges en besked, når der har været inaktivitet ved tændt apparat i et givent tidsrum og brugeren er gjort opmærksom på dette via et advarsels skærm - se næste afsnit. Hvis brugeren reagerer på dette, logges dette også. Hvis ingen reaktion spores inden for et vist tidsinterval, slukkes strømmen og det logges, at brugeren ikke har reageret. Derved kan en hjemmehjælper let danne sig et overblik over den ældres vaner. Dette kan hjælpe til at spore tegn på demens og afgøre om det at være tid for den ældre, at komme på plejehjem.

Figur 12. Log

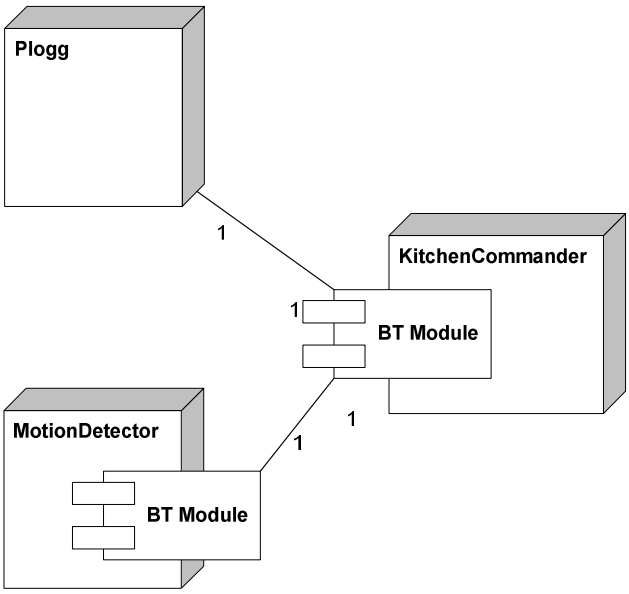
9.3 Advarsel

På Figur 13 ses et billede af den dialog, som vises for brugeren, når der har været inaktivitet ved kogepladen i et vist tidsrum. Den ældre gøres opmærksom på, at kogepladen har været tændt i lang tid uden aktivitet i nærheden. Hvis den ældre er klar over dette og fortsat vil anvende kogepladen, trykkes knappen blot og systemet logger handlingen.

Figur 13. Advarsels dialog

10 Kommunikation

Vores intention med KitchenCommander var fra start af, at den skulle kunne kommunikere med både bevægelsessensor og Plogg og reagere ud fra aflæsningerne af disse, hvilket kan ses på Figur 14. Vi har dog været stærkt begrænsede af, at Bluetooth modulet kun kan kommunikere med én enhed af gangen. Vi afprøvede kort muligheden for runtime at skifte Bluetooth forbindelsen imellem detektor og Plogg, men pga. Bluetooth modulets hastighed ved afbrydelse og oprettelse af forbindelse, som ofte tog op til flere minutter (!), valgte vi at simulere en bevægelse via knaptryk. Det skal dog understreges, at denne beslutning bunder i, at vi blev frataget et af vores Tahoe II kort sidst i udviklingsforløbet, og havde derfor ikke mulighed for at teste systemet med bevægelsessensor tilkoblet via Bluetooth.



Figur 14. Ideelt deployment diagram

Fra KitchenCommander skulle der også sættes en smule op, for at få den til automatisk at forbinde til Plogg/bevægelsessensor.

Dette klarede vi, med hjælp fra følgende AT kommandoer fra Bluetooth modulets manual [6]:

1. *ATRO:* Opsætning af modul til master role
2. *ATD0:* Nulstilling af evt. tidligere indkodet adresse
3. *ATD=xxxxxxxxxxxx:* Indkodning af ny adresse på klient (12 hex cifre)
4. *ATO0:* Enable auto connect til en defineret adresse.
5. *ATD?*: Returnerer indkodet adresse - tjek om ny adresse er gemt

11 Resultater

Som før nævnt har vi kun haft ét Tahoe II til rådighed i den sidste tid af projektforløbet. Dette har medført, at de opstillede Use Cases ikke kan testes i deres fulde udstrækning. Vi har dog ved hjælp af simulering af bevægelsessensor gjort det muligt at teste systemet. Der er opnået Bluetooth kommunikation mellem de forskellige enheder, og den modtagne data behandles og vises korrekt. Den færdige løsningsmodel svarer i store træk til den løsningsmodel, der er illustreret på "Tidligt sekvensdiagram " -Figur 3 i starten af rapporten. Den umiddelbare forskel er at bevægelses sensoren ikke bliver pollet, men selv sender en besked til Kitchen Commanderen, når den har registreret bevægelse.

12 Konklusion

Det ønskede system er i store træk blevet implementeret som ønsket. Projektets krav var, at der skulle inddrages et Tahoe II board, og det er opfyldt, i og med der bruges 2 af slagsen. I udarbejdelsen af Bluetooth bevægelses-sensor er der udført hardware design. Flere forskellige Bluetooth moduler er blevet afprøvet, og desværre er de fleste som nævnt ikke kommet til at virke. Der er dog fundet en kommerciel adapter, der muliggør kommunikation mellem Kitchen Commanderen og de perifere Bluetooth enheder. Dette medførte dog nogle andre problemer, i form af at det benyttede BT moduls reaktionstid, men vi fandt dog en "løsning" på det. Der er designet en grafisk brugergrænseflade, der betjenes via touch og/eller fysiske knapper. Den viser bl.a. det nuværende strømforbrug, hvornår der sidst er detekteret bevægelse, samt system-loggen. Der er implementeret logik, der styrer input og output fra de perifere Bluetooth enheder, så vel som input fra brugeren.

Alt i alt har vi været godt tilfredse med at udvikle i MF til Tahoe II, hvorimod hele Bluetooth delen har været knap så positiv. Vi har opnået en masse erfaringer i kraft af de problemer, der er opstået, og løsninger , der er lavet, og disse vil sikkert kunne bruges til fremtidige udviklingsprojekter.

13 Referencer

1. *"Prevention of Accidents through Pervasive Power Management" + video*; se vedlagt CD
2. *Asus Eee Top*; http://asus.com/product.aspx?P_ID=XPEvtodKRTfnQbCm
3. *Plogg*; http://plogginternational.com/
4. *Motion Sensor AMN31111*; se vedlagt CD
5. *IC HCF4049UBE*; se vedlagt CD
6. *Rayson BTM-222*; se vedlagt CD
7. *Sparkfun WRL-08461*; se vedlagt CD
8. *LM-058*; se vedlagt CD