NTNU
Det skapande universitetet

Utvikling av ei plattform for styring av installasjonar i hus

Ole Morten Haaland

Master i teknisk kybernetikk
Oppgåva levert: Juni 2011
Hovudrettleiar: Geir Mathisen, ITK

Noregs teknisk-naturvitskapelege universitet

Oppgåvetekst

Vi ønsker å utvikle en plattform for styring av installasjoner i hus, basert på et lavpris trådløst nett. Plattformen må inneholde basiselementer for styring og overvåking av tilstander, samt et brukergrensesnitt. Plattformen må kunne være et grunnlag for flere bruksområder, så som styring av lys og varme, energioptimalisering, adgangskontroll osv. I oppgaven skal kandidaten også utvikle et av bruksområdene samt demonstrere egnetheten for andre bruksområder.

1. Foreta en litteraturstudie av systemer for styring av tekniske bygningsinstallasjoner på en planmessig måte.
2. Foreslå et system for styring av tekniske bygningsinstallasjoner for hjem og kontor, som kan masseproduseres.
3. Implementer og test ut hele eller deler av systemet foreslått i punkt 2.

Oppgaven gitt: 10. januar 2011.
Hovedveileder: Geir Mathisen, ITK.

Føreord

Denne oppgåva er gjort som avslutning på sivilingeniørstudiet i teknisk kybernetikk, ved Noregs teknisk-naturvitskaplege universitet våren 2011. Arbeidet skal tilsvara 30 studiepoeng, over eit halvt år.

På mange måtar må dette seiast å vera fortsetjinga av prosjektarbeidet eg gjorde hausten 2010. Resultata då tyda på at ZigBee var eit godt grunnlag for å byggja eit nytt heimestyringssystem, og nettopp det å laga eit slikt system er det denne oppgåva handlar om.

Før denne oppgåva starta hadde eg aldri designa eigne kretskort, og lodda berre store gjennom-hols-komponentar. No, når arbeidet er ferdig, har eg designa fire-lags, eksternt produserte kretskort for radiokommunikasjon, og lodda komponentar utan bein, med 0,5 mm mellom pinnane, det meste utan nevneverdige problem. Dette halvåret har gitt meg meir praktisk erfaring enn alle dei føregåande semestra av studiet til saman.

Ein av lærdomane er at maskinvaredesign alltid tek lenger tid enn ein reknar med. Nettopp difor er systemet ein sit att med ikkje så funksjonsrikt som eg hadde håpa i januar. Likevel verkar det som eit godt grunnlag for vidare arbeid, og eg har difor ambisjonar om å halda fram arbeidet med systemet som eit fritidsprosjekt. Svært mykje av det som ikkje vart så bra som håpa i denne omgang, er definitivt mogleg å gjera noko med.

Det er fleire som fortener takk for all hjelp dei har gitt meg. Rettleiar Geir Mathisen har vist like stor interesse for framgangen og resultata som eg sjølv har hatt. Gjennom timelange møter nesten kvar veke har me hatt mange interessante diskusjonar om korleis eit system som dette bør vera og fungera, og kva som er viktig i rapportskriving. Tusen takk for all hjelp og alle innspel!

Solveig Haukås og Anne Oddveig Haaland har lest gjennom store delar av rapporten, og gjeve verdifulle kommentarar om rettskriving og formuleringar. I tillegg har Solveig blitt utsett for tallause samtalar og diskusjonar om oppgåva, løysingar og problem. Tusen takk for dei timane de har brukt, og alle innspel de har gitt!

Trondheim, 6. juni 2011.

Ole Morten Haaland

Samandrag

I denne oppgåva blir det designa og laga eit system for styring av tekniske installasjonar i bygg. Systemet er basert på ZigBee som kommunikasjonsteknologi.

Fleire forskjellige kommersielle system blir sett på og vurderte. I tillegg ser ein på ei rekke forskingsprosjekt som er relevante for denne typen system. Basert på ei vurdering av desse systema og prosjekta, blir det sett opp eit slags bruksscenario. Dette skisserer eit fullstendig system, og korleis dette kan nyttast.

Fordi arbeidet med denne oppgåva er avgrensa i tid, ser ein det ikkje som realistisk å implementera eit komplett system, som skissert i bruksscenarioet. I staden er det ønskeleg å laga eit sett basekomponentar som kan fungera i seg sjølv, men som òg kan vera ei plattform for andre applikasjonar. Dette systemet blir basert på eit desentralisert konsept, der all konfigurasjon ligg ute i dei enkelte komponentane. Ein vil likevel ha ein sentral i systemet, men denne skal primært nyttast til konfigurasjon og eventuell sentralisert, automatisert styring.

For å få eit system som er brukbart i seg sjølv, vel ein i denne omgang å laga ein brytarnode, med knappar som kan fungera som inngangar i systemet, ein relénode som kan slå av og på opp til 16 A ved 240 V, og ein sentral. Denne sentralen blir basert på Beagleboard -xM, eit enkelt kort med ein komplett ARM-datamaskin. Sentralen skal tilby styring og konfigurasjon gjennom eit web-grensesnitt, for å vera tilgjengeleg frå flest mogleg einingar.

Systemet blir implementert med ein Atmel Atmega128RFA1 som mikrokontrollar og kommunikasjonsbrikke. Denne mikrokontrollaren har innebygd fast minne, EEPROM, som dermed kan nyttast til å lagra konfigurasjonen. Lysbrytarnoden blir drive av to AAA-batteri, medan relénoden har si eiga, transformatorbaserte straumforsyning.

Produksjon av einingar og kretskort for testing blir delvis gjort på verkstaden til Institutt for teknisk kybernetikk, og delvis av eksterne produsentar. Det blir fokusert på å gjera programvara lett distribuerbar, og lett å vidareutvikla, gjennom bruk av pakkesystem på Beagleboardet.

Testing av dei ferdige einingane gir delte resultat. Batterilevetid for lysbrytarnoden viser seg til dømes svært god, med håp om godt over 3 år levetid i faktisk bruk. Òg faktisk bruk av systemet fungerer bra, og gir til dømes korte responstider frå brytartrykk til faktisk tilstandsendring.

Det mest alvorlege problemet med designen er truleg rekkevidde. Truleg har tilpassinga av impedansen i antennebanane ikkje vore god nok, og

einingane klarer berre å kommunisera over avstandar på ein til to meter. Dette kan truleg fiksast ved å endra kretskortdesignen noko. Fordi resultatet var godt nok for testing, og det var avgrensa tid tilgjengeleg, har dette ikkje blitt gjort i denne omgang.

Totalt sett er resultatet difor ikkje perfekt, men det framstår som eit godt grunnlag for vidare arbeid. Det blir òg skissert ein del aspekt ved systemet som kan eller bør forbetrast. Konseptuelt sett verkar det å ha ein kraftig sentral og enklare, ZigBee-baserte einingar godt.

Innhald

1. Innleiing

1.1. Bakgrunn

Det er i år om lag 131 år sidan elektrisitet vart innført i Noreg, og det første denne vart brukt til i private heimar, var lys. [1] På mange måtar har styring av lys, og for så vidt av andre elektriske einingar, likevel endra seg svært lite på den tida: brytaren på veggen koplar framleis fysisk frå eller til den tilkopla lampa, og det er alt som finst av styringsmoglegheiter.

I løpet av desse åra har det vore gjort fleire forsøk på å laga styringssystem, nokon med suksess, andre utan. X10 er eit eksempel på ein relativt gammal teknologi, frå 70-talet, som i alle fall i enkelte miljø har hatt ein viss suksess. KNX har nok vore brukt i ein del større installasjonar, men ingen av dei har klart å gjera store innhogg i marknaden.

Eit felles problem er kostnadar. Dei fleste eksisterande system har vore svært kostbare, samtidig som dei ikkje har gitt så mange moglegheiter som ein kanskje kunne ønska eller sjå føre seg. Mange av dei har vore bundne til kablar, som dermed har bidrege til høge installasjonskostnadar, eller vore trådlause, men basert på upålitelege kommunikasjonsteknologiar, slik at systema ikkje har vore til å stola på i dagleglivet.

Med dei siste åra sine utviklingar innan lågstraums, lågpris, trådlause teknologiar, burde det vera mogleg å gjera noko betre, og det finst minst to grunnar til at det er ønskeleg å ha meir avansert styring.

For det første har det blitt eit sterkare fokus på dei miljømessige aspekta ved straumforbruk dei siste åra. Med ei form for sentral styring, er det langt enklare å slå av dei einingane som ikkje er i bruk akkurat no. Til dømes kan ein ha ein «god natt»-knapp ved senga, som automatisk slår av alt i huset som bør vera av om natta. Ein slik fordel gir òg ei økonomisk innsparing.

For det andre har avansert styring definitivt òg ei komfort-side. Den nemnde «god natt»-knappen, er i tillegg til å vera ein miljømessig fordel, òg ein praktisk fordel. Dessutan kan ein tenka seg trådlause brytarar, som kan plasserast på meir fornuftige plassar enn dei relativt rigide kabel- og straumleidningsbundne brytarane me er vande med.

1.2. Motivasjon

Personleg ligg motivasjonen i eit ønske om å ein dag kunna ha eit hus der styring av lys er noko meir enn det å mekanisk kopla frå og til straumen til lyspæra. Både det å kunna «slå av» huset om kvelden, «slå på» når ein kjem inn døra, eller kanskje endåtil frå smarttelefonen, verkar forlokkande.

Om eit slikt system i tillegg kan vera ope, på ein måte som gjer at både interesserte privatpersonar og andre bedrifter kan samarbeida om å laga nye bruksområde, blir det truleg endå meir fristande. Å få gode resultat i denne oppgåva er difor ikkje berre motivert ut frå det reint utdanning- og karrieremessige, eit større mål er å faktisk få eit brukbart system ut av arbeidet.

1.3. Scenario

For å kunna laga eit godt design, er ein avhengig av å ha klart føre seg kva ein siktar seg inn på. Det er stor skilnad på ei enormt kontorbygg og ei to-roms leiligheit.

For eit stort bygg er kanskje det viktigaste å kunna gjera sentralisert styring, slik at ein kan gjera energiioptimalisering på ein enkel måte. Vidare vil ein kanskje ha støtte for styring av ventilasjonsanlegg, kombinert med oppvarming. Å kunna styra bygget frå andre stadar, er derimot kanskje heilt unyttig.

For ei lita leiligheit er det kanskje heller motsett. I slike finst sjeldan noko ventilasjonsanlegg i det heile, og det primære argumentet for eit slikt system er kanskje komfort, i det at ein kan ha trådlause brytarar. I ei hytte kan ein tenka seg nettopp fjernstyringa som den essensielle eigenskapen.

I oppgåveteksta heiter det at systemet her skal lagast for «hjem og kontor», og dette kan tolkast som at det er småskalaanlegg det er snakk om, kanskje med eit hundretals einingar. Eit slikt system vil truleg òg kunna vera nyttig for mindre kontor, sjølv om det primære fokuset ligg på private heimar.

1.4. Avgrensingar

Ei oppgåvetekst vil alltid innehalda rom for tolking. Med dette avsnittet er målet å klargjera på kva måte oppgåva har blitt forstått, og kva ein ser på som meir og mindre viktig.

Til dømes er ordet plattform nytta i oppgåveteksten. Dette ordet kan tolkast i fleire retningar, men det har her blitt forstått som eit grunnlag å

byggja applikasjonar på toppen av. Dette gjer til dømes at ein har søkt å gjera koplingane mellom ulike delar av systemet relativt lause, slik at ein kan nytta nokon delar av resultatet uavhengig av andre.

Likevel finst det her grenser for kor mykje abstrahering som har blitt gjort. Delar av systemet køyrer på små mikrokontrollarar, der ein ikkje har uavgrensa med ressursar og minne tilgjengeleg. Av den grunn vart det valt å ikkje gjera nokon abstraksjonar på toppen av programvareløysinga som vart nytta for ZigBee-kommunikasjon, sjølv om dette kanskje kunne gjort vidare bruk enklare.

Vidare heiter det at eitt av bruksområda skal utviklast, og at andre bruksområde skal demonstrerast. Ein har difor valt å å laga eit system som er klart til bruk for styring av lys og varme. Andre bruksområde er ikkje utvikla i fysiske einingar, men blir skissert i tekst.

Delsetninga om masseproduksjon er òg viktig å forstå. På mange måtar står dette i kontrast med målet om å laga ei plattform, sidan masseproduksjon aller helst krev spesialiserte einingar og kort, for å halda kostnadar nede. Ein har difor måtta gjera enkelte kompromiss, til dømes i kortdesign. Dette kjem ein nærare inn på i dei kapitla som omhandlar dette.

I prosjektoppgåva i haust vart fleire ulike teknologiar å basera eit slikt system på vurdert. Av desse verka ZigBee å vera den mest interessante. Denne oppgåva byggjer difor på dette arbeidet, i det at målet er å implementera eit heimestyringssystem, basert på nettopp ZigBee.

1.5. Oppbygging av oppgåva

Oppgåva startar med eit blikk på kommersielt tilgjengelege system, i forhold til pris, moglegheiter og teknologi. Vidare er det interessant å sjå på kva forsking som skjer innan området heimestyring, og kva ein ser for seg skal skje i framtida. I vurderinga av kva prosjekt ein skulle sjå på, har ein nytta heimestyring i vid forstand. Ein del av prosjekta inneheld difor ikkje ting som er direkte nyttige, men kan likevel vera interessante for å visa kva som faktisk blir forska på.

Basert på ein del av forskingsprosjekta og dei eksiterande systema, blir det difor laga eit sett krav, som verkar fornuftige for eit slikt system. Desse krava blir så nytta i designen av eit nytt system for husstyring.

Kapitlet «Implementasjon» ser på dei meir praktiske sidene ved å laga systemet, både frå programvare- og maskinvaresida. Det ferdige systemet blir så testa opp mot dei tidlegare oppsette krava.

Gjennom arbeidet viser det seg ein del punkt som kunne vore gjort på ein betre eller ein annan måte. Dette, saman med korleis ein kan sjå for seg

å nytta systemet som ei plattform for andre bruksområde, blir lagt fram i «Diskusjon».

Avslutningsvis er det naturleg å sjå på moglegheiter for vidare arbeid, og gjera ei oppsummering av arbeidet som er gjort.

Vedlegga inneheld meir tekniske detaljar, både i forhold til verktøy som er brukt og design som har blitt produsert. Innhaldet på den vedlagte CD-en blir òg forklart her.

2. Eksisterande husstyringssystem

Det er naturleg å ta utgangspunkt i eigenskapane til eksisterande system for å byggja eit nytt system for husstyring. I tilegg finst det ei rekke aktuelle forskingsprosjekt på området det er naturleg å sjå på. Trass i at dei foreløpig ikkje har materialisert seg som kommersielle system, inneheld mange av dei interessante tankar og konsept.

2.1. Kommersielle system

Med kommersielle system, meiner ein her system som er moglege å få tak i for sluttbrukarar, men ikkje nødvendigvis i Noreg. Dei følgande avsnitta er delvis baserte på prosjektoppgåva, der kommunikasjonsmetodar og -teknologiar i dei ulike systema er dekka meir inngåande. [2]

Det finst ei rekke aspekt ved desse systema det hadde vore svært interessant å sett nærare på, særleg i forhold til responstider, batterilevetider og praktisk bruk. Denne typen informasjon er ikkje tilgjengeleg om fleire av systema, og kombinert med lite samarbeidsvillige leverandørar, gjer det at ein del opplysningar dessverre manglar her.

2.1.1. Control4

Control4 er eit amerikansk selskap som lagar eit ZigBee-basert husstyringssystem under same namn. Systemet prøver å vera ei løysing for alle typar styring i ein heim eller eit kontor, både lys, varme, lyd, bilete, låsar, alarmar og kamera.

Control4 sitt system er bygd opp rundt einingar dei kallar «Home controller». Det verkar som ein kan ha ein eller fleire slike, men nøyaktig kva rolle desse har i den daglige drifta av systemet, er noko uklart i dokumentasjonen som finst. Forespurnadar til Control4 har heller ikkje gitt svar, men det verkar som om systemet krev minst ein slik kontrollar for å fungera. Kva som skjer om denne er av, eller øydelagt, er dermed vanskeleg å seia.

Control4 er lagt opp til at kvar kunde skal få tilpassa og sett opp systemet av ein lokal forhandlar. Oppsett av systemet skjer ved hjelp av spesiell

Figur 2.1.: Forskjellige måtar å styra eit Control4-system på. Henta frå [3].

programvare, og i utgangspunktet er det berre godkjende installatørar som kan gjera dette.

Styring av systemet kan derimot skje frå ei lang rekke ulike einingar: spesielle trådlause og faste einingar med berøringsskjermar, applikasjonar for telefonar og datamaskinar, ein nettportal, fjernkontrollar med og utan skjerm samt veggbrytarar som liknar på tradisjonelle lysbrytarar. I tillegg kan ein kopla eit fjernsyn til ein «Home controller» og styra systemet ved hjelp av dette.

Det ser ut til at Control4 har valt å nytta det same brukargrensesnittet på alle einingar som har skjerm, inkludert i mobilapplikasjonar som kan kjøpast for å styra huset. Dermed blir det enklare å læra seg å nytta systemet. For einingar utan skjerm, blir respons gitt gjennom lysdiodar. I enkelte tilfelle ser det òg ut til at ein kan sjå resultat av kommandoar på til dømes fjernsynsskjermen når ein trykker på ein fjernkontroll.

I eit Control4-system er det ingenting utanom fjernkontrollane som er heilt trådlaust. Til dømes krev veggbrytarane som erstattar vanlege lysbrytarar ei 36 V spenningsforsyning, noko som gir ekstra kabling, men gjer at ein slepp å skifta batteri. Uansett går all styringsinformasjon trådlaust. [3]

For å styra eit hus utanfrå må ein abonnera på ei betalingsteneste kalla 4Sight. Denne gir moglegheit til å styra eit hus utanfrå, tilsynelatande gjennom dei same applikasjonane som ein kan nytta internt i huset. I tillegg til styring av huset, gir dette systemet òg moglegheit for å sjå på videostraumar frå huset.

Control4 tilbyr òg 4Store, ein slags programvarebutikk, der ein kan kjøpa applikasjonar som utvidar funksjonaliteten til systemet. I denne butikken finst det mange ulike typar applikasjonar, både betalvare og gratisvare, frå program for administrasjon av dørlåsar til reine spel. Nokon av applikasjonane krev spesielt oppsett, og kan difor berre installerast av godkjende

forhandlarar.

I tillegg til Control4 sine eigne løysingar, er butikken òg open for uavhengige utviklarar. Ved å nytta seg av dette, kan andre bedrifter laga ei fysisk eining, med tilhøyrande programvare for å realisera funksjonalitet Control4 ikkje tilbyr. Det er uklart om det er noko krav til godkjenning for slike løysingar, men Control4 tilbyr ei sertifisering av fysiske produkt som kan nyttast med systemet. Det er dermed mogleg for andre å tilby ei kombinert løysing av programvare og maskinvare. [4]

Fordi ZigBee som teknologi har låg bandbreidde, nyttar Control4 Ethernet eller WiFi[1] i mange av einingane i systemet, for å overføra bandbreiddeintensive data, som video og lyd.

Systemet blir tilsynelatande primært marknadsført mot det amerikanske marknaden, og det er difor vanskeleg å vurdera prisnivået på ein samanliknbar måte. Det finst nettbutikkar som forhandlar Control4-komponentar, med prisar tilgjengeleg, men det er usikkert om desse er representative for det generelle prisnivået. I alle fall er ulike typar heimekontrollarar der tilgjengelege frå 450 til 1600 amerikanske dollar, og brytarar og utgangar til om lag 130 dollar. [5] Med dagens dollarkurs blir dette frå litt under 2500 til opp mot 9000 kroner for sentralen, og om lag 700 kroner for brytarar eller utgangar. I tillegg kjem norsk moms og eventuell montering.

2.1.2. Nobø Orion

I motsetnad til Control4, er Nobø Orion eit heimestyringsystem som primært fokuserer på varmestyring. Systemet vart laga av den norske bedrifta Nobø, men denne har seinare blitt kjøpt opp og skifta namn til Glen Dimplex Nordic. All styringskommunikasjon i systemet går trådlaust over ein udokumentert, proprietær protokoll i 868 MHz-bandet.

Hovudkomponenten i systemet blir kalla Orion 700. Denne fungerer tilsynelatande som hjerne i systemet, og gir det meste av funksjonaliteten. Som mottakarar finst det termostatmodular for varmeomnar som Glen Dimplex produserer, samt generelle utgangsrelé, anten innebygde eller utanpåliggande. Sidan fokus er på varmestyring, finst det ingen modular som kan nyttast for til dømes dimming av lys.

Systemet er tenkt slik at ein programmerer det éi gong, og deretter berre unntaksvis gjer endringar i oppsettet. Likevel er det tilsynelatande ingenting i vegen for at ein som sluttbrukar sjølv kan gjera endringar, i motsetnad til Control4. All programmering skjer frå sentraleininga, vist i figur 2.2. Brukargrensesnittet på denne består av ein relativt liten, tekstbasert

[1] Vanleg, 802.11-basert trådlaust nettverk

(a) Sentraleininga, Orion 700. (b) Veggmontert termostat. (c) Omnmontert termostat.

Figur 2.2.: Ulike brukargrensesnitt for Nobø-systemet.

skjerm, samt nokon knappar. Det finst ikkje moglegheit for å programmera systemet frå til dømes ein PC.

Styringa i Orion-systemet er bygd opp rundt to temperaturinnstillingar: komfort-temperatur og sparetemperatur. Tanken er at ein lagar seg eit vekeprogram som definerer når dei to temperaturane skal nyttast, og at systemet deretter regulerer temperaturen på eiga hand.

På ein del av omn-termostatane stiller ein inn dei ønska temperaturane ved hjelp av skyvebrytarar. På dei veggmonterte termostatane finst eit sett knappar og eit vindauge som viser gjeldande innstilling. Begge desse termostattypane er viste i figur 2.2.

I tillegg til mottakarar som kan koplast til omnar og generelle stikkontaktmottakarar, finst det mottakarar som kan koplast til golvvarme. Dette gjer det mogleg å ha styring òg på varmekablar frå sentralen, men det er uklart om det gir styring over termostaten eller berre moglegheit til å slå av og på kablane.

Orion 700 gir moglegheit til å knytta dei ulike mottakarane saman i soner, og dermed gjera styring på sonenivå. Frå nokon typar mottakarar kan ein overstyra vekeprogrammet, og tvinga ein sone i ein spesiell modus. Dette er òg mogleg frå sentralen, både for individuelle sonar og for heile systemet. Ein kan òg setja systemet i frostsikringsmodus eller slå det heilt av. [6]

Ved å kopla Nobø-systemet saman med ein GSM-mottakar, kan ein få moglegheit til å styra frå mobiltelefon, ved hjelp av tekstmeldingar. Systemet kan òg knyttast opp mot ein internettportal, kalla «HyttaMi», for fjernstyring frå PC. Dette gir berre heilt enkel av/på-funksjonalitet, og ingen ekstra styringsmoglegheiter utover det ein kan gjera frå sentralen sjølv. I tillegg til styring, kan GSM-mottakaren overvaka temperatursensorar og straumtilstand, og varsla brukaren med tekstmeldingar ved gitte temperaturar eller ved straumbrot. [7]

(a) Veggmontert, dobbel brytar.

(b) Avansert fjernkontroll.

Figur 2.3.: Døme på einingar som kan nyttast til å styra NEXA-systemet.

Prisen for ein Nobø Orion 700 ligg i overkant av 3000 kroner, inkludert norsk moms. Ulike utgangstypar ser alle ut til å ligga rundt 700 kroner, medan den meir avanserte termostaten vist i figur 2.2 ligg i overkant av 2000 kroner. [8] Alt dette er likevel primært meint for å gjera varmestyring.

2.1.3. System NEXA

I motsetnad til Control4 og Nobø Orion, er System NEXA eit svært enkelt system. NEXA er ei svensk bedrift som produserer mange ulike elektriske produkt. Dei har fleire ulike husstyringssystem, nokon som nyttar 433 MHz-bandet og nokon som nyttar 868 MHz-bandet.

I NEXA-systemet finst det ingen sentral kontrollar, all kommunikasjon er direkte samband mellom brytaren og det som skal styrast. Systemet tilbyr ulike typar brytarar og fjernkontrollar, samt mottakarar med og utan dimmar, innebygde og utanpåliggande. Heile systemet er marknadsført direkte mot sluttbrukar, og i Noreg får ein til dømes kjøpt systemet på butikken JULA.

Fordi ein ikkje har nokon sentral i systemet, må oppsettet gjerast annleis enn for Control4 og Nobø. Metoden er svært enkel: For å la ein ny kontrollar styra ei eining, set ein eininga i ein spesiell innlæringsmodus, og aktiverer den kontrollen ein vil skal styra eininga. Dermed er dei knytta saman, slik at den gitte kontrollen styrer den gitte mottakaren. Dei fleste mottakarane i systemet har fleire minneplassar, slik at same mottakar kan styrast av fleire kontrollar. [9]

Nokon variantar av System NEXA har òg inngangseiningar som ikkje direkte brukarstyrt, som til dømes skumringsrelé, rørslesensorar og magnetbrytarar. Desse programmerer ein på liknande måte som vanlege bry-

tarar, ved å setja mottakaren i ein spesiell modus og deretter trykka på ein læringsknapp på sensoren. [10]

NEXA produserer ei rekke andre elektriske produkt, mellom anna dørklokker. Nokon av desse kan òg nyttast som inngangar i systemet, slik at ein til dømes kan la lyset blinka når det ringer på døra. Systemet er òg utvidbart med styring frå mobiltelefon, ved hjelp av ein ekstra kontrollar. Dette gjer det mogleg å styra systemet gjennom tekstmeldingar. [11]

Prismessig er NEXA-systemet det absolutt rimelegaste, og det systemet med minst funksjonalitet. Fjernkontrollen som er vist i figur 2.3 ligg på knapt 300 kroner, medan utgangar og brytarar ligg rundt 200 kroner.

2.2. Aktuelle forskingsprosjekt

Det finst ei rekke forskingsprosjekt rundt intelligente hus, som inneheld punkt det kan vera verdt å sjå nærare på. Nokon av desse prosjekta blir gjennomgått nedanfor.

2.2.1. AMIGO

AMIGO er eit prosjekt som gjekk frå 1. september 2004 til 29. februar 2009, og vart delvis finansiert av EU gjennom det såkalla «6th framework programme». Mellom deltakarane fann ein både Philips, Microsoft og ei rekke europeiske universitet. Målet med prosjektet var å ta nettverksfunksjonalitet ut av PC-til-PC-domenet, til å integrera òg heimestyring, mobile einingar og heimeunderhaldning. Dette skulle gjerast ved å utvikla open kjeldekode-løysningar for å kommunisera mellom slike einingar. [12]

Denne programvareløysinga opptrer som mellomvare mellom ulike einingar. Programvareløysinga er publisert som kjeldekode, i fleire ulike programmeringsspråk. Delar er programmert i Java, medan andre delar er laga ved hjelp av .NET-rammeverket frå Microsoft. Dei ulike delane er òg dekka av ulike lisensar. Nokon av lisensane forbyr kommersiell bruk av programvara, og gjer det dermed vanskeleg å bygga vidare på systemet. [13]

AMIGO er bygd opp av fleire ulike modular. Til dømes finst det modular som tek inn data frå fysiske sensorar og brukarhendingar, og basert på desse konstruerer ein «context» som identifiserer ein systemtilstand. Andre einingar kan dermed spørja denne modulen om kva som er gjeldande «context», og gjera val basert på dette. Dette gjer det til dømes mogleg å gjera filtrering, slik at ein berre presenterer val som faktisk er moglege å gjennomføra i gjeldande tilstand.

Tilsvarande finst det modular for å gje beskjedar til brukaren, halda styr på medieinnhald, brukarprofilering og meir. Ved hjelp av «context»-modulen, kan meldingar bli levert der brukaren faktisk er.

Kommunikasjon mellom ulike system er i AMIGO delvis basert på UPnP[2] og delvis på reine web-tenester, som kommuniserer ved hjelp av XML. [14]

I det heile spesifiserer AMIGO ei rekke høgnivå-tenester, der det er litt uklart kor mykje av desse som faktisk er implementert og fungerande. Eit døme på ei slik teneste er tenkt å bidra til betre kommunikasjon mellom menneske som bur på ulike stader. Dette er tenkt gjort ved å gjera det mogleg å kommunisera mellom alle typar einingar. Til dømes kan ein tenkja seg ein videosamtale frå telefon til fjernsyn, eller moglegheit for å spela virtuelle spel saman, eller sjå på bilete saman. Dette skal bidra til å viska ut fysisk avstand, slik at ein kan halda betre kontakt med til dømes familie som bur andre stader enn ein sjølv. [15]

Generelt verkar det som om fokuset i AMIGO hovudsakleg har vore å laga eit grunnlag for å byggja tenester på, og ikkje det å faktisk implementera tenestene. Fordi ein del av scenaria som er brukte krev tilpassa maskinvare, er det heller ikkje enkelt å testa det.

I tillegg gjer bruken av Java og .NET det vanskeleg eller umogleg å nytta AMIGO-plattforma direkte på mikrokontrollarar.

2.2.2. SM4ALL

SM4ALL er eit prosjekt som vart starta 1. september 2008, og skal vara i tre år. Prosjektet er delfinansiert av EU gjennom «7th Framework Programme», og har både universitet og bedrifter som medlemmar.

SM4ALL er ei forkorting for «Smart houses for all», og prosjektet liknar på AMIGO i det at det hovudsakleg går ut på å laga ei plattform for å byggja smarthus på. Eit aspekt SM4ALL har, som AMIGO ikkje har, er eit fokus på at systemet skal vera tilgjengeleg for alle. Dette gjer at SM4ALL som ein del av prosjektet føreslår eit hjerne-maskin-grensesnitt. Eit slikt grensesnitt gjer det vanskelegare å behandla mange ulike val, og det blir dermed endå viktigare å gi intelligente alternativ til handlingar i ulike situasjonar. [16]

I introduksjonen til prosjektet, blir det sett opp fire punkt som er viktige for eit smarthus: [17]

dynamicitiy: Systemet må handtera det at sensorar og utgangseiningar ikkje er faste, men at det kontinuerleg kan dukka opp nye og forsvinna gamle. Ved å tilpassa seg dette, og tilpassa seg brukarhandlingar

[2] Universal Plug and Play

og -vanar, kan systemet stadig tilby nye tenester, basert på kva som er tilgjengeleg akkurat no.

scalability: Systemet må skalera godt, og handtera hundrevis av einingar utan problem.

dependability: Ein må alltid kunna stola på at systemet fungerer som det skal.

security and privacy: Det er svært viktig at systemet er sikkert, og at det ikkje lek informasjon.

Om ein ser på arkitekturen i SM4ALL, er denne bygd opp av fleire komponentar. Mest grunnleggjande er sensorar. Desse tilbyr sine funksjonar gjennom eit tenesteorientert grensesnitt direkte tilgjengeleg for alle andre einingar. Dermed skal dei kunna knyttast saman og nytta funksjonar frå andre sensorar utan sentral styring. SM4ALL opererer med både distribuert og sentralisert styring. Utan at skiljet er klart, skal det både gå an å koordinera fleire einingar i ei eining, i tillegg til at det skal finnast ein sentral komponent som kan gjera meir komplisert styring.

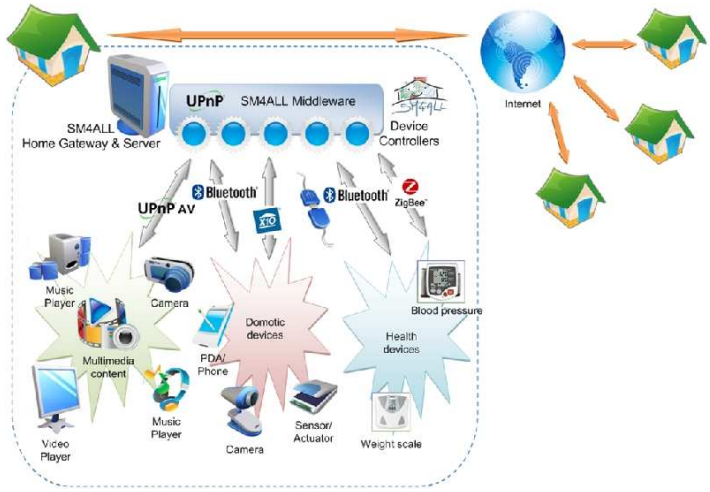
Teknisk sett ser ein for seg at denne sentralen køyrer SM4ALL-programvare, som nyttar UPnP for å knytta saman ulike delar av systemet. I tillegg skal denne sentralen ha bruer til andre kommunikasjonsteknologiar, slik at systemet kan byggjast opp av fleire delsystem, basert på ulike teknologiar, slik figur 2.4 viser. [18]

SM4ALL fokuserer sterkt på å sentralisera ein del tenester i systemet. Til dømes ser ein for seg at element i brukargrensesnittet skal veljast ut sentralt, og at den enkelte eininga berre viser fram det han får frå den sentrale tenesta. Denne strukturen skal gjera det enkelare å tilpassa dei tilgjengelege vala til gjeldande situasjon, og å gi dei same vala uavhengig av kva eining ein nyttar.

Sjølv om det ut frå nokon av rapportane ser ut som om det har blitt utvikla programvare, er denne tilsynelatande ikkje offentleg tilgjengeleg. Det er heller ikkje mogleg å finna ut noko om, eller eventuelt når, denne vil bli tilgjengeleg.

2.2.3. eDIANA

eDIANA er eit prosjekt som har det å redusera energibruk som hovudfokus. Det fulle namnet er «Embedded systems for energy efficient buildings», og prosjektet vart starta i februar 2009 og skal vara i tre år. Prosjektet er delfinansiert gjennom Artemis Joint Undertaking, og har hovudsakleg bedrifter som deltakarar, med Philips og ST Microelectronics som dei mest kjende.

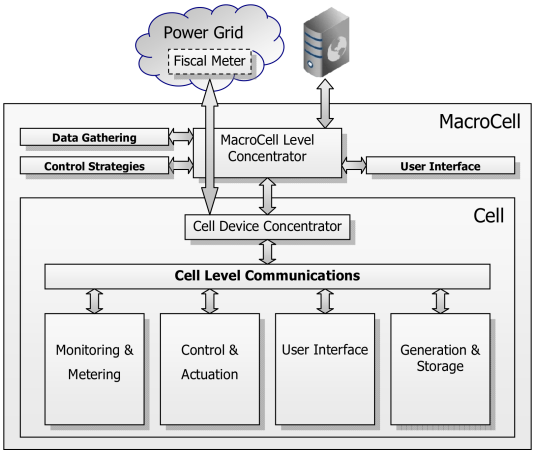


Figur 2.4.: Arkitekturen i eit SM4ALL-system. Denne viser klart korleis ein ser for seg ein SM4ALL-server, som overset mellom ulike teknologiar, delvis ved hjelp av UPnP. Henta frå [18].

eDIANA seier at teknologien som skal utviklast skal redusera energibruken i bygningar med 25 %, i tillegg til å auka komforten for brukarane. Dette skal skje ved hjelp av innebygde, intelligente system, som samarbeider for å rasjonalisera og prioritera energibruk. [19] Det er uklart om dette er snakk om reduksjon i gjennomsnittleg forbruk eller om det dreier seg om å redusera toppar i forbruk.

eDIANA er bygd på strukturen vist i figur 2.5. Her er dei grunnleggande konsepta celle og makrocelle. Ein tenkjer seg at makrocella representerer den som har kontrakt med til dømes straumleverandøren, og at vanlege celler er undereiningar av ei makrocelle. Dermed kan ei celle vera eit hus, ein leilegheit, ein butikk eller liknande, og består av alle einingane som finst i denne. Ei makrocelle kan vera ei større samling med celler, til dømes eit leilegheitsbygg eller liknande. Styringa er delt på tidsnivå mellom makrocella og cella: Ei makrocelle har ansvar for den langsiktige styringa, som kvar enkeltcelle så brukar som grunnlag for si sanntidsstyring av dei enkelte einingane i denne cella. [20]

Som prosjekt skal eDIANA resultera i det dei kallar ein «modellbasert referansearkitektur», ved hjelp av «opne mellomsystem». Her ser ein for seg å laga modellar for korleis til dømes eit hus brukar straum, og basera val i prosjeket på korleis denne modellen oppfører seg i ulike tilstandar. Dette håper ein skal gi systemet moglegheit til å gjera meir intelligente val.

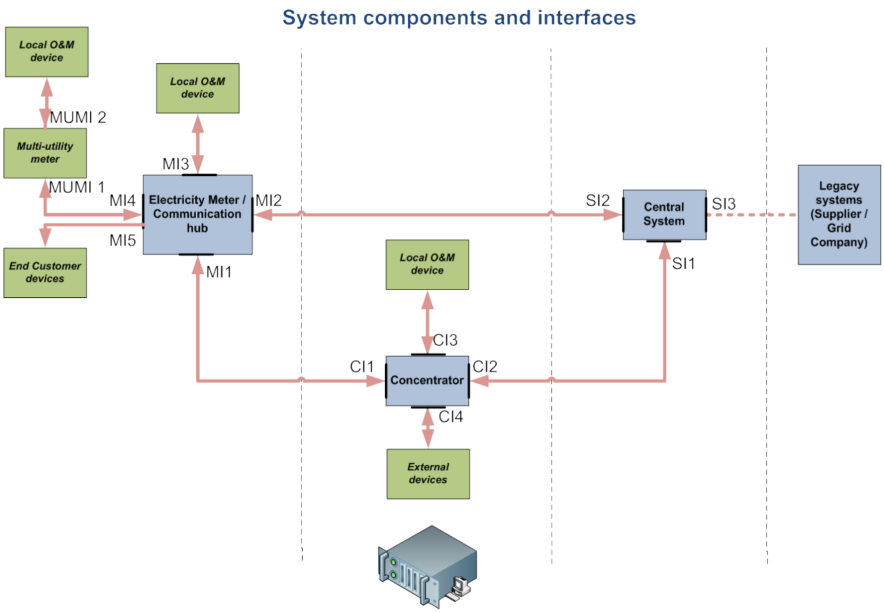


Figur 2.5.: Strukturen i eit eDIANA-system. Ei celle er typisk ei enkel bueining, medan ei makrocelle kan innehalda fleire bueiningar. Henta frå [20].

Kommunikasjon i eDIANA skjer ved hjelp av ulike teknologiar, alt etter kva nivå kommunikasjonen skjer på. Innad i ei celle ser ein for seg å nytta ZigBee, medan kommunikasjon mellom celler er tenkt å nytta Ethernet over straumlinjer. [21]

Noko av det mest interessante med eDIANA-prosjektet, er måten det handterer varierande straumprisar og -laster på. Sidan veldig mange menneske har ein veldig lik døgnrytme, får ein i eit faktisk straumnett ganske kraftige toppar i straumbruken om morgonen og om ettermiddagen. Dette gjer at straumprisane aukar nettopp då.

Ved å bidra til å flytta straumbruken i tid, prøver eDIANA å redusera desse effektene. Dette kan skje ved at ein sørgjer for at visse typar hushaldsartiklar ikkje køyrer akkurat når presset er størst, men kanskje nokon timar tidlegare eller seinare. Nettopp skiljet mellom makroceller og celler skal gjera dette enklare. Ein ser for seg at makrocellenivået, som tek seg av den langsiktige styringa, gir beskjed til cellekontrollaren om at straumen er dyr no, og at laster som kan, bør slåast av eller utsetjast. Dermed kan cellekontrollaren velja å køyra vaskemaskina litt seinare i staden for akkurat no. I nokon tilfelle bør truleg brukaren spørjast om godkjenning, men dette er det i så fall òg cellekontrollaren som eventuelt må ordna. [22]



Figur 2.6.: Gjennom OPEN meter-prosjektet ønskjer ein å etablera standardar for alle kommunikasjonar som er vist i figuren. Det er ikkje eit mål å bruka same standard til alle samband. Figuren er henta frå [24].

2.2.4. OPEN meter

OPEN meter har som mål å utvikla opne standardar for bruk i samband med såkalla «smart metering». Prosjektet starta 30. januar 2009, og skal avsluttast innan 30. juni 2011. På same måte som mange av dei andre prosjekta, er OPEN meter delvis EU-finansiert, gjennom «7th Framework Programme». Av deltakarar kan ein merka seg at ST Microelectronics er aktive òg her.

Prosjektet dreier seg altså om kontinuerleg avlesing av til dømes straum- eller gassforbruk, i tillegg til å gjera det mogleg for forbrukaren å tilpassa sitt forbruk til gjeldande straumpris. Dette liknar litt på eDIANA-prosjektet, men i OPEN meter er det ingen automatikk inne i biletet for å gjera sjølve styringa. I staden er det større fokus på å etablera standardane som trengs i informasjonsutvekslinga. Prosjektet har som mål å nytta eller byggja vidare på eksisterande standardar der det er mogleg. [23]

Primært fokuserer OPEN meter på kommunikasjon mellom sjølve smartmeteret og utstyret hos straumleverandøren. Ein ser for seg at ein brukar

har fleire ulike «smartmeter», typisk for gass, straum og vatn. Av desse ser ein det som enklast å nytta straummålaren som hovudkontaktpunkt hos brukaren, sidan denne har god tilgang på straum.

I dokumenta frå prosjektet er det eit tilsynelatande komplett forslag til XML-skjema for dataoverføring mellom målaren og straumleverandøren. Som kommunikasjonsteknologi ønskjer ein å nytta datatrafikk over mobilnettet, gjennom til dømes GPRS[3] eller UMTS[4]. [24]

I tillegg til kommunikasjon mellom målar og leverandør, har ein i prosjektet òg tankar om at målaren skal kunna kommunisera med sluttbrukaren og eventuelt smarte apparat i heimen. Desse grensesnitta er foreløpig ikkje fullstendig spesifisert, men det ser ut til at prosjektet ønskjer å bruka ZigBee og Blåtann som kommunikasjonsteknologi, litt etter applikasjon. [25]

Som nemnt er målet med OPEN meter-prosjektet å laga opne standardar. Dette skal skje ved å laga såkalla «drafts», som ein overlet til standardiseringsorganisasjonar på vanleg måte. Dermed vil desse koma ut på høyring, og til slutt enda opp som opne, internasjonale standardar som kan nyttast i faktiske produkt. Foreløpig har ingen slike «drafts» blitt lagt fram.

2.2.5. DIEM og SOFIA

DIEM og SOFIA er eigentleg to ulike prosjekt, men det mest interessante resultatet herfrå er det det samarbeider om. SOFIA er delvis finansiert gjennom EU sitt Artemis JU-program, medan DIEM starta som eit prosjekt ved universitetet i Tampere, i Finland. SOFIA blir leia av Nokia, men òg Philips og brikkeprodusenten NXP er deltakarar. [26]

Dei interessante delane av desse prosjekta vart først skissert i DIEM-prosjektet, starta i 2008, men blir vidareført gjennom SOFIA, som først starta i 2009.

DIEM identifiserer ein del problem med eksisterande løysingar, som prosjektet søkjer å løysa:

- Løysingane er ofte domene-spesifikke, og dermed lite tilgjengelege for andre applikasjonar enn dei opphaveleg tenkte.

- Løysingane er typisk lukka slik at berre den opphavlege utviklaren kan byggja vidare på dei. I tillegg må heile løysinga ofte kjøpast og implementerast ved bygging.

[3] General packet radio service, ein metode for dataoverføring i GSM-nett

[4] Universal Mobile Telecommunications System, tredjegenerasjons mobiltelefonnettverk, med god datakapasitet

- Bedrifter som opererer i smarthus-marknaden samarbeider ikkje, og ein klarer ikkje å finna forretningsmodellar som gjer det økonomisk fristande å samarbeida.

- Løysingane er for komplekse, og skalerer til dømes ikkje ned til heilt enkle sensoreiningar eller mobile einingar.

Basert på desse problema, gjer prosjektet nokon observasjonar om korleis ei løysing må vera. Til dømes må ein opna for å dela mest mogleg informasjon. DIEM påstår at jo meir ein klarer å dela og bruka informasjon, jo høgare verdi har informasjonen. For å oppnå dette, må det lagast ei løysing som er enkel og billeg å ta i bruk for produsentar. Dette gjer at løysinga bør vera open, utan lisens- eller betalingskrav knytta til seg. Jo meir informasjon som er tilgjengeleg, jo meir fristande er det òg for produsentar å nytta teknologien.

Vidare argumenterer DIEM for at det er umogleg å på førehand sjå for seg alle moglege bruksområde. Dette gjer at det er viktig at løysinga er generell og skalerbar, og ikkje set store krav til reknekraft eller berre passar til ein type einingar.

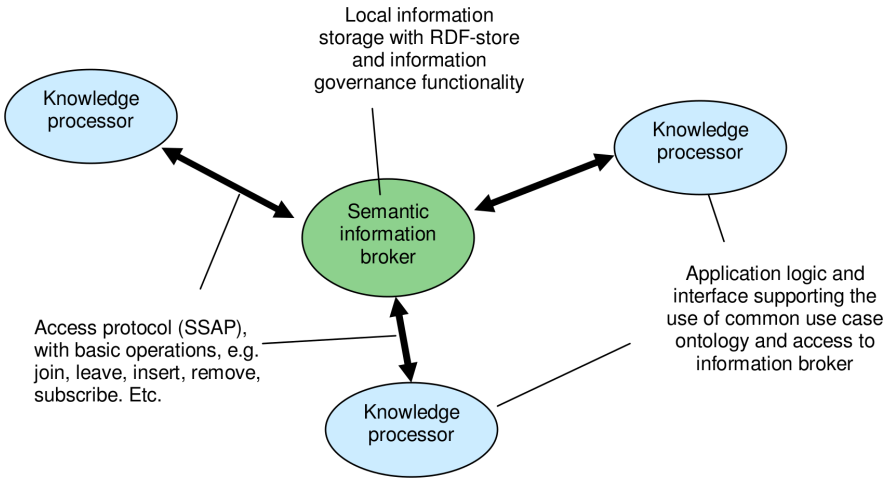
Løysinga er òg avhengig av å bli akseptert av sluttbrukarane. Dette krev truleg at ein kan implementera han bit for bit, og unngå store startinvesteringar. Ho må òg gi faktiske fordelar i bruk, og gjera det mogleg å oppnå ting ein ikkje kan klara utan.

Som eit svar på desse problema, laga DIEM-prosjektet Smart-M3, som dei kallar ei interoperabilitetsplattform. Smart-M3 skal vera open, enkel og utvidbar, og mellom anna difor set ho ingen krav til kommunikasjonsteknologi i einingane som nyttar løysinga. I staden baserer plattforma seg på at kvar eining nyttar sine eksisterande kommunikasjonsprinsipp. Dette gjer at Smart-M3 framstår som ei langt mindre og lettare løysing enn dei liknande løysingane i til dømes AMIGO eller SM4ALL. Dette kjem dermed delvis av at ho på ein måte løyser færre problem, ved å ikkje spesifisera eller setja fleire krav enn det som er absolutt nødvendig.

Smart-M3 er bygd opp hovudsakleg på to konsept, kalla «Semantic information broker» (SIB) og «Knowledge processor» (KP), som vist i figur 2.7. Informasjonsforhandlaren er ansvarleg for å lagra og tilby informasjon til kunnskapsprosessorane. Forhandlaren nyttar eit format kalla RDF[5] for å lagra data.

Ein kunnskapsprosessor er typisk ein programvaremessig del av ei eining, som både kan henta og abonnera på informasjon frå ein SIB, og putta inn eigen informasjon. Ein KP er typisk svært knytta til eininga han køyrer

[5]Resource Description Framework



Figur 2.7.: Strukturen og hovudkomponentar i Smart-M3-systemet. Henta frå [27]

på, og det finst difor ikkje nokon generelle slike, som ein kan nytta i eigne prosjekt.

Typisk finst det éin informasjonsforhandlar og fleire kunnskapsprosessorar i eit system. Desse snakkar saman ved hjelp av ein protokoll kalla «Smart space access protocol» (SSAP), over ein uspesifisert kommunikasjonskanal. SSAP ein ein svært enkel protokoll, med kommandoar for å starta og avslutta kommunikasjon med ein SIB, manipulera data og abonnera på hendingar.

Konsepta og protokollane frå desse prosjekta er meint å vera enkle, og krava til system som nyttar dei er få. Dermed skal det vera lett å bruka desse i eigne system. Hovudtanken er på mange måtar nettopp det å sentralisera informasjonen i systemet, slik at han kan bli brukt på nye måtar, utan at ein nødvendigvis må ha sett og planlagt alle bruksområde på førehand, eller må forhalda seg til at informasjonen er samla frå svært mange ulike einingar.

Det eksisterer ein referanseimplementasjon av ein Smart-M3-informasjonsforhandlar, men denne er ikkje komplett. Til dømes har han ikkje tilgangskontroll. Smart-M3 er tenkt vidareutvikla gjennom både DIEM og SOFIA. [27]

2.2.6. SmartProducts

Det finst òg forskingsprosjekt som fokuserer på ein del smalare felt innan smarthuskonseptet. Eit døme på det er prosjektet SmartProducts, som òg er eit delvis EU-finansiert prosjekt. Prosjektet dreier seg om å gjera produkt smartare, på ein slik måte at dei blir enklare for brukarane å forhalda seg til. I dette prosjektet blir det fokusert på tre område: produksjon av fly, livsløpet til ein bil og smart kjøkken.

I forhold til husstyring, er det smarte kjøkkenet mest interessant. Her snakkar ein om smarte produkt ikkje som produkt som kan styrast på spesielt revolusjonerande måtar, men som eit uttrykk for produkt som samarbeider. På nettsidene til prosjektet, finst det to videoar som døme på kva ein ser føre seg: ein smart kaffimaskin og ein såkalla «CocktailCompanion». [28]

Den smarte kaffimaskinen er eit døme på korleis hushaldsapplikasjonar kan vera meir intelligente i måten dei oppfører seg ovanfor menneske. Maskinen nyttar koppar med RFID-taggar, som er knytta til personar på arbeidsplassen. Det gjer at han automatisk kan laga den «vanlege» kaffien din når du set koppen på plass. Vidare kan han identifisera brukaren, og tiltala han ved namn når maskinen treng til dømes påfyll av vatn eller reinsing. Reinseprosessen tek lang tid, men brukaren kan forlata kaffimaskinen og få ei melding på datamaskinen sin eller telefonen når han må gå tilbake og ordna kaffimaskinen. Vidare gjer identifikasjon det mogleg å tella koppar for ulike brukarar, slik at ein til dømes kan fakturera riktig.

«CocktailCompanion» er eit litt anna type døme, då det i større grad dreier seg om samarbeid mellom ulike einingar. Videoen viser korleis maskinen basert på kva ingrediensar og einingar som er tilgjengelege viser kva drinkar ein kan laga. Når ein vel ein type, blir ein leia gjennom laginga, steg for steg. Maskinen er til dømes kopla til ei vekt du set glaset på, slik at han kan måla når du har nok væske. I tillegg kan han kommunisera med ein kaffimaskin for å laga riktig type kaffi til den drinken ein lager.

Desse to applikasjonane er meint som døme på kva som er mogleg, og fokuset for prosjektet er ein del vidare. Det interessante er mest i forhold til det litt annleis perspektivet og fokuset dette prosjektet har på det som dreier seg om intelligente hus, enn dei andre prosjekta som her er omtala.

2.2.7. SmartHouse Roadmap

SmartHouse Roadmap er i motsetnad til ein del av dei andre omtalte prosjekta ikkje eit prosjekt som har som mål å utvikla noko nytt. I staden dreier prosjektet seg om å sjå på eksisterande standardar, og vurdera kva som kan gjerast for å betra samarbeid og sambruk mellom einingar. Prosjektet

Figur 2.8.: Oppsettet «CocktailCompanion» nyttar for å hjelpa med drinkmiksing. Til venstre i biletet kan ein sjå vekta som blir nytta og skjermen som viser oppskrifta. Til høgre, oppå hylla, står kaffimaskinen systemet nyttar. Skjermbilete frå video på [28].

er starta av CENELEC, som er ei europeisk samanslutning av standardiseringsorganisasjonar som arbeider innan elektrotekniske fagfelt. Ei rekke bedrifter og andre organisasjonar er òg deltakarar, mellom andre ABB, Intel, KNX Association, ZigBee Alliance og det norske Post- og teletilsynet.

SmartHouse Roadmap prøver å løysa problema som blir skapt av at det er så mange aktørar innan smarthusbransjen i Europa, der svært mange nyttar eigne eller i alle fall domenespesifikke standardar. Det skal her merkast at prosjektet nyttar omgrepet smarthus i ein svært vid forstand, og inkluderer alt frå tradisjonelle PC-nettverk til brukargrensesnitt på mobiltelefonar. Dette gjer det på mange måtar urealistisk å sameina alle i ein felles standard.

I staden er målet å sjå på korleis ein kan gjera dei eksisterande standardane samarbeidande, og gjera det mogleg å bruka einingar på tvers av ulike standardar. Dette blir bygd ut frå konseptet om ulike økosystem. Prosjektet kallar ei gruppe produkt eller einingar som samarbeider om å løysa ei oppgåve for eit økosystem, til dømes Blåtann, USB, Wi-fi og liknande. Dermed kan ein definera eit smarthus som ein stad der ulike økosystem kan samarbeida og dela informasjon. [29]

Prosjektet skisserer ei rekke såkalla brukstilfelle, «use cases», som ikkje er moglege i dag, men som burde vore det. Eit døme er å visa bilete direkte

frå ein mobiltelefon på fjernsynsskjermen. I mange tilfelle er infrastrukturen som skal til på plass, med både fjernsyn og mobiltelefon tilkopla same trådlause nettverk. Likevel er det ikkje mogleg i praksis, hovudsakleg fordi einingane ikkje snakkar same «språk».

SmartHouse Roadmap meiner at insentiva for dei enkelte aktørane i denne situasjonen ikkje er sterke nok til at dei er villige til å investera nødvendig tid og utvikling. Dette dreier seg både om at fordelane ikkje framstår som store nok, og at kompleksiteten kanskje ser større ut enn han faktisk er. Eit mål med prosjektet er difor å setja opp forslag til korleis slik interoperabilitet kan nåast, med føreslåtte standardar for kommunikasjon og samarbeid.

I tillegg har prosjektet som mål å analysera både gjeldande og framtidige forbrukarbehov og -ønske, og kva som må til for å tilfredsstilla desse. [30]

Foreløpig er resultata frå dette prosjektet noko uklare, og prosjektet har ikkje ei eiga nettside. Det blir snakka om verktøy som er utvikla for å analysera kva krav ein får til interoperabilitet, og kva standardar som bør implementerast for å tilfredsstilla eit gitt brukstilfelle. Om desse er tilgjengelege for andre enn prosjektdeltakarar er likevel ikkje kjent. Dersom ein skal oppsummera arbeidet i SmartHouse Roadmap i ein setning, er det truleg at dette er eit område det må arbeidast meir med.

3. Krav til systemet

Basert på dei systema som tidlegare er omtalte, er det naturleg å gå vidare med å setja opp krava til eit nytt system. Ein utviklingsmetodikk er eit godt grunnlag for det vidare arbeidet med kravspesifikasjonen.

Som eit utgangspunkt for sjølve kravspesifiseringa blir det skissert eit bruksscenario. I dette blir det referert til ulike krav i den endelege kravspesifikasjonen, som blir sett opp etterpå.

Systemet som blir laga, er basert på ZigBee. I tidlegare arbeid har det blitt gjort ei vurdering av moglege kommunikasjonskanalar å bygga eit system som det her er snakk om på, og ZigBee vart då funne å vera det beste alternativet. [2]

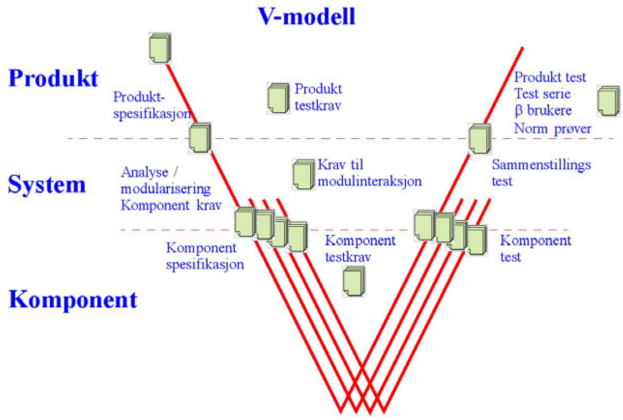
3.1. Utviklingsmetodikk

Det finst svært mange forskjellige metodar for korleis ein skal gå fram i arbeidet med å utvikla eit system som det her er snakk om. Ein utviklingsmetodikk vil i denne samanhengen seia noko om kva rekkefølge ting skal skje i, kva ein skal fokusera på og liknande. Det blir ikkje gått inn på kva ulike metodikkar som finst, eller teoriane bak, men ein har valt å basera arbeidet på den såkalla V-modellen.

V-modellen har truleg namnet sitt frå forma som ofte blir nytta for å skildra han, vist i figur 3.1. Den grunnleggande flyten i utviklinga er å starta øverst til venstre i V-en og arbeida seg nedover. Ein startar dermed med eit slags konsept ein ønskjer å laga, arbeider vidare med meir nøyaktig spesifikasjon og design, før ein implementerer på botnen.

På veg opp igjen frå botnen av V-en, er målet å verifisera kvart steg ein var innom på veg ned. Dette kan til dømes innebera å testa den ferdige implementasjonen opp mot spesifikasjonen som er utvikla. Til slutt må ein vurdera om resultatet ein sit igjen med, faktisk kan oppfylla eller implementera konseptet ein starta med.

Denne modellen er særleg fordelaktig for system av typen det her er snakk om, då det i utgangspunktet kanskje er noko uklart akkurat kva ein vil arbeida fram, og korleis det bør gjerast. Ved å arbeida med konsept, design og spesifikasjon, får ein eit godt grunnlag for å gjera den faktiske



Figur 3.1.: Forenkla V-modell. Henta frå [31].

implementasjonen, og deretter til å vurdera om ein har oppnådd det ein ønskte.

Det var nødvendig å gjera nokre modifikasjonar av flyten, for å koma innanfor tidsrammene. Det viste seg relativt tidleg at ein var nødt til å bruka ein ekstern leveradør til å produsera eitt av kretskorta i systemet. Dette tek naturleg nok ekstra tid, og det var difor viktig å få ferdig denne designen tidleg, slik at ein slapp å venta på dette kortet for å gå vidare i arbeidet.

Ein valde difor å fokusera på å designa denne delen tidleg, og senda bestillinga, før resten av systemet var ferdig designa.

3.2. Systemoppsett

3.2.1. Bruksscenario

Tor bur i eit hus med eit system som det her er snakk om. Systemet hans består av ei rekke trådlause brytarar og andre inngangar, som har ein bestemt funksjon, men som kan flyttast fritt rundt. (S 4) Desse kontrollerer ulike utgangar i systemet, og kan til dømes slå av og på lys. Ved senga har han ein «god natt»-brytar som slår av alt som skal slåast av om natta, og ved utgangsdøra har han ein brytar som fortel systemet at han går ut, og slår av det som skal slåast av då. (I 2)

Kvar av desse brytarane set systemet i ein tilhøyrande modus, i tillegg til å slå av ting. Til dømes set brytaren ved senga huset i nattmodus, slik

at alle lys blir dimma ned. Styringa fungerer òg som vekkarklokke, slik at systemet veit når Tor ønskjer vanleg funksjonalitet igjen. (I 4)

Ved utgangsdøra har Tor eit panel, der han fortel systemet når han reknar med å vera tilbake. Basert på dette, kan systemet vita om det er økonomisk å slå ned varmen, og kva tid han eventuelt bør slåast på igjen.

Dei enkelte utgangselementa er skjulte inni veggane, og er difor ikkje synlege. Dei utgangselementa som er knytta til straumbrukande einingar, måler kontinuerleg kor mykje straum den tilkopla eininga treng. Dette gjer at Tor kan få melding om noko trekk meir straum enn det vanlegvis gjer, og at han kan få råd om kor mesteparten av straumen blir brukt. (R 4) Slike målingar gjer det lettare å sjå kor ein kan redusera straumforbruket.

Brytarane er batteribaserte, og kan flyttast fritt rundt i huset. Dei held same funksjon, uavhengig av plassering. (I 5)

Når Tor trykkjer på ein brytar, er det ikkje alltid det som skal skje, kan skje umiddelbart. For å indikera at systemet likevel har registrert trykket, er alle einingar som har ei eller anna form for brukargrensesnitt utstyrt med ein lysdiode. Denne blinkar når brytaren blir trykkja på, for å visa at trykket er registrert, og at handlinga vil bli utført. (I 1)

Systemet har ein såkalla heimesentral (S 5), med eit web-basert brukargrensesnitt. (H 2) Ved hjelp av dette kan Tor styra alle funksjonar i huset frå einkvar nettlesar, til dømes frå ein smarttelefon eller datamaskin. (H 4) I tillegg til dei fysiske brytarane, er dette systemet sitt primære grensesnitt.

Sentralen gjer det òg mogleg å styra huset frå andre stader, utanfor huset, gjennom det same brukargrensesnittet heimesentralen tilbyr internt. Det finst to måtar å ordna dette på: Anten kan han sjølv konfigurera brannmuren sin og opna for tilgang til heimesentralen frå internett, eller han kan abonnera på ei løysing som overfører kommandoar frå ei anna netteneste til heimesentralen. Med sistnemnde løysing treng Tor berre å leggja brukarnamn og passord til den eksterne tenesta inn i heimesentralen sin, og ikkje noko meir konfigurasjon. (H 5)

Alle brytarar og lokale styringseiningar fungerer utan heimesentralen. (S 6) Den einaste funksjonaliteten som er avhengig av sentralen, er styring og konfigurasjon ved hjelp av web-grensesnittet, både lokalt og eksternt.

Naboen til Tor har same system, men på grunn av at systemet nyttar kryptering med unike nøklar, er det verken mogleg for Tor å sjå kva naboen gjer, eller å styra naboen sine einingar. (S 9)

Når ei ny, ukonfigurert eining blir slått på, søker ho etter tilgjengelege nettverk å kopla til. Ho vil velja å kopla til det sterkaste, og vil då dukka opp i ei liste over ukonfigurerte einingar i web-grensesnittet. Frå denne lista kan Tor velja kva funksjon han vil at eininga skal ha. (H 3)

Fordi ei eining alltid koplar seg til det sterkaste nettverket, kan ho i nokon tilfelle kopla seg til feil nettverk. Ved å trykka på ein knapp på

eininga er det då mogleg å få eininga til å finna eit nytt nettverk. (S 7)

Ein del funksjonalitet er innebygd i systemet, men ein leverandør av einingar kan òg laga ei eining som skal fungera saman med eit lite program. Dette gjer det mogleg å laga meir avansert funksjonalitet. Når slike einingar blir starta, hentar heimesentralen dette programmet frå leverandøren si nettside, utan at Tor treng å gjera noko. Eininga er dermed klar for konfigurasjon når ho dukkar opp i web-grensesnittet. (S 10)

Systemet er heller ikkje bunde til å berre styra lys og varme. Nettverket i seg sjølv set ingen avgrensingar på kva det kan brukast til.

3.2.2. Struktur

Dersom ein ser på dei eksisterande systema nemnde i underkapittel 2.1, kan ein skilja mellom dei sentraliserte systema Control4 og Nobø og det heilt distribuerte Nexa-systemet. Begge desse strukturane har fordelar: Den sentraliserte strukturen gjer det mogleg å gjera avansert, ekstern styring, medan den distribuerte varianten gjer systemet robust mot feil i enkeltkomponentar.

I bruksscenarioet er det tenkt eit system som Nexa-systemet, men med ein sentral i tillegg. Dermed kan all informasjon om kva som styrer kva ligga i dei enkelte einingane, samstundes som sentalen kan gi utvida styringsmoglegheiter. Ved å gjera informasjonsdelinga slik, kan grunnfunksjonaliteten i systemet oppretthaldast sjølv om sentralen skulle vera utilgjengeleg.

I tillegg bør det merkast at både Nobø og System Nexa er spesifikt retta inn mot styring av varme og eventuelt lys. Control4 er i så måte friare, då dette nettverket er ope for å bli nytta til andre bruksområde enn akkurat heimestyring. Å vera ope på den måten, er eit viktig krav til systemet, då ein vanskeleg kan sjå føre seg alle moglege bruksområde på ein gong.

3.2.3. Fleksibilitet

I scenarioet er det skissert trådlause brytarar. Dette er ein svært viktig del av systemet, då det gir enorm fleksibilitet i forhold til tradisjonelle, kabelbundne brytarar.

For det første gir trådlausheiten moglegheit til å plassera brytarane fritt, utan å trekka nye kablar. Dersom ein frå starten av kan designa eit elektrisk anlegg for bruk med slike brytarar, kan dette òg redusera kostnaden betydeleg, då ein slepp å leggja kablar og boksar til brytarar. I eksisterande anlegg er den største fordelen at ein enkelt kan setja opp ein ny brytar, utan å gjera endringar på eksisterande anlegg.

For det andre gir slike brytarar, om dei som skissert i scenarioet kan rekonfigurerast, store moglegheiter til å ombestemma seg i forhold til kva ein vil styra med ein gitt brytar. Dette gjer det mogleg å enkelt slå av ei ny lampe med same brytaren som allereie slår av dei eksisterande lysa i eit rom.

3.2.4. Brukargrensesnitt for heimesentralen

Som brukargrensesnitt for heimesentralen er det ønskeleg å oppnå størst mogleg fleksibilitet, med minst mogleg arbeid. I bruksscenarioet er det difor skissert å nytta eit web-grensesnitt på heimesentralen til både styring og konfigurasjon.

Å nytta eit web-grensesnitt gjer at ein utan å gjera noko ekstra, kan nytta ei lang rekke ulike einingar til å styra systemet, i staden for å måtta tilpassa grensesnittet til kvar enkelt eining. Dette gir langt høgare fleksibilitet enn dagens system, som i den grad dei har andre styringsorgan enn brytarar på veggen, baserer seg på spesialtilpassa applikasjonar.

Tilpassing av grensesnittet til spesifikke einingar er sjølvsagt likevel mogleg. Med svært varierande storleikar på skjermar, kan det til dømes vera naturleg å laga spesielle variantar for mobiltelefonar eller liknande. Slik tilpassing vil likevel primært dreia seg om presentasjon. Dei bakanforliggjande teknologiane og prinsippa er dei same.

3.2.5. Styring

Ved å basera brukargrensesnittet for heimesentralen på web-teknologi, er det som nemnt enkelt å gjera lokal styring av systemet frå ei lang rekke einingar. I tillegg blir det mogleg å opna for ekstern styring relativt enkelt.

Her kan ein, som nemnt i scenarioet, sjå for seg minst to moglegheiter: Å opna grensesnittet heimesentralen tilbyr ut mot verda, eller å nytta ei ekstern teneste til å vidaresenda alle kommandoar. Det første alternativet vil kreva noko konfigurasjon av brannmuren hos brukaren, medan den andre løysinga bør kunna gjennomførast svært enkelt.

Ei ekstern teneste kan òg gi betre sikkerheit, fordi heimesentralen då ikkje treng å vera tilgjengeleg frå internett, og dermed ikkje er utsett for eventuelle ondsinna angrep. Ei slik teneste kan òg gi grunnlag for å tena noko pengar for eit sentralt firma, gjennom ei abonnementsordning.

3.2.6. Konfigurasjon

Korleis systemet skal konfigurerast er òg svært viktig å tenka på. Dersom ein ser på eksisterande system, kan ein sjå at til dømes Nexa si løysing her

er todelt: Først set ein utgangen ein vil konfigurera i konfigurasjonsmodus, og deretter aktiverer ein den inngangen ein vil bruka for å styra utgangen. Nobø fungerer på ein svært liknande måte.

Eit problem med denne metoden, er at han krev at ein har fysisk tilgang på både inngangs- og utgangseining. Dette kan vera ein vanskeleg føresetnad, då den eine eller begge einingane kan vera inni kontaktar i veggen eller liknande.

Control4 løyser dette systemet svært annleis. I eit slikt system har brukaren i utgangspunktet ingen konfigurasjonsmoglegheiter, men er i utgangspunktet prisgjeven installatøren sin. Dersom ein kjøper ekstra programvare, ser det likevel ut til at i alle fall visse aspekt ved systemet kan endrast av sluttbrukaren.

For at systemet skal kunna tilpassast ulike situasjonar og i større grad gjera det mogleg å oppfylla brukaren sine ønske, bør brukaren sjølv kunna endra konfigurasjonen. Som skissert i scenarioet ovanfor, ser ein difor for seg at konfigurasjon skjer gjennom eit web-grensesnitt på heimesentralen.

3.2.7. Andre krav

I tillegg til dei funksjonelle krava som kjem fram gjennom bruksscenarioet, er det òg viktig at systemet tilfredsstiller krav til tryggleik for brukaren, elektrisk støy, straumforbruk og storleik.

Tryggleik for brukaren dreier seg først og fremst om å ha ei form for innkapsling, slik at det ikkje er mogleg å koma i kontakt med straumførande delar. I tillegg må ein passa på at einingar til dømes ikkje blir så varme at dei kan antenna ein brann.

Elektrisk støy er eit aspekt som gir utslag i designfasen. Ein må sørga for at dei enkelte einingane i systemet ikkje støyar meir enn det som er akseptabelt. Her får ein noko hjelp gjennom å nytta ein eksisterande kommunikasjonsstandard, då ZigBee allerede set visse grenser for korleis ei nettverkseining kan oppføra seg i forhold til radiosambandet.

Straumforbruk er eit problem med fleire sider. Enkelte einingar vil vera batteridrivne, og for desse er det eit mål med lengst mogleg levetid på kvart sett batteri. For andre einingar har ein tilgang på straum frå veggen, og levetid er dermed ikkje eit problem. Likevel gjer auka straumforbruk at ein må nytta større og dyrare komponentar i straumforsyninga, og det er ønskeleg å unngå dette. Dessutan kan einingane sitta inni veggen, eller andre stader der høg effektforbruk vil kunna vera problematisk i forhold til varmeutvikling. Som tidlegare arbeid viser, er straumforbruket for sjølve ZigBee-kommunikasjonen relativt lågt, men tap i straumforsyninga kan likevel gi for høgt forbruk.

Fysisk storleik er òg eit krav ein må ta omsyn til. For nokon einingar er det ønskeleg å ha einina inni veggboksen ein set kontakten i, og dette set ganske strenge krav til storleiken på einingane. I tillegg bør heller ikkje brytarane som skal henga på veggen gjera meir av seg enn tradisjonelle brytarar.

3.2.8. Avgrensingar

Fordi ei masteroppgåve på sivilingeniørstudiet ved NTNU berre er eit halvt års arbeid, er det ikkje realistisk å implementera alle delar av systemet slik det er skissert i bruksscenarioet i denne omgang. Scenarioet inneheld mange forskjellige nodetypar, ein del av dei inkludert problem som i seg sjølv er store nok for ei masteroppgåve, til dømes det med å dimma ulike lyskjelder.

Det er difor eit mål å implementera berre det heilt grunnleggande som skal til for å visa at systemet er mogleg å byggja, og dermed laga ei plattform ein seinare kan byggja vidare på. Dette er òg tenkt på i oppgåveteksta.

For å i det heile kunna ha nokon funksjonalitet, er ein i utgangspunktet nøydd til å ha minst ein inngang og ein utgang. I tillegg er det nødvendig å laga heimesentralen, sidan konfigurasjon og meir avansert styring skjer frå denne.

For å gi størst mogleg fleksibilitet med minst mogleg arbeid, er det ønskeleg å ha ein inngang som har fleire «kanalar». Ein vel difor å implementera ein lysbrytar med fire individuelle knappar, som må kunna styra ulike einingar.

Som utgang er et ønskeleg å kunna styra alle vanlege elektriske apparat. I vanlege hus, kan eit vanleg apparat trekka opp til 16 ampere ved 230 volt. Det er difor naturleg å prøva å laga ein utgang som kan slå av og på nettopp eit slikt apparat. Ein skulle gjerne hatt dimming òg, men dette er som sagt eit problem som per i dag er ganske komplekst, på grunn av ein stor mengde ulike lyskjelder på marknaden. Dette blir difor ikkje prioritert.

Sjølvsagt er det òg mange andre nodetypar som ville vore interessante å laga, men som det ikkje er tid til å gjera i denne omgang. I kapittel 8 blir nokre slike nodar føreslått. Noko av målet med oppgåva er òg å gjera slik vidareutvikling så lett som mogleg.

3.3. Kravlister

Basert på bruksscenarioet, saman med avgrensingane og utdjupingane ovanfor, kan ein setja opp spesifikke krav som bør oppfyllast av systemet. Det

er naturleg å dela desse i krav som gjeld heile systemet, krav som gjeld inngangsnodar og krav som gjeld utgangsnodar. Desse er viste i tabell 3.1

Vidare kan ein for kvar av dei tre nodetypane som er definerte i delkapittel 3.2.8 setja opp krav, som berre gjeld desse spesifikke typane. Desse er lista opp i tabell 3.2, 3.3 og 3.4.

I tabellane med kravlister blir det snakka om inngangar og utgangar, der ein brytar er eit eksempel på ein inngang, og eit relé er eit eksempel på ein utgang.

Id	Forklaring
S 1	Systemet må kunna fungera som ei plattform for mange ulike bruksområde.
S 2	Alle delar og einingar i systemet må vera trygge å bruka.
S 3	Systemet må ikkje bruka meir straum enn det som er nødvendig.
S 4	Systemet skal støtta ulike typar inngangar og utgangar.
S 5	Systemet skal ha ein heimesentral, for å kunna gjera sentralisert, avansert styring, både internt og eksternt.
S 6	Lokal bruk skal fungera utan heimesentralen.
S 7	Alle einingar i systemet skal ha ein knapp, for å kunna ta i mot brukarhandlingar.
S 8	Alle einingar i systemet bør ha ei rekkevidd på minst ti meter innandørs.
S 9	All kommunikasjon i systemet bør skje kryptert.
S 10	Systemet bør vera utvidbart på ein slik måte at ein tredjeparts-produsent av einingar kan tilby tilhøyrande programvare til brukarane, utan at brukaren gjer noko spesielt.

(a) Krav til systemet som heilskap

Id	Tekst
I 1	Alle brukarhandlingar skal gi respons i løpet av eit halvt sekund. Respons inkluderer alle former for respons til brukaren om at systemet har motteke kommandoen, til dømes å blinka med ein lysdiode.
I 2	Ein inngang skal kunna styra fleire utgangar uavhengig av kvarandre.
I 3	Ein inngang skal kunna styra fleire einingar saman, som ei gruppe.
I 4	Ein inngang i systemet bør kunna ha ulike funksjonar basert på definerte parametrar, til dømes tid på døgnet.
I 5	Einingar som ikkje har naturleg tilgang på straum, skal vera batteridrivne og trådlause.

(b) Krav til inngangseiningar

Id	Tekst
U 1	Ein utgang må kunna styrast av fleire inngangar.

(c) Krav til utgangseiningar

Tabell 3.1.: Generelle krav til dei ulike delane av systemet.

Id	Tekst
H 1	Heimesentralen skal kunna kommunisera både med internett og med dei ulike einingane i husstyringssystemet.
H 2	Brukargrensesnittet for heimesentralen skal vera eit web-grensesnitt.
H 3	All konfigurasjon av systemet skal skje gjennom webgrensesnittet.
H 4	Web-grensesnittet bør vera laga slik at det fungerer frå alle plattformer: telefonar, PC-ar, nettbrett og liknande.
H 5	Heimesentralen bør kunna knyttast opp mot ei ekstern teneste for å gi fjerntilgang utan å konfigurera brannmurar hos brukaren.

Tabell 3.2.: Spesifikke krav til heimesentralen.

Id	Tekst
L 1	Lysbrytaren skal vera om lag like stor som tradisjonelle brytarar.
L 2	Lysbrytaren skal ha fire uavhengige knappar.
L 3	Lysbrytaren skal vera batteridriven, med batterilevetid på over eitt år.
L 4	Lysbrytaren må vita si eiga batterispenning, slik at brukaren kan bli informert når det er nødvendig å skifta dei.

Tabell 3.3.: Spesifikke krav til lysbrytarnoden.

Id	Tekst
R 1	Relénoden bør ikkje vera større enn at han kan monterast inni ein elektrisk boks saman med ein stikkontakt.
R 2	Relénoden skal ikkje vera batteridriven, men ta straum frå nettet.
R 3	Relénoden skal kunna slå av og på 230 V med 16 A.
R 4	Relénoden bør kunna måla kor mykje straum tilkopla utstyr trekk.

Tabell 3.4.: Spesifikke krav til relénoden.

4. Design

I det følgande kapitlet blir dei ulike delane av systemet designa, basert på dei oppsette krava. Verktøy som vart nytta i denne prosessen blir òg gjennomgått.

Det prismessige aspektet er viktig for dette systemet, og for kvart kort som blir designa blir det òg sett på komponentprisar for dette kortet. Ein har valt å ikkje inkludera kretskortkostnadar i prisrekninga. Grunenn til dette er at prisskilnaden mellom prototypar og masseproduksjon er så stor, at det er vanskeleg å vurdera kva den reelle kostnaden eventuelt ville vore.

4.1. Designverktøy

Eit system som blir laga i denne oppgåva, kan ha potensiale for å bli vidareutvikla både kommersielt og som hobby. På grunn av dette er det valt å nytta designverktøy som er fritt tilgjengelege, utan tilknytta lisenskostnadar eller andre bruksavgrensingar, som at programmet til dømes berre kan nyttast for ikkje-kommersielle prosjekt. Dette valet gjer det mogleg å vidareutvikla systemet utan å betala dyre lisensar, uavhengig av om systemet blir kommersialisert.

4.1.1. gEDA-pakka

gEDA[1]-pakka er ei samling program for skjemateikning og kretskortutlegg. Pakka består hovudskaleg av skjemaprogrammet gSchem og utleggsprogrammet PCB. Ho er utvikla over lang tid, og har tilsynelatande relativt stor brukarmasse. Dette gjer at det er mykje hjelp å få på internett. Pakka er GPL-lisensiert, ein fri-programvare-lisens, som ikkje har bruksavgrensingar, og tilfredsstiller dermed det ovannemnde kravet.

I gEDA-pakka er alle programma i utgangspunktet sjølvstendige program. Dette gjer dei noko tyngre å koma i gang med, då dei til dømes nyttar ulike tastatursnarvegar for tilsvarande handling. I tillegg nyttar dei ulike filformat. Det finst heller ikkje særleg store symbolbibliotek, og ein

[1] GPL Electronic Design Automation

måtte difor laga ein del symbol og fotavtrykk før ein kunne starta å designa sjølve korta.

Nokon av desse samhandlingsproblema mellom programma blir redusert av at det finst verktøy som kan utføra nokon typar oppgåver. Ein god mappestruktur, er òg med på å redusera dei. Meir informasjon om dette, samt prosjektoppsett kan finnast i vedlegg B.

4.1.2. PlantUML

For å teikna UML-diagramma som er vist i underkapittel 4.3, vart eit program kalla PlantUML nytta. [32] Programmet tek inn ei tekstfil i eit spesielt format, og gir ut eit diagram i SVG[2]-format, som dermed kan skalerast fritt. Dette kan så konverterast til andre format, som vart inkludert i sjølve oppgåva.

PlantUML er som gEDA-pakka GPL-lisensiert.

4.2. Maskinvare

Som nemnt i underkapittel 3.2.8 valde ein i denne oppgåva å laga ein lysbrytarnode, ein relénode og ein heimesentralnode. Dersom ein ser etter fellestrekk for desse nodane, blir det raskt tydeleg at alle treng å kunna kommunisera over ZigBee. I prosjektoppgåva som vart gjort som ei førebuing til denne oppgåva, vart det nytta ZigBee-løysingar frå Atmel. [2] For å utnytta kunnskapen som vart opparbeida då, er det ønskeleg å gjera det òg i denne omgang.

Dokumentasjon frå Atmel tydar på at det er nødvendig å nytta firelags kretskort for å oppnå stabil ZigBee-kommunikasjon, og dette gjorde prosessen litt meir tungvinn. [33] Institutt for teknisk kybernetikk har berre utstyr for å laga tolags kretskort, og det var difor nødvendig å få kretskort for ZigBee-kommunikasjon produsert eksternt.

Firelags kretskort er ikkje nødvendig for resten av funksjonaliteten i nodane. Då det var raskt og enkelt å laga tolags kort med instituttet sitt utstyr, valde ein difor å skilja design i ein ZigBee-del og tre ulike funksjonsdelar, ein for kvar nodetype. Designen av kvar av desse blir gjennomgått i det følgande. Fordi ZigBee-delen skulle produserast eksternt, og dermed ville få noko leveringstid, starta ein med denne.

Å skilja ZigBee-funksjonaliteten ut i sitt eige kort, gjer systemet langt vanskelegare å masseprodusera. Dette går dermed på tvers av oppgåveteksta. Likevel er det store fordelar i forhold til systemet som ei plattform,

[2]Scalable Vector Graphics - eit XML-basert vektorgrafikkformat

då eit separat kort er langt enklare å integrera i andre prosjekt. I tillegg gjer det kostnaden med prosjektet lågare, samt feilsøking i denne omgang enklare. ZigBee-kortet blir òg tett opptil referansedesignen, noko som reduserer sjansen for å gjera feil på denne delen. Dette er ekstra gunstig, sidan ZigBee-korta må produserast eksternt. Til saman gjer dette at valet likevel kan forsvarast.

Alle kort vart designa med overflatemonterte komponentar, der det var mogleg.

4.2.1. ZigBee-kort

Under arbeidet med prosjektoppgåva vart det klart at Atmel har ei brikke som er kombinert mikrokontrollar og RF-brikke, kalla Atmega128RFA1. [34] Denne er rimelegare enn å nytta separate brikker, som det var gjort i dei ferdige nodane prosjektoppgåva var basert på. Ei enkel brikke gir òg eit enklare kretskortdesign, og det vart difor valt å nytta denne brikka her.

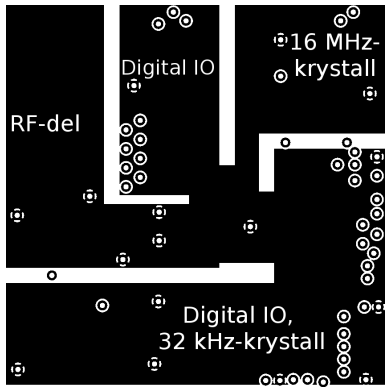
I databladet for Atmega128RFA1 er det skissert ein enkel applikasjonskrets, med ei liste over foreslåtte komponentar. [34] På grunn av ingen tidlegare erfaringar med design av RF-kommuniserande kretskort, vart designen her lagt svært nært opp til denne.

Brikka det her er snakk om er relativt ny, og det eksisterer lite dokumentasjon utanom sjølve databladet frå Atmel si side. For å få eit betre innblikk i utfordringane rundt design av RF-kretsar, vart det søkt etter dokumentasjon på tidlegare ZigBee-kretsar frå Atmel, til dømes brikka som er nytta i nodane frå prosjektoppgåva. [33] Denne dokumentasjonen gjorde det klart at ein burde ha eit eige jordplan i slike kretsar, og at ein dermed måtte ha firelags kretskort. Dette er praktisk talt berre mogleg å gjera på ein skikkeleg måte med firelags kretskort.

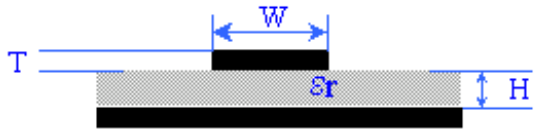
Dokumentasjonen på tidlegare komponentar anbefaler òg å laga jordplanet som ei slags stjerne, med separate armar for dei ulike delane av kortet, RF-delen, digitale inn- og utgangar, krystallar og liknande. Armane skal berre vera samla i jordplata under mikrokontrollaren. Dette vart difor gjort, og det resulterande jordplanet er vist i figur 4.1

Eit anna aspekt som er svært viktig i samband med RF-design, er impedansen i antenne-linjene. Ut av Atmega128RFA1 får eit signal balansert mellom to linjer, som skal gå gjennom ein såkalla balun til antenna. Balunen gjer eit signal som er balansert rundt jord om til eit ubalansert signal, som kan sendast gjennom ei antenne. [35] Impedansen i dei ubalanserte linjene skal vera 100 ohm, medan impedansen i linja til antenna skal vera 50 ohm, i følge databladet for mikrokontrollaren.

I IPC 2251, ein standard som gir råd om korleis høgfrekvens-kretsar bør designast, er det gitt ein formel for å rekna ut denne impedansen. Standar-



Figur 4.1.: Jordplan på ZigBee-kortet, delt opp etter komponentområde. Planet er vist i omtrent dobbel storleik.



Figur 4.2.: Tversnitt av ein kretskortbane. Henta frå [36].

den i seg sjølv er ikkje fritt tilgjengeleg, men det finst fleire kalkulatorar på nett for å rekna med denne. [36]

Impedansen til ein kretskortbane er der gitt som

$$Z_0 = \frac{87}{\sqrt{\varepsilon_r + 1.41}} \ln \frac{5.98H}{0.8W + T} \tag{4.1}$$

der W, H og T er definert som i figur 4.2, og ε_r er den elektriske permittiviteten til kretskortmaterialet.

Både permittiviteten, kor tjukke banane er og høgda til referanselaget er gjeve av kretskortproduksjonen. Dermed er det i utgangspunktet berre breidda på kretskortbanane ein kan justera for å få riktig impedans, i tillegg til å eventuelt nytta kondensatorar for fintilpassing. På kortet vart banebreidda forsøkt tilpassa, i tillegg til at det vart laga plass for ein kondensator for å betra tilpassinga ytterlegare.

Nærast tilfeldig kom ein òg over ein forumpost som fortalte at det var nødvendig å trekka ein spesifikk pinne, TST, på mikrokontrollaren til jord,

for å kunna nytta JTAG[3]-grensesnittet til å programmera kontrollaren. [37] Dette gjorde truleg at ein unngjekk problem seinare, då programmereren som er nytta i oppgåva, nytta nettopp JTAG.

Det var eit mål å laga ZigBee-kortet så fleksibelt som mogleg. Ein valde difor å laga kortet slik at dei fleste inn- og utgangane på mikrokontrollaren er tilgjengelege, særleg alle typar kommunikasjonsbussar og analog-til-digital-konvertering. For å kopla ZigBee-kortet saman med andre kort, valde ein difor å nytta to samankoplingskontaktar på baksida av kortet, med 16 pinnar i kvar. Dette var nok til å gi tilgang til dei fleste pinnane på mikrokontrollaren, når ein ser vekk frå pinnar som berre er nytta til referansespenningar eller ZigBee-kommunikasjon.

Fordi dei ulike nodetypane i systemet kjem til å nytta ulike straumkjelder, valde ein å ikkje inkludera verken spenningsregulatorar eller batterihaldarar på ZigBee-kortet.

Utover det som skal til for ZigBee-kommunikasjon, samt samankoplingskontaktane, er ein glattekondensator, ei pull-up-kopling for RESET-pinnen og motstandar for å dra utgangar til jord dei einaste komponentane på ZigBee-korta. RESET-signalet er aktivt-lågt, og pull-up-koplinga er laga sånn at ein får ei viss forseinking frå straumen blir slått på til RESET-pinnen når eit nivå slik at mikrokontrollaren startar. Dette er vanleg å nytta for å unngå problem ved eventuell ustabil spenning under oppstart.

Det er litt delte meiningar om ein må trekka ubrukte pinnar på moderne mikrokontrollarar til jord eller ikkje. Truleg kunne ein klart seg utan, men dei vart inkludert her primært for å kunna gi tilgang til endå fleire inn- og utgangspinnnar, om det skulle vera nødvendig.

Ein prøvde å gjera modulen så liten som mogleg, og det resulterte i at det endelege kortet berre vart 2,5 cm·2,5 cm. Dei fleste komponentane er på oversida av kortet, medan tilkoplingskontaktane, pull-up-koplinga og glattekondensatoren er plassert på baksida.

Prisen for ZigBee-kortet påverkar i stor grad prisen på systemet som heilskap, då alle nodar treng ZigBee-kommunikasjon. I tabell 4.1 er prisane for dei enkelte komponentane kortet nyttar vist.

Den endelege skjema- og kortdesignen er vist i vedlegg D.

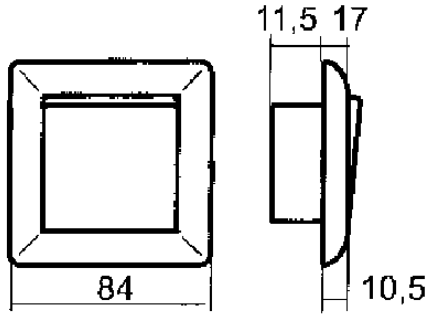
4.2.2. Lysbrytarnode

Oppgåva til brytarnoden er å vera ein inngang til systemet. I tabell 3.3 er det sett opp ei rekke krav til denne nodetypen. Til dømes skal brytaren vera av same storleik som ein tradisjonell lysbrytar (L 1). Svært mange

[3]Joint Test Action Group - programmering- og avlusingsgrensesnitt for mikrokontrollarar og liknande.

Komponent	Type	Tal	1	1000	Sum	Prosent
Mikrokontrollar	Atmega128RFA1	1	9,46	6,31	6,31	41,70 %
Antenne	-	1	1,07	0,64126	0,64	4,24 %
Balun	-	1	0,854	0,45	0,45	2,97 %
Kontaktar	Hokjønn	2	6,16	2,926	5,85	38,67 %
Krystall	32 kHz	1	1,50	0,90	0,90	5,95 %
	16 MHz	1	1,50	0,76	0,76	5,02 %
Kondensator	4 pF	2	0,09	0,00748	0,01	0,10%
	12 pF	2	0,061	0,0143	0,03	0,19 %
	22 pF	2	0,072	0,0169	0,03	0,22 %
	100 nF	1	0,011	0,0026	0,00	0,02 %
	1 µF	4	0,077	0,0182	0,07	0,48 %
Motstand	10 kΩ	6	0,03	0,0032	0,02	0,13 %
Sum					15,13	

Tabell 4.1.: Komponentkostnad for ZigBee-kortet. For kvar komponent er det gitt fem tal: Tal per kort, pris per stykk ved kjøp av 1 komponent og 1000 komponentar, pris per kort gitt kjøp av 1000 komponentar og prosent av prisen for eitt kort kvar komponenttype utgjer. Prisane er gitt i amerikanske dollar, henta frå [38].



Figur 4.3.: Målskisse for standard Elko-brytar. Henta frå [39].

norske hus nyttar brytarar frå Elko, og desse er 84 mm·84 mm, inkludert ramme, som vist i figur 4.3. [39] Ved å laga brytarnoden slik at han var litt mindre enn ein slik brytar, håpa ein å kunna nytta same deksel som desse brytarane, eventuelt med litt tilpassing.

Vidare seier kravlista (I 1) at alle inngangseiningar skal ha ein lysdiode for å kunna gi ei synleg tilbakemelding til brukaren. Ein enkel, lågenergi lysdiode vart difor plassert i midten av brytaren. [40]

Det var sett opp som eit krav at brytaren skulle ha fire uavhengige knappar. Som knappar vart det valt enkle, overflatemonterte trykknappar, produsert av Panasonic. [41]

For å kunne tilfredsstilla kravet om at systemet skal fungera uten heimesentralen, må konfigurasjonen kunna lagrast i nodane på ein måte. (S 6) Ein måte å løysa dette, er å lagra konfigurasjonen i den innebygde EEPROM[4]-en i mikrokontrollaren. EEPROM er skrivbart minne, som held informasjon utan straum, og Atmega128RFA1 har 4 kilobyte innebygd.

Ein viktig funksjon i systemet er trådlause lysbrytarar som kan plasserast der ein ønskjer, og eventuelt flyttast seinare. Dermed må brytarane vera batteridrivne. (I 5) Det finst svært mange forskjellige batteri på marknaden, men truleg er det beste alternativet såkalla AAA-batteri. Desse er svært lett tilgjegnelege, relativt små fysisk, men har relativt høg kapasitet, opp mot 1200 mAh.

Kravlista inneheld eit punkt om batterilevetid, og krev i utgangspunktet levetid på meir enn eitt år. (L 3) Med to AAA-batteri, er grensa for eitt år levetid på gjennomsnittleg 0,27 mA straumtrekk. Resultata frå prosjektoppgåva, tyder på at dette skal vera mogleg å oppnå. [2]

To seriekopla AAA-batteri gir ei spenning på 3 V når batteria er nye, ned mot i overkant av 2 V når batteria nærmar seg utbrukte. Drivspenninga for

[4]Electrically Erasable Programmable Read-Only Memory

Komponent	Type	Tal	1	1000	Sum	Prosent
Kontakt	JTAG	1	0,40	0,11418	0,11	1,51
	UART	1	0,22	0,08291	0,08	1,09
Batterihaldar	2xAAA	1	1,61	0,97	0,97	12,81
Knapp	-	4	0,29	0,1458	0,58	7,70
Lysdiode	-	1	0,16	0,08744	0,09	1,15
Motstand	1,5 kΩa	1	0,04	0,00448	0,00	0,06
	10 kΩ	4	0,03	0,0032	0,01	0,17
Kontaktar	Hankjønn	2	6,02	2,8595	5,72	75,51
Sum					7,57	

Tabell 4.2.: Komponentkostnad for brytarnoden. For kvar komponent er det gitt fem tal: Tal per kort, pris per stykk ved kjøp av 1 komponent og 1000 komponentar, pris per kort gitt kjøp av 1000 komponentar og prosent av prisen for eitt kort kvar komponenttype utgjer. Prisane er gitt i amerikanske dollar, henta frå [38].

Atmega128RFA1 er anbefalt å ligga mellom 1,8 og 3,6 V, og batterispenninga kan difor nyttast direkte, utan spenningsregulering. Batteri skal òg gi svært glatt spenning, og ein går difor ut frå at glattekondensatoren på sjølve ZigBee-kortet er stor nok til å gi stabil straumforsyning.

For å letta utviklinga, og fordi det er god plass på kretskortet for lysbrytaren, vart det lagt til ein UART-kontakt og eit JTAG programmeringsgrensesnitt på noden.

Tabell 4.2 viser komponentkostnaden for brytaren. Her er ikkje kostnaden for ZigBee-kortet teke med.

I vedlegg D er skjema og utlegg vist.

4.2.3. Relénode

Relénoden er maskinvaremessig den mest komplekse noden, og det følgjande delkapitlet er difor delt opp meir enn dei andre nodane. Denne noden skal fungera som utgang i systemet. I tabell 3.4 er krava for denne nodetypen gitt.

Storleik

Eit viktig punkt å legga merke til her, er kravet om storleik. (R 1) Det er sett som eit krav at noden skal kunna monterast i ein vanleg elektrisk boks saman med ein stikkontakt.

Dersom ein igjen ser på Elko-utstyr, har dei fleste boksane deira indre diameter på 71 mm og djupn på 57 mm. [42] Ein vanleg, dobbel stikkontakt er 20 mm djup, og dermed er det tilgjengeleg om lag 35 mm i djubderetning. [43] På denne plassen skal det òg vera plass til dei nødvendige kablane, og noden bør difor vera endå mindre enn dette.

Relé-funksjonalitet

Eit anna krav er at relénoden skal kunna slå av og på einingar som trekk opptil 16 A ved 230 V. (R 2) For å få til dette, er ein nødt til å bruka eit relé.

Som eit generelt krav er det sett at systemet ikkje skal trekka meir straum enn nødvendig. (S 3) I eit tradisjonelt relé må det gå ein kontinuerleg straum gjennom spolen for å halda releet i ein gitt tilstand. Å trekka ein kontinuerleg straum slik, gir i tillegg til høgare forbruk òg høgare varmeutvikling, og det er difor ikkje ønskeleg. Dersom ein i staden nyttar eit bistabilt relé[5], slepp ein dette, då slike relé berre trekk straum akkurat når dei skiftar tilstand.

Utvalet av bistabile relé som klarer straumar på 16 A, er ikkje veldig stort, men Tyco Electronics har ein serie med dei ønska eigenskapane. [44] Denne finst i versjonar med ein eller to spolar, med spolespenningar frå 3 V til 24 V. Talet på spolar avgjer korleis ein styrer reléet. I eit relé med éin spole må spenninga over spolen snuast for å skifta tilstanden til releet. Eit relé med to spolar har ein fast plusspol, medan kva for ein av dei to andre polane ein trekk til jord avgjer om releet slår på eller av.

I og med at mikrokontrollaren kan gå på 3 V, ville det vore ønskeleg å nytta denne varianten av releet i dette tilfellet, for å kunna nytta same spenninga i heile kretsen. Dessverre er tilsynelatande berre 12 V-utgåva med to spolar tilgjengeleg på marknaden i små leveransar. Relénoden vart difor designa med denne. I databladet er denne spolen spesifisert til å kreva minst 8,4 V for å slå på, og minst 6,6 V for å slå av. Dette gjer dermed at noden òg treng ein spenningskjelde med høgare nivå enn 3 V til kontrollaren.

Det at releet kan slå av og på 16 A set krav til banane på kretskortet. 16 A er mykje straum, og kan med for smale banar føra til varmgang og potensielt farlege situasjonar. For å unngå dette problemet vart kretskortet laga med breie banar, som det i tillegg vart lagt tinn på med loddebolt.

[5] Bistabile relé blir og kalla «latching» relé

Dette er nødvendig fordi koparlaget i seg sjølv er svært tynt. Eit anna aspekt ein må passa på er avstand mellom banar. Med spenningar på 230 V, kan ein få overslag om banane ligg for nære.

Straumforsyning

Relénoden skal ta straum frå nettet, og kombinert med storleikskrava, gjer dette at noden må ha ei eiga straumforsyning, og vera tilkopla 230 V-nettet. (R 2) Noden sjølv treng straum, i tillegg til at minst ein fase må bli broten av reléet. Dette gjer at ein minst treng tre tilkoplingar. For å gjera tilkoplinga meir intuitiv, valde ein å nytta ein kontakt med fire tilkoplingar, slik at ein må kopla begge fasane i 230 V-nettet både inn og ut av noden.

Det er eit krav at noden skal kunna slå av og på 16 A. Denne straumen må dermed gå gjennom noden, og gjennom tilkoplingane. Desse må difor tåla slike straumar. I elektriske anlegg i Noreg, skal det nyttast kablar med tversnitt på 2,5 mm^2 for kursar med 16 A sikringar. Ein valde difor ein type tilkoplingskontaktar frå On-Shore Technology, som har plass til kablar opp til denne storleiken. [45]

Det finst svært mange ulike konsept for å laga straumforsyningar. For å gjera det så enkelt som mogleg, laga ein i denne omgang ei tradisjonell straumforsyning, med ein transformator, ei likerettarbru og ein glattekondensator. For å dimensjonera komponentane, må ein vita kor mykje straum noden kjem til å trekka. Dei to komponentane som trekk mest straum, er mikrokontrollaren og reléet. Mikrokontrollaren er i databladet spesifisert til å trekka opp til 15 mA. [34] Releet har ein 12 V spole, med ein motstand på 240 Ω, noko som gir eit straumtrekk på $12/240 = 50$ mA. Dermed endar ein opp med eit maksimalt straumtrekk på 65 mA, pluss det som går til småkomponentar. Om ein nyttar lågare spenning over spolen, vil òg straumtrekket bli lågare.

Fordi 12 V-varianten av reléet vart nytta, finst det eigentleg to separate straumkrav for denne noden: reléet må ha straum til å skifta tilstand, som nemnt meir enn 8,4 V, og mikrokontrollaren må ha straum til å køyra, mellom 1,8 og 3,3 V.

Det vart vurdert ulike måtar å løysa dette på. Ein ende opp med å nytta ein transformator med enkel sekundærspole, som gav høg nok spenning til å driva reléet, samtidig som ho òg kunne regulerast ned til 3 V. Ved å ta straum til reléet rett etter glattekondensatoren, før regulatoren, burde begge krava dermed kunna tilfredsstillast. Dette vart vurdert til å gi totalt sett færre komponentar, og dermed enklare krets enn andre alternativ.

Transformatorar finst i mange storleikar, og må dimensjonerast etter utspenning og maksimalt straumtrekk. Lågare utgangsspenning for ein

transformator gir mindre fysisk storleik for same maksimale straumtrekk straum, og låg utgangsspenning er difor ein fordel. I dette tilfellet var det nødvendig at transformatoren tok inn 230 V, og gav ut noko som kunne regulerast ned til 3 V, samtidig som det var høgt nok til å driva reléet.

I marknaden finst ein transformator med maksimal straum på 83 mA og utgangsspenning på 6 V. [46] Dette er tilsynelatande for lite for releet, men ein må hugsa på at dette er ei vekselspenning. I databladet er det spesifisert at toppunktet for utspenninga ligg på $6\text{V} \cdot 1,8 = 10,8$ V, noko som er høgt nok. Ved å setja ein relativt stor kondensator etter diodebrua, rekna ein difor med at det ville gå bra.

På lik linje med transformatorar, blir òg diodebruer dimensjonert etter maksimalt straumtrekk. Krava i dette tilfellet var ikkje så store, og ein valde difor ei enkel bru frå Micro Commercial. [47]

Etter diodebrua må ein ha ein kondensator for å glatta ut spenninga. Storleiken på denne avgjer kor glatt innspenninga på regulatoren og drivspenninga for reléet blir, og det er difor eit krav at spenninga ikkje går under 8,4 V. Dersom ein ser i forhold til toppspenninga, 10,8 V, er dette eit dropp på 2,4 V. Fordi 10,8 V er toppspenning utan last, og truleg vil reduserast noko med last, valde ein å auka marginane. Ein dimensjonerte difor kondensatoren for å få eit maksimalt dropp på om lag 0,5 V.

Nettspenninga har konstant frekvens på 50 Hz, og denne blir ikkje endra av transformatoren. Likerettaren er ein fullbølgelikerettar, og perioden mellom toppane ut av denne blir difor $\frac{1}{2 \cdot 50\,\text{Hz}} = 10$ ms. Utlading av ein kondensator skjer etter likninga

$$V(t) = V_0 e^{-\frac{t}{R \cdot C}} \tag{4.2}$$

der R er lastmostand, C kondensatorkapasitans og V_0 startspenninga.

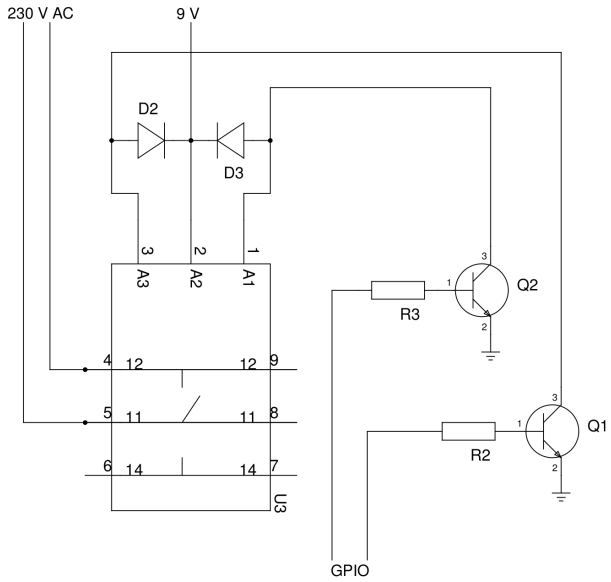
Spolemotstanden vart nytta som lastmotstanden i utrekninga. Dette er ei tilnærming, sidan regulatoren òg er ein del av lasta. Fordi målet her berre er å dimensjonera ein kondensator, og ein allereie har stor margin, ser ein vekk frå det her. Ein er altså interessert i å finna storleiken som er nødvendig for at spenninga ikkje skal droppa under $V_0 - 0,5$ V i løpet av 10 ms.

Ved å løysa likninga ovanfor med omsyn på C, får ein følgande

$$C = -\frac{t}{R \ln\left(1 - \frac{0,5}{V_0}\right)} \tag{4.3}$$

$$= -\frac{0,010}{240 \ln\left(1 - \frac{0,5}{10,8}\right)} \tag{4.4}$$

$$\simeq 860\,\mu\text{F} \tag{4.5}$$



Figur 4.4.: Styring av reléet, ved hjelp av to transistorar. Utsnitt av skjema vist i vedlegg D.

Det vart difor vald å nytta ein 1000 µF kondensator i kretsen, og ein får forhåpentleg ei glatta spenning på om lag 10 V over kondensatoren. Denne spenninga kan dermed nyttast direkte til å driva reléet.

For å få passande spenning for mikrokontrollaren, må ein dermed ha ein regulator. Ein valde her å nytta ein MIC5205 3 V regulator. MIC5205 er ein standard lineærregulator, som fungerer med høg inngangsspenning, har maksimalt straumtrekk på 150 mA og lågt spenningsfall for lette laster. [48]

Reléstyring

Som nemnt har reléet to spolar, ein som skal spenningsetjast for å slå på og ein som skal spenningssetjast for å slå av straumen gjennom det. Både spenninga og straumen reléet treng er større enn det mikrokontrollaren kan gi gjennom vanlege utgangspinnar, og ein må difor ha ekstra komponentar.

Det viste seg at ein kunne nytta BC847-transistorar til dette. Ved å kopla dei opp som vist i figur 4.4, oppnår ein ønska funksjonalitet. Når transistor Q1 er aktiv går det straum i eine spolen, når Q2 er aktiv går det straum i den andre spolen. Dermed kan ein frå utgangane av mikrokontrollaren

setja reléet i kva tilstand ein vil.

Om både Q1 og Q2 er aktive, er det vanskeleg å seia kva relétilstanden vil bli. Dette må ein difor unngå i programvare.

Diverse

I tillegg til dei nødvendige komponentane for å slå av og på straum, og konvertera nettstraumen til 3 V, krev spesifikasjonen òg at alle einingar skal ha ein knapp (S 7). Det vart òg valt å leggja til ein lågenergi lysdiode, for å kunna gi tilbakemelding til brukaren når knappen blir trykka. Ein UART-kontakt vart òg lagt til, for bruk i avlusing og testing, men av plassomsyn valde ein å ikkje ha programmeringsgrensesnitt på denne noden.

Det vart òg lagt til ein termistor på kretskortet, som kan nyttast til å måla temperatur. Temperaturmåling er nyttig både til å overvaka temperaturen under testing og eventuelt til regulering av romtemperatur i eit ferdig system.

Eitt av krava til utgangsnodar er òg at dei skal måla straumforbruket til det tilkopla utstyret. Det vart funne ein komponent som truleg er svært godt eigna til dette, Allegro ACS758. [49] Denne komponenten måler straumtrekk ved hjelp av Hall-effekt, og gir ut ei lineær, analog spenning, proporsjonal med straumtrekket.

Denne komponenten vart likevel ikkje teke med i den endelege designen. Hovudsakleg var årsaka til dette at komponenten er relativt stor fysisk, og det ville vore vanskeleg å få fysisk plass til han innan grensene krava gir. (R 1) I tillegg er komponenten relativt dyr, med ein kostnad på nesten 4 amerikanske dollar per stykk ved bestilling av totalt 1000 einingar. Om ein finn eit nytt konsept for straumforsyning, kan det henda dette vil gi betre plass, og gjera det enklare å integrera denne komponenten.

Komponentkostnadane for relénoden er viste i tabell 4.3, medan skjema er vist i vedlegg D.

4.2.4. Heimesentral

Heimesentralen skal tilby eit web-grensesnitt for styring og oppsett av systemet. For å gjera utviklinga av dette grensesnittet enklare, er det ønskeleg å ha eit tradisjonelt operativsystem å byggja på, då det gjer det mogleg å nytta kjente, eksisterande web-rammeverk. Eit slikt operativsystem gjer at ein får langt større krav til maskinvara i denne noden enn i dei andre.

Det vart konkludert med at det å laga eiga maskinvare med kraftig prosessor og operativsystemstøtte var for mykje arbeid for denne oppgåva. Ein ønska difor å finna eit ferdig kort med kraftig prosessor, ein del minne og god lagringskapasitet. Det er òg nødvendig med nettverkstilkopling

Komponent	Type	Tal	1	1000	Sum	Prosent
Relé	-	1	4,15	2,11925	2,12	17,94
Transformtator	-	1	4,5	2,25	2,25	19,04
Diodebru	-	1	0,44	0,12	0,12	1,02
Regulator	3 V	1	0,66	0,33	0,33	2,79
Diode	-	2	0,3	0,02795	0,06	0,47
Transistor	NPN	2	0,3	0,03546	0,7092	0,60
Knapp	-	1	0,29	0,1458	0,15	1,23
Lysdiode	-	1	0,16	0,08744	0,09	0,74
Kondensator	1 µF	1	0,09	0,00748	0,01	0,06
	1000 µF	1	0,72	0,2108	0,21	1,78
Termistor	10 kΩ	1	0,6	0,3	0,31	2,60
Motstand	1 kΩ	2	0,03	0,0032	0,01	0,05
	1,5 kΩ	1	0,04	0,00448	0,00	0,04
	10 kΩ	4	0,03	0,0032	0,01	0,05
Kontakt	230 V	1	0,49	0,2916	0,29	2,47
	UART	1	0,22	0,08291	0,7	
	ZigBee	2	6,02	2,8595	5,72	48,40
Sum					11,82	

Tabell 4.3.: Komponentkostnad for relénoden. For kvar komponent er det gitt fem tal: Tal per kort, pris per stykk ved kjøp av 1 komponent og 1000 komponentar, pris per kort gitt kjøp av 1000 komponentar og prosent av prisen for eitt kort kvar komponenttype utgjer. Prisane er gitt i amerikanske dollar, henta frå [38].

for å kunna koplast til internett, i tillegg til tilkoplingsmoglegheiter for å leggja til ZigBee-kommunikasjon. (H 1)

Eit kort kalla BeagleBoard -xM tilfredsstilte alle desse kriteria, med ein 1 GHz ARM Cortex A8-prosessor, 512 MB minne og MicroSD-kortspor for lagring. Kortet har òg nettverksport, USB-portar og støtte for fleire ulike kommunikasjonsgrensesnitt, mellom anna I^2C og UART. [50]

BeagleBoard er utvikla som open maskinvare, og alle skjemateikningar er fritt tilgjengelege, under Creative Commons-lisensar. Det finst òg svært mykje dokumentasjon tilgjengeleg på internett. Kortet har god støtte for Linux, noko som er eit godt grunnlag for både kommunikasjon med ZigBee-einінga og for å byggja web-grensesnittet. Andre NTNU-studentar har òg tidlegare brukt kortet, og hatt gode erfaringar med det. [51]

Prismessig ligg kortet på 149 amerikanske dollar. Som tabell 4.4 viser, er det ingen mengderabatt. Dette er bevisst frå produsenten si side, då dei ikkje ønskjer at Beagleboardet skal nyttast som det er i sluttbrukarprodukt. I staden skal Beagleboard vera ei enkel løysing for å koma i gang. For masseproduksjon er det difor nødvendig å skifta ut Beagleboardet i dette systemet med noko anna. Dette skal vera enkelt å gjera, så lenge ei ny plattform òg køyrer Linux og tilbyr den nødvendige maskinvare.

BeagleBoard er utstyrt med ein såkalla utvidingskontakt, der mange av kommunikasjonsgresesnitta og straum er tilgjengeleg. Både UART og I^2C vart vurdert som grensesnitt mellom Linux og mikrokontrollaren, og begge desse vart kopla opp på kretskortet, I^2C med sine såkalla pull-up-motstandar på busslinjene.

Utvidingskontakten gir straum i form av ei 5 V- og ei 1,8 V-linje. 1,8 V-linja er meint som ei referansespenning, og kan ikkje levera nok straum til drift. Komponentane på BeagleBoardet køyrer på 1,8 V, og ved å lata mikrokontrollaren òg gjera det, slepp ein nivåkonvertering på UART- og I^2C-bussen. Ein valde å nytta ein Microchip TC1014 1,8 V regulator. [52]

I tillegg til regulator og kontaktar, vart det òg på dette kortet plassert ein liten lysdiode. UART- og JTAG-kontaktar vart lagt til, til bruk i prograrmmering, avlusing og testing. Det vart ikkje plassert nokon knapp på kortet, fordi web-grensesnittet skal vera det primære brukargrensesnittet.

Fullstendig skjema og utlegg er vist i vedlegg D.

4.3. Programvare

I det følgande blir strukturen i dei ulike programvaredelane systemet treng skissert. All programvare i sjølve nodane køyrer på ZigBee-kortet.

Komponent	Type	Tal	1	1000	Sum	Prosent
BeagleBoard -xM	-	1	149,00	-	-	-
Kontakt	BeagleBoard	1	2,99	1,16964	1,17	15,69
	JTAG	1	0,40	0,11418	0,11	1,53
	UART	1	0,22	0,08291	0,08	1,11
	ZigBee	2	6,02	2,8595	5,72	76,70
Lysdiode	-	1	0,16	0,08744	0,09	1,17
Kondensator	1 µF	2	0,052	0,01121	0,02	0,30
Motstand	1,5 kΩ	1	0,04	0,00448	0,00	0,06
	10 kΩ	2	0,03	0,0032	0,01	0,09
Sum					7,46	

Tabell 4.4.: Komponentkostnad for heimesentralen. For kvar komponent er det gitt fem tal: Tal per kort, pris per stykk ved kjøp av 1 komponent og 1000 komponentar, pris per kort gitt kjøp av 1000 komponentar og prosent av prisen for eitt kort kvar komponenttype utgjer. Prisane er gitt i amerikanske dollar, henta frå [38].

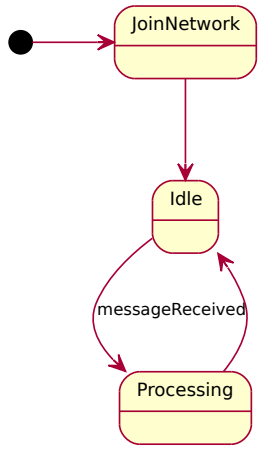
4.3.1. Relénode

Relénoden er truleg den enklaste eininga programvaremessig. Det einaste denne nodetypen treng å gjera, er å starta opp, bli med i ZigBee-nettverket, og senda ei melding til heimesentralen om at han har starta. Deretter må han venta på å motta kommandoar.

Slik systemet er bygd opp, har utgangsnodar ingen konfigurasjon å ta vare på, men skal berre motta kommandoar og behandla dei. I og med at dei fysiske nodane som blir laga i denne oppgåva berre støttar av/på-funksjonalitet, blir òg sjølve meldingsutvekslinga enkel. Ein utgangsnode treng berre å kunna ta i mot tre typar kommandoar: slå av, slå på og skift tilstand.

Relénoden er maskinvaremessig designa med tilgang på straum. Dette gjer det mogleg for denne nodetypen å vera med i rutinga i nettverket. Dette er likevel ikkje noko brukarapplikasjonen treng å ta omsyn til, sidan dette blir gjort på eit lågare nivå enn brukarapplikasjonen.

I figur 4.5 er eit enkelt, overordna tilstandsdiagram for relenoden vist.



Figur 4.5.: Overordna tilstandsdiagram for relénoden.

4.3.2. Lysbrytarnode

Programvaremessig er lysbrytarnoden noko meir komplisert enn relénoden. Ein lysbrytarnode vil starta så fort han får batteri. Under oppstart skal han melda seg inn i eit ZigBee-nettverk. Om noden har ein lagra konfigurasjon, skal denne lastast frå EEPROM og inn i minnet, før noden går i sovemodus.

Konfigurasjon vil i dette tilfellet seia informasjon om kva nodar dei fire ulike knappane på brytaren skal styra, og kva type melding som skal sendast ved knappetrykk. Det er sett som eit krav at ein knapp skal kunna styra fleire utgangar.

Medan han køyrer, må noden vera i stand til å ta i mot ein konfigurasjon over nettverket. Konfigurasjonen for kvar knapp må lagrast i EEPROM, uavhengig av kvarandre.

Kvar gong brukaren så trykker på ein knapp, skal noden vakna og prosessera knappetrykket. Dette vil typisk innebera å senda ei melding til den noden knappen er konfigurert til å styra, med innhald basert på den lagra konfigurasjonen.

Batterilevetid er svært viktig for lysbrytarnode, og dette får òg innverknad på programvara. Det som har størst påverknad på levetida, er forholdet mellom tida noden er aktiv og tida noden søv. Det er difor eit mål å la noden sova størst mogleg del av tida. Etter at handlingar er utført, må noden difor gå tilbake til å sova så fort som mogleg.

Lysbrytarnoden bør vita kor mykje spenning batteria har, slik at den-

ne informasjonen kan presenterast for brukaren, og han kan vita når det er tid for å skifta dei. Dette kan løysast ved å bruka analog-til-digital-konverteraren til å måla ei intern referansespenning i mikrokontrollaren med drivspenninga som referanse. [53] Denne målinga må så gjerast tilgjengeleg frå heimesentralen.

Figur 4.6 viser eit UML-diagram for lysbrytarnoden si overordna tilstandsmaskin.

4.3.3. Heimesentral

Heimesentralen er den viktigaste og mest komplekse delen av systemet. Programvara her skal la brukaren både konfigurera og styra utstyret sitt.

Maskinvareløysinga som er valt gjer at ein må skilja mellom det som køyrer på Beagleboardet, og det som køyrer på mikrokontrollaren. I skildringa av designen, er det difor naturleg å starta med det som køyrer på sjølve ZigBee-kortet.

ZigBee-kortet

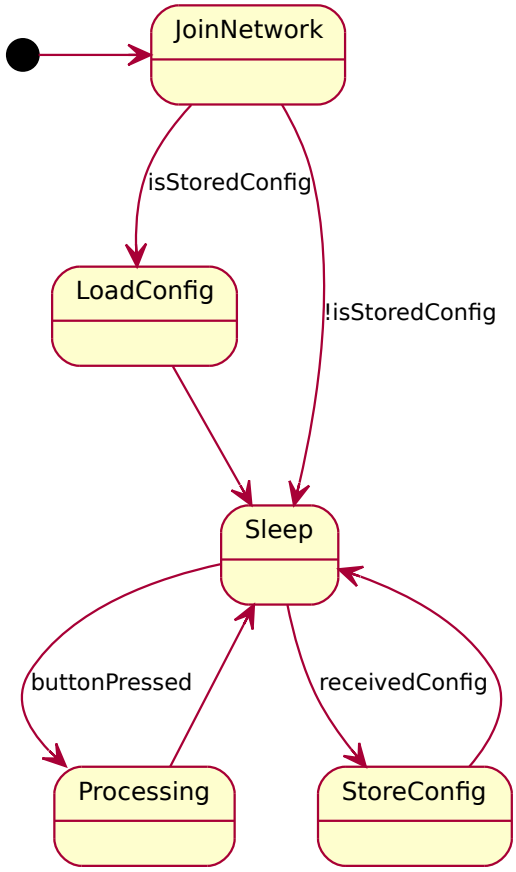
Heimesentralen nyttar det same ZigBee-kortet som dei andre nodane, og både mikrokontrollaren og Beagleboardet støttar fleire ulike grensesnitt for kommunikasjon mellom dei. Kva ZigBee-kortet skal gjera, er likevel uavhengig av kva grensesnitt ein nyttar.

ZigBee-kortet må kunna starta nettverket, og deretter overlata det meste av styringslogikken til programvara som køyrer på Beagleboardet. Når det skjer ulike hendingar i ZigBee-nettverket, må mikrokontrollaren senda ei melding om dette til programvara på Beagleboardet. Motsett må heimesentralen kunna få ting til å skje i ZigBee-nettverket, ved å senda meldingar til ZigBee-kortet.

For å gjera systemet mest mogleg fleksibelt, er det ønskeleg at programvara på mikrokontrollaren òg klarer å behandla meldingar han ikkje er førebudd på. Likevel gjer strukturen i ZigBee, med meldingssending til førehandsdefinerte endepunkt, dette litt vanskeleg. Dette kjem ein tilbake til i delkapittel 5.1.

Operativsystem for Beagleboardet

I kravlista for heimesentralen heiter det at han skal tilby eit web-grensesnitt for styring og konfigurasjon. (H 2) Som nemnt gjer dette at det er ønskeleg å ha eit meir skikkeleg operativsystem i botnen enn det ein tradisjonelt har nytta på mikrokontrollarar. I og med at heimesentralen held seg unna alt



Figur 4.6.: UML-diagram som viser den overordna tilstandsmaskina for ein lysbrytarnode.

som har med faktisk respons på brukarhandlingar å gjera, har ein ingen harde sanntidskrav i denne.

Valet av Beagleboard som maskinvareplattform gjer det attraktivt å nytta Linux som operativsystem. Linux er tilgjengeleg under GPL, og har difor ingen lisenskostnadar knytta til seg. Derimot må ein gjera eventuelle modifikasjonar av Linux tilgjengelege, men dette er lite relevant for dette systemet. I tillegg finst det ei lang rekke webrammeverk som primært køyrer på Linux.

Kommunikasjon med ZigBee-noden

Ein kunne tenkt seg at ein overlot kommunikasjon med ZigBee-kortet til kvart enkelt Linux-program som trengte å snakka med ZigBee-nettverket. Dette framstår likevel som ei relativt dårleg løysing, sidan det i så fall blir svært tett knytting mellom alle program som køyrer på Beagleboardet, og programmet som køyrer på ZigBee-kortet. Dermed er det vanskeleg å gjera endringar i grensesnittet mellom dei.

I staden laga ein eit lite program, som har som einaste oppgåve å formidla informasjon frå ZigBee-nettverket til program som køyrer på Beagleboardet og motsett. Dermed kan anna programvare på Beagleboardet designast mot grensesnittet til dette programmet, som kan vera uavhengig av kommunikasjonen med ZigBee-kortet. Ei eventuell endring i kommunikasjonsgrensesnittet mellom ZigBee-kortet og Beagleboardet får dermed berre konsekvensar for koden på mikrokontrollaren og dette programmet, i staden for at òg dei enkelte klientapplikasjonane må endrast.

Det er ønskeleg at dette programmet òg har ei form for lagringsmekanisme, slik at det til ei kvar tid til dømes veit kva nodar som finst i systemet. Det gjer programmet i stand til å tilby eit rikare programmeringsgrensesnitt, og vidareformidla informasjon òg til klientprogram som ikkje nødvendigvis alltid køyrer.

Web-grensesnitt

Ved å skilja sjølve kommunikasjonen med ZigBee-kortet ut i ein eigen prosess, blir webgrensesnittet svært mykje enklare. I struktur dreier det seg om å lytta etter dei signala som nodekommunikasjonen gir, og visa fram informasjonen som ligg i desse på ein fornuftig måte, inkludert å gi moglegheit til å styra einingane. I tillegg må grensesnittet la brukaren setja opp systemet og knytta saman einingar slik han ønskjer.

4.4. Brukargrensesnitt

Dersom ein ser på systemet som heilskap, har det hovudsakleg to brukargrensesnitt: dei fysiske knappane, på til dømes brytarnoden, og webgrensesnittet heimesentralen tilbyr. Til fysiske knappar knyttar det seg forventingar til funksjon, og ein bør i stor grad følga desse. Eit døme på slike, kan vera forventinga om at ein brytar rett innanfor ei dør til eit rom, slår på lyset i dette rommet. I forhold til webgrensesnittet har ein derimot langt friare rammar. Det vart difor sett på som fornuftig å tenka gjennom på førehand korleis dette burde sjå ut og fungera.

Som utgangspunkt for dette, vart det tenkt at alle brukarhandlingar med eit slikt system kan delast mellom to ulike konsept: oppsett og styring.

4.4.1. Oppsett

Med oppsett meiner ein her å knytta funksjonalitet til inngangane i systemet, til dømes knappane på brytarnoden. Grensesnittet må her gjera det mogleg å velja kva utgangar ein spesifikk inngang skal styra, på ein intuitiv og brukarvenleg måte.

I denne situasjonen er ein primært interessert i inngangseiningane i systemet. Utgangseiningane har, som nemnt i delkapittel 4.2, ingen lagra konfigurasjon, og det er dermed i utgangspunktet ikkje noko som kan setjast opp med desse. Eit unntak kan vera moglegheiter for å setja namn på einingane, og knytta dei til rom eller grupper.

Dersom ein ser for seg eit meir utvida system enn det som er tenkt laga i denne oppgåva, er det ikkje gitt at inngangane er direkte brukarstyrde. Til dømes kan ein tenka seg at ein ved hjelp av magnetbrytarar i vindauget vil sørga for at omnen slår seg av når vindauget blir opna. Dette må kunna setjast opp frå same grensesnitt som vanlege brytarar.

I mange tilfelle vil det òg vera nyttig å kunna definera fleire utgangar saman i ei gruppe, som kan styrast saman. Her kan ein til dømes tenka seg å gruppera alle omnar i eit rom, slik at ein enkelt kan slå på eller av alle på ei gong, eller la dei bli styrt felles av same termostat.

Ein kan òg tenka seg eit ønske om å setja alle einingar til ein bestemt tilstand. Denne funksjonaliteten blir i andre system ofte kalla scenario, og tanken er at ein til dømes kan ha eit «film-scenario», som slår av alle lys, trekk føre gardina og slår på prosjektor eller fjernsyn, eller eit vaskescenario, som slår på alle lys på maks.

Det verka naturleg å skilja oppsettet for einingar, grupper og scenario frå kvarandre, og dermed enda ein opp med ei skisse som i figur 4.7 og 4.8. Her er det vist skisser for korleis ein kan sjå for seg grensesnittet for å setja

opp både einingar og grupper. Scenario kan tenkast å bli relativt likt som gruppeoppsett.

4.4.2. Styring

Styring dreier seg i motsetnad til oppsett om den daglege bruken av systemet, om å faktisk styra det. Ein må gå ut i frå at styring er ein meir brukt funksjon enn å setja opp ting, og kanskje er det difor endå viktigare å halda brukargrensesnittet enkelt her.

Det er òg her interessant å leggja merke til kva einingar ein treng å sjå. Ved oppsett er det tydeleg at det er inngangar ein treng å ha tilgang til, sidan det er desse ein kan påverka. Dette blir ikkje like tydeleg når det gjeld styring. Ein kan nemleg tenka seg å styra både utgangar direkte, og å «trykka» på til dømes fysiske knappar på inngangsnodar frå webgrensesnittet. Begge deler bør difor vera mogleg.

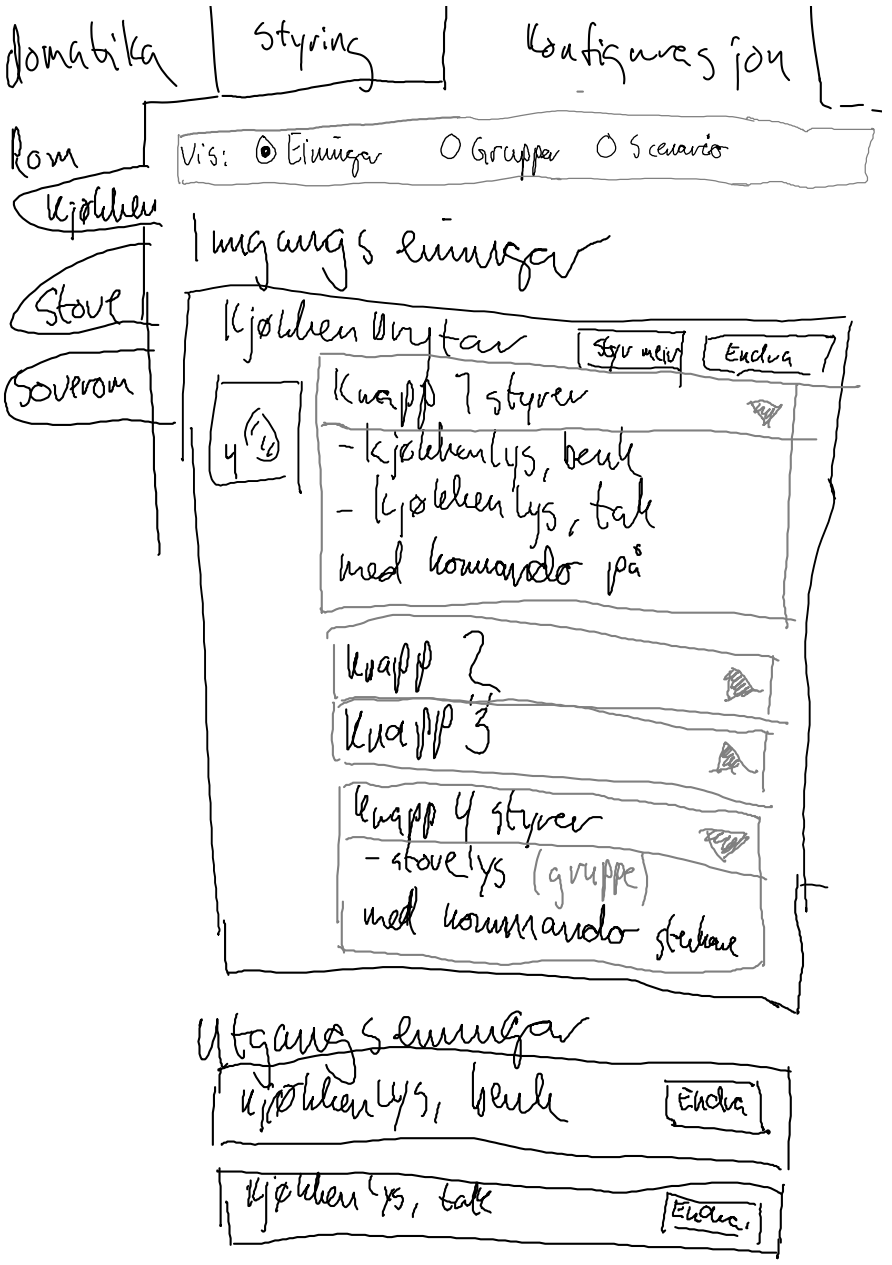
I tillegg til å styra enkelteiningar, eller trykka på «knappar», må systemet òg gjera det mogleg å styra ei gruppe saman, eller å setja systemet til eit scenario.

Figur 4.9 viser ei skisse over korleis eit grensesnitt med desse moglegheitene. Her er det haldt same skiljet mellom styring og oppsett som i brukargrensesnittet skissert i figur 4.7 og 4.8.

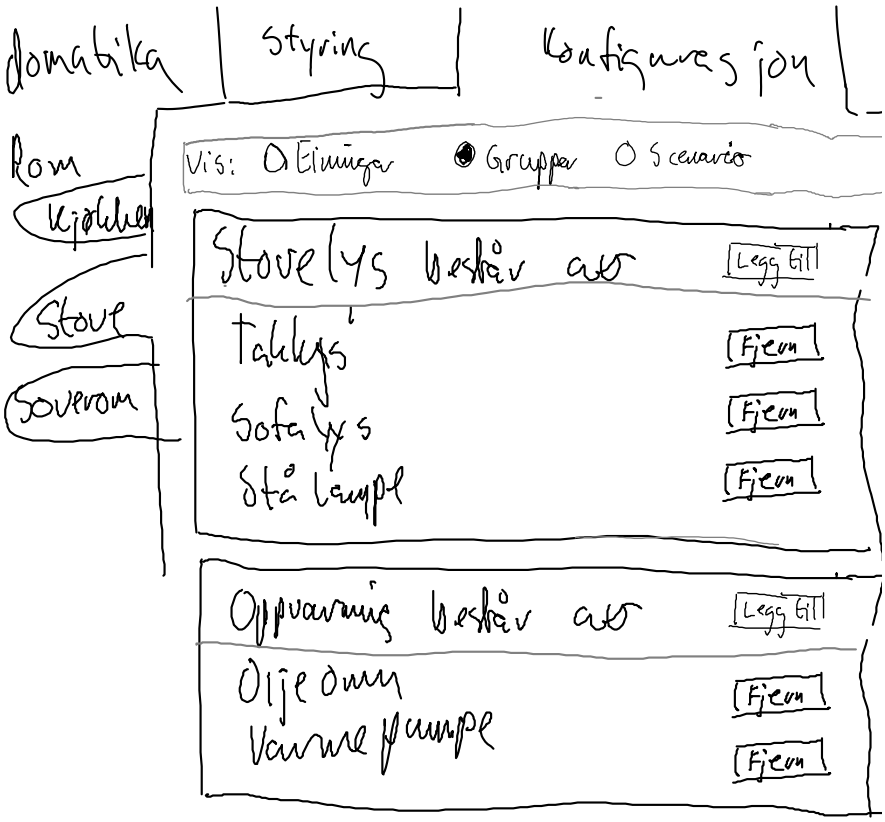
Det verkar naturleg å kunna dela einingar mellom ulike rom, slik at lysa i stova ikkje er det øvste valet når ein vil slå på lyset på soverommet. Til venstre kan ein difor velja mellom ulike rom i systemet. På sikt kan ein tenka seg moglegheiter for å posisjonera eininga som no viser fram grensesnittet, og basert på det visa val for riktig rom.

Sentralt i figuren er lista over einingar ein kan styra. Her er det vist scenariostyring på toppen, med styring av ulike einingar nedover. «Sofakrok» er her tenkt som ei gruppe, der ein har moglegheit til å gå inn og styra enkelteiningar. Skissa viser òg eit forslag til korleis ein kan sjå for seg styring av kontinuerlege einingar, sjølv om dette ikkje blir implementert i denne omgang.

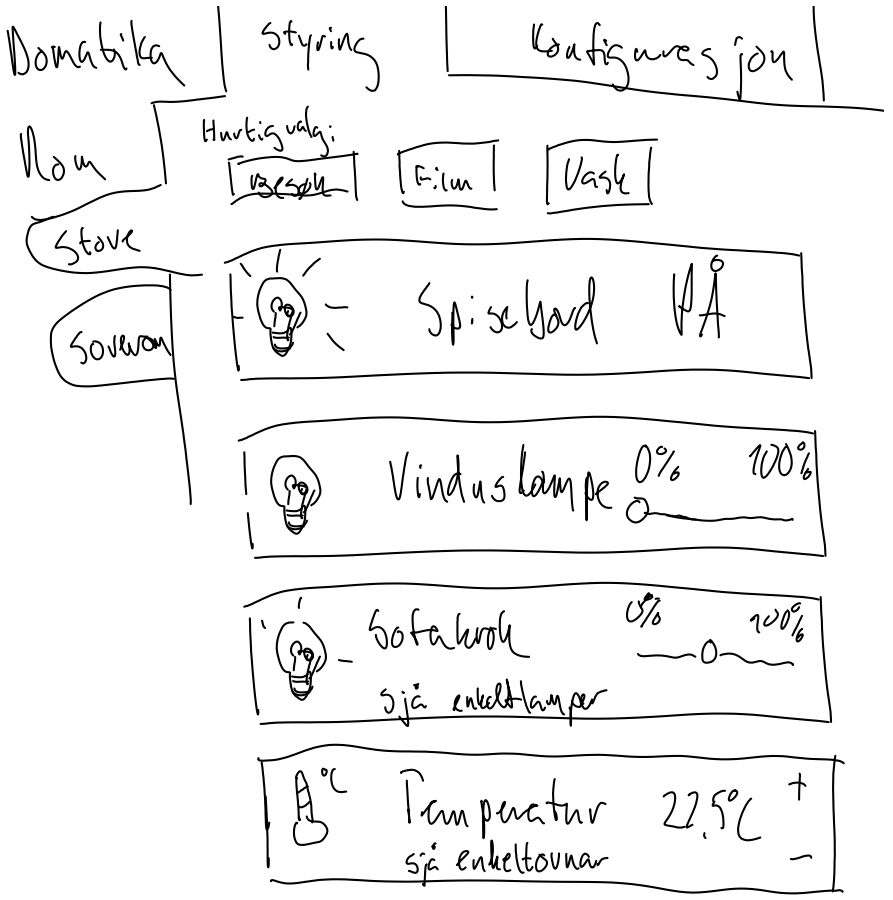
Nedst i biletet er temperaturinnstilling vist, òg denne med moglegheit til å styra enkeltomnar. Det er vist ikon for alle einingar, og det er tenkt at ein kan nytta dette til å visa noverande tilstand på ein enkel måte.



Figur 4.7.: Skisse for brukargrensesnitt for oppsett av enkelteiningar.



Figur 4.8.: Skisse for brukargrensesnitt for oppsett av grupper.



Figur 4.9.: Ei skisse med forslag til brukargrensesnitt for styring.

5. Implementasjon

Å laga fysiske nodar som faktisk kan testast og brukast, er ein viktig del av oppgåveteksta. I det følgande kapitlet, blir det skissert korleis arbeidet med dette gjekk føre seg.

For å betre forstå oppbygginga av programvara, er det nødvendig å gå noko nærare inn på korleis ZigBee er bygd opp. Dette får nemleg konsekvensar for strukturen i den utvikla programvara.

Den vidare implementasjonen har to ganske ulike fasar: maskinvare og programvare. For å ha noko å testa programvara på, framsto det naturleg å starta implementasjonen med maskinvara.

5.1. Meir om ZigBee

ZigBee er ein standard for trådlaus kommunikasjon, basert på IEEE 802.15.4, i 2,4 GHz-bandet. I prosjektoppgåva er det gjort ein teknisk gjennomgang av ZigBee, og dette vil ikkje bli gjenteke her. [2] I staden er det nødvendig å sjå nærare på korleis konsept frå ZigBee-standarden og -miljøet får konsekvensar for programvara. Det er difor nødvendig å gå inn på omgrepa profil, klynge (cluster) og endepunkt, samt binding, adressar og sikkerheit.

5.1.1. Endepunkt

Alle ZigBee-nodar har eitt eller fleire endepunkt. Som namnet delvis tilseier, er eit endepunkt eit element ein kan senda data frå og til. På mange måtar kan ein i ZigBee seia at det ikkje er nodane som kommuniserer, men endepunkta som køyrer på nodane.

Eit endepunkt har visse eigenskapar knytta til seg. Til dømes kan eit endepunkt støtta ei eller fleire klynger. Eit endepunkt støttar ei klynge anten som klient, tenar eller begge.

Eit endepunkt er òg knytta til ein profil, og har eit identifikasjonsnummer som er unikt innan denne profilen. Ein profil kan vera offisiell, godkjent og utgjeven av ZigBee Alliance, eller produkt- eller produsentspesifikk. Dei offisielle profilane er bygd på ZigBee Cluster Library (ZCL).

Det skal nemnast at endepunktet er det einaste kommunikasjonskonseptet ein er nødt til å forhalda seg til som ZigBee-brukar. Som nemnt går

nemleg all kommunikasjon frå og til spesifikke endepunkt, og ein må difor byggja opp koden sin rundt dette. Både klynger og standardiserte profilar kan ein derimot lata vera å ta omsyn til, om ein skal laga eit lukka nett, berre for eigne einingar.

5.1.2. Klynger

Før ein går vidare på kva ZCL er, må omgrepet klynge forklarast. I ZigBee-terminologi kan ei klynge seiast å vera eit slags kommunikasjonsgrensesnitt. Alle klynger er definert med ein tenar- og ein klientdel, og tilbyr visse kommandoar og attributtar. Mange klynger tilbyr ulik funksjonalitet på tenar- og klientsida.

Ein kommando er ei melding ein kan senda til ein node for å få noko til å skje, medan ein attributt er ein verdi som kan lesast, eller eventuelt setjast. Funksjonalitet for å manipulera attributtar er gitt i ZigBee Cluster Library.

5.1.3. ZigBee Cluster Library (ZCL)

ZigBee Cluster Library er eit bibliotek av standardfunksjonalitet, som er nødvendig for å implementera andre standardiserte klynger. Biblioteket tilbyr funksjonalitet på fleire nivå. Til dømes inneheld det funksjonalitet for å lesa og setja attributtar. Dette er svært grunnleggjande funksjonalitet, som kan nyttast både av offisielle og private klynger. Dette er dermed grunnlag for dei standardklyngene ZCL definerer, som har sine tilhøyrande kommandoar og attributtar.

Eit døme på ei slik standardklynge er ei klynge som gjer det mogleg å spørja ei eining etter informasjon. Slik informasjon kan til dømes vera kva endepunkt ho tilbyr, kva klynger desse støttar, om eininga går på nettstraum eller batteri eller liknande.

Mange brikkeprodusentar tilbyr ZCL som eit ferdig sett metodar og grensesnitt, som enkelt kan nyttast i eigen kode.

5.1.4. Profilar

Det finst som nemnt to ulike typar profilar: offisielle, utgitt av ZigBee Alliance, og produkt- og produsentspesifikke. I og med at klynger er unike innan ein profil, kan ein eigentleg sjå på dei profilane som ikkje er offisielle som ein måte å unngå at bruk av ZigBee-utstyr frå ulike produsentar saman gir konflikter.

Dei offisielle er derimot meir interessante, då det er dei som gjer det mogleg å nytta ulike produsentar sitt utstyr saman. Det finst fleire ulike

profilar, mellom anna «Smart Energy», «Home automation», «Remote control» med fleire. Poenget er at ein leverandør kan få sine produkt sertifisert etter ein profil, i meininga at dei støttar alle kommandoar og attributter denne definerer. Ulike einingar som støttar same profil skal kunna nyttast saman.

Dei offisielle profilane er baserte på bruk av ZigBee Cluster Library, for å tilby grunnfunksjonaliteten nemnt i forrige delkapittel.

5.1.5. Adressering

I forhold til adressar, er det verdt å merka seg at eit ZigBee-nettverk opererer med to typar adressar: 64-bit såkalla IEEE-addressar og 16-bit nettverksadressar. IEEE-addressen er konstant for ei gitt eining, og kan ikkje endrast etter kompilering.

Nettverksadressen er dynamisk, og blir tilfeldig generert kvar gong ei eining blir med i eit nettverk. Det er berre nettverksadressen som kan nyttast til sending av pakkar, og nodar må difor halda ein tabell over koplinga mellom IEEE-adressar og nettverksadressar i minnet. Første gongen ein sender ei pakke til ei eining ein er bunde til, kan ein dermed måtta gjera ei ekstra spørjing for å finna kva nettverksadresse IEEE-adressen høyrer til.

5.1.6. Binding

Eit anna konsept ZigBee tilbyr, som blir nytta i denne oppgåva, er binding. Binding er å knytta saman to endepunkt, slik at ein seinare kan senda meldingar mellom desse utan å spesifisera adresse. Dette er mogleg ved at adressa ein vil senda til blir lagra i ein såkalla bindingstabell.

Ei binding er uavhengig av dei såkalla nettverksadressane, då ho er basert på IEEE-addressane.

5.1.7. Sikkerheit

ZigBee tilbyr to ulike former for sikkerheit: såkalla «standard»-modus og «High security»-modus. «High security»-modus er eit tillegg til den opphavlege ZigBee-spesifikasjonen, og kom med ZigBee 2007-revisjonen. I eit ZigBee-nettverk skal det eksistera eit tillitssenter, «Trust Center», som har ansvar for det som har med sikkerheit å gjera. Oftast fungerer koordinatoren òg som tillitssenter.

I standard-modus nyttar nettverket ein nettverksnøkkel til å sikra kommunikasjonen mellom nodar. Denne nøkkelen er fastsett for eit gitt nettverk, og kan ikkje endra seg.

«High security»-modus nyttar derimot tre ulike typar nøklar: såkalla «master»-nøklar, «link»-nøklar og nettverksnøklar. Ein master-nøkkel blir brukt til å sikra utvekslinga av nettverksnøkkel. Ein link-nøkkel er ein nøkkel som er unik mellom to nodar, og berre blir brukt mellom desse. I høgsikkerheitsnettverk blir òg nettverksnøkkelen endra med jamne mellomrom.

Totalt sett finst det dermed langt fleire nøklar å halda styr på i eit høgsikkerheitsnettverk, og dette gir høgare krav til minnestorleik for nodane.

For å bli med i eit standardnettverk må ein node på førehand ha den korrekte nettverksnøkkelen. Denne må sendast til tillitssenteret for godkjenning, og dette skjer ukryptert i standard-nettverk. Dermed kan ein inntrengjar lytta på ei slik overføring, for å skaffa seg den rette nøkkelen. I eit høgsikkerheitsnettverk skjer aldri dette, fordi ein nyttar masternøkkelen for å gjera ei sikker nøkkelutveksling.

5.2. Maskinvareproduksjon

Å laga maskinvare består av to steg: laga kretskort og lodda på komponentar. Kretskorta som var nødvendige i nodane, vart delvis produsert på verkstaden til Institutt for teknisk kybernetikk (ITK), og delvis ved hjelp av eksterne produsentar.

5.2.1. Utstyr på verkstad

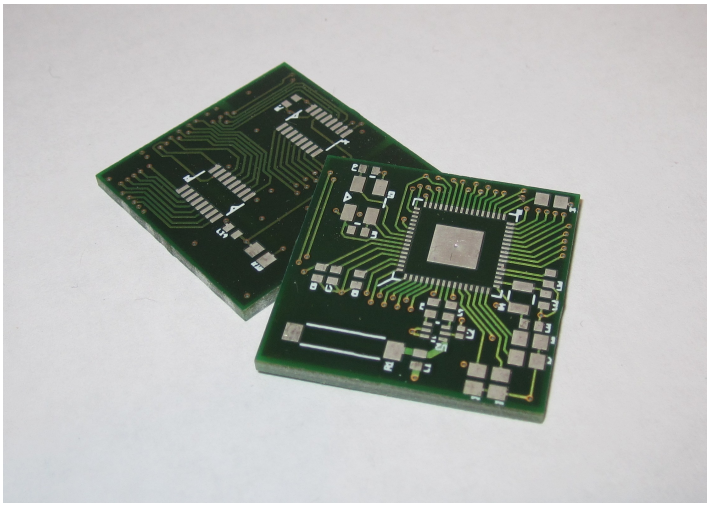
På verkstaden vart det nytta ein LPKF-fresemaskin. Denne borar hol og fresar ut spor og område i koparen. Ved å nytta plater med kopar på to sider, og snu plata midtvegs i prosessen, kan ein enkelt laga tolags kretskort.

Når maskinen har bora eit hol, er det ikkje kopar i holet, og dermed ikkje kontakt mellom dei to sidene av kortet. For å oppretta slik kontakt, må ein slå i ein såkalla via manuelt etterpå.

Maskinen kan laga banar ned mot 0,1 mm, men fordi det ikkje var så strenge storleikskrav vart det nytta 0,2 mm banar på korta som vart laga her. Via-hola som vart nytta var frå 0,6 til 1,2 mm, med tilhøyrande holstorleikar frå 0,9 til 1,7 mm. Dei stadane der ein berre trengte samband mellom dei to laga vart det nytta 0,6 mm, medan dei større vart nytta der det skulle loddast på gjennomgåande komponentar.

5.2.2. Ekstern produsent

For dei delane av kretsen som har sjølve ZigBee-kommunikasjonen var det, som nemnt i kapittel 4.2.1, nødvendig å ha firelags kretskort.



Figur 5.1.: Kretskorta som vart produsert eksternt.

Etter vurdering av fleire potensielle leverandørar av slike kretskort, vart pcb-pool.com valt. [54] Denne bedrifta kunne levera små seriar med kretskort relativt rimeleg, med banar ned til 0,150 mm og via-hol på ned til 0,3 mm.

Bestilling skjedde via nettsida deira, ved at ein lastar opp såkalla Gerber-filer. [54] gEDA-pakka har innebygd støtte for å laga slike filer. Leveringstid kan veljast under bestilling, og kortare leveringstid gir høgare pris. Å fullføra designen av ZigBee-korta tidleg i arbeidet, gjorde det mogleg å redusera kostnaden.

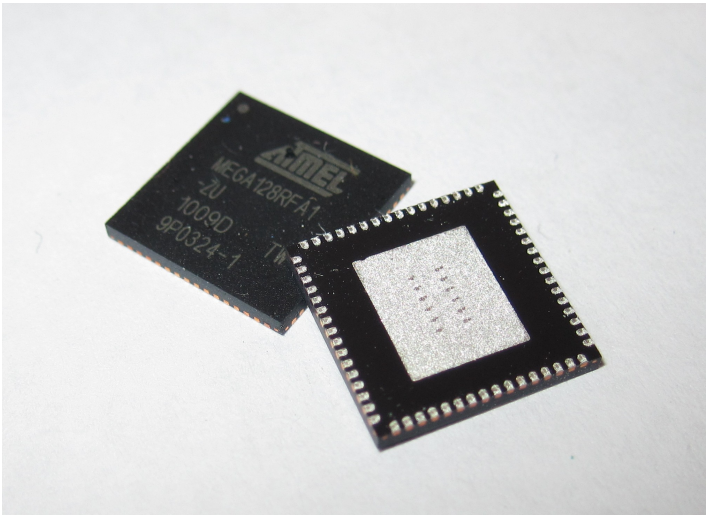
Det ferdige kretskortet er vist i figur 5.1.

5.2.3. Lodding

Dei kretskorta som vart laga på verkstaden til ITK vart òg lodda der, ved hjelp av ein vanleg loddebolt med fin spiss, loddepasta og forstørringskamera som finst der. Dette fungerte relativt bra for desse korta.

ZigBee-kortet vart òg prøvd lodda på denne måten. Mikrokontrollaren som vart nytta, Atmega128RFA1, finst berre i såkalla QFN64[1]-pakke, og på slike er det berre 0,5 mm mellom pinnane. Kontrollaren er vist i figur 5.2. John Olav Horrigmo, som arbeider på verkstaden, anbefalte å lodda denne ved å først leggja tinn på alle 64 pinnar, så skjera vekk overflødig tinn, før

[1] «Quad flat pack no leads», med 64 pinnar



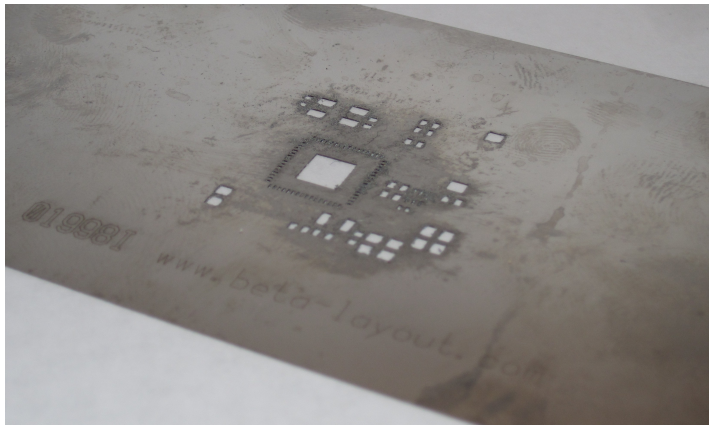
Figur 5.2.: Atmels Atmega128RFA1 mikrokontrollar.

ein set kontrollaren nedpå kretskortet og varmar kvar pinne igjen, slik at han sit fast.

Denne metoden fungerte godt, men tok svært lang tid. Det var difor ønskeleg å finna ein raskare metode, då det skulle lagast minst tre kort av denne typen. På Omega Verkstad vart det anbefalt å leggja pasta under alle pinnane på mikrokontrollaren, og deretter varma heile kretskortet. På den måten smeltar pastaen, og trekk seg til dei rette plassane på grunn av overflatespenninga. Resultatet er at ein kan lodda alle pinnane på mikrokontrollaren, samt utvalde andre komponentar på ei gong. Resten av komponentane loddar ein med vanleg bolt etterpå. Òg denne metoden vart prøvd, og fungerte godt.

Saman med dei bestilte kretskorta kom det òg ein såkalla stensil, som vist i figur 5.3. Ein stensil er ei plate, i dette tilfellet av aluminium, som har hol der det er skal loddast på komponentar på kretskortet. Tanken med dette er at ein nyttar ein spatel til å stryka loddepasta utover ein stensil, plassert over eit kort. Dermed får ein loddepasta på alle kontaktflater på kortet på ei gong. Ved å deretter plassera komponentar på alle desse flatene, kan ein varma opp og lodda alle komponentar på kortet på ei gong.

I produksjon nyttar ein omnar med nøyaktig temperaturregulering til slik oppvarming, men då ein ikkje hadde tilgang på dette her, valde ein å nytta ein ordinær steikeomn. Det viste seg at om lag 10 minutt på 270 grader celsius var passeleg, og gav svært fine loddingar.



Figur 5.3.: Stensilen som vart nytta til å lodda dei ferdigproduserte kretskorta.

Eit felles problem med både Omega Verkstad-metoden og stensilmetoden, er at ein ikkje på ein enkel måte kan lodda komponentane på baksida samtidig med framsida. I dette tilfellet har kortet så få komponentar på baksida at ein enkelt kunne gjera dette for hand etterpå, så det vart ikkje gjort store forsøk på å gjera dette enklare.

5.2.4. Ferdig maskinvare

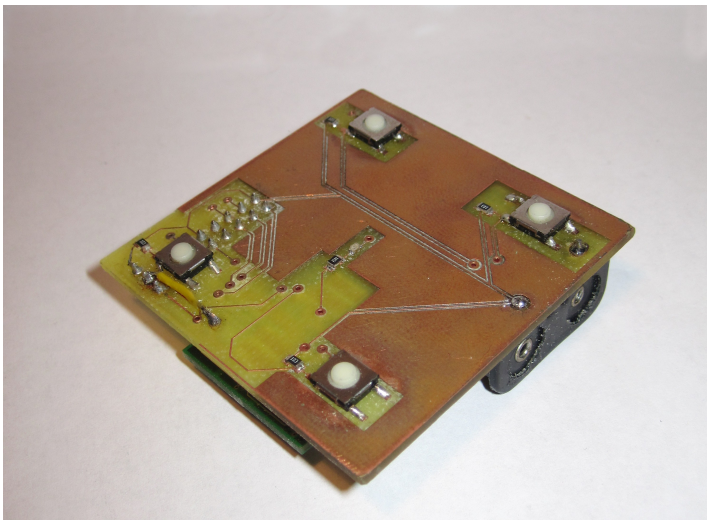
Framsida av dei ferdige nodane er viste i figurane 5.4, 5.5 og 5.6. Baksida av nodane er vist i figur 6.1, som òg viser tilkoplinga av ZigBee-kortet til funksjonskorta.

5.3. Grunnlag for eigen programvare

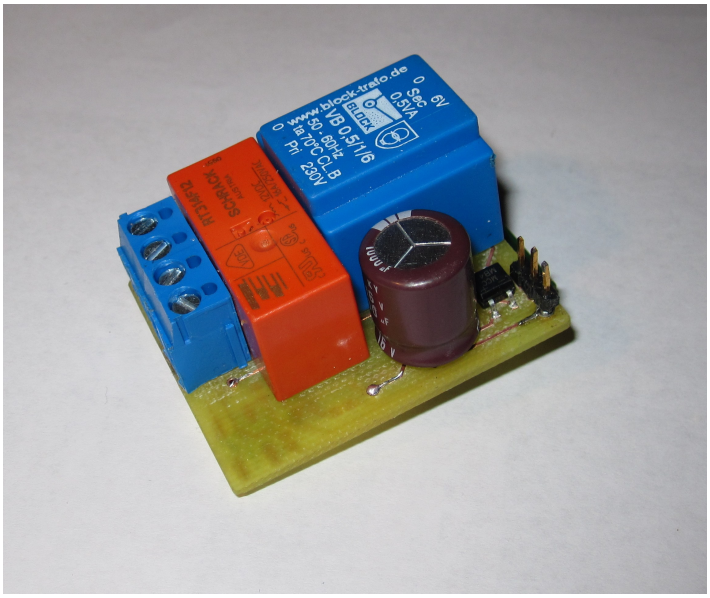
Programvara som er utvikla i denne oppgåva er delvis basert på, og gjort mogleg av eksisterande programvareløysingar. Desse blir kort gjennomgått i det følgande.

5.3.1. BitCloud

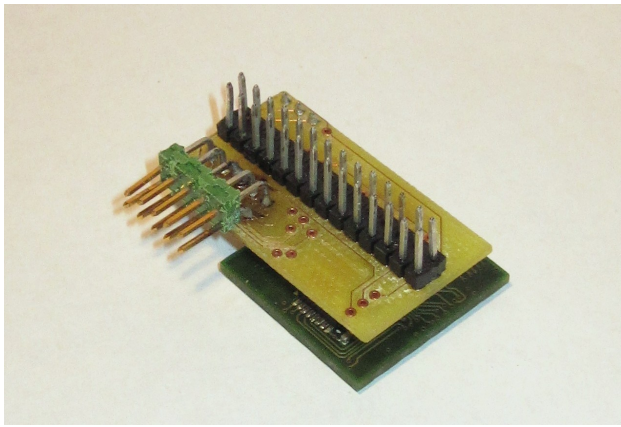
BitCloud er Atmel si programvareløysing for ZigBee-kommunikasjon i mikrokontrollarar. All programvare som er utvikla for mikrokontrollarar er bygd på denne løysinga.



Figur 5.4.: Framsida av lysbrytarnoden.



Figur 5.5.: Framsida av relénoden.



Figur 5.6.: Den eine sida av heimesentralnoden.

Ein del om oppbygginga av BitCloud er dekka tidlegare, og blir difor ikkje gjenteke her. [2] Ein skilnad er likevel at det i denne oppgåva vart nytta ein ny versjon i forhold til sist, 1.11 mot 1.10 i haust. Den nye versjonen gir mellom anna brukaren tilbake kontrollen over hovudfunksjonen i mikrokontrollarprogrammet, noko som ikkje var mogleg i 1.10.

Atmel har ingen fullstendig implementasjon av ZigBee Cluster Library, men noko dei kallar «Profile Suite». [55] Desse tilbyr truleg ein del av funksjonaliteten frå ZCL, men er dårleg dokumentert. Dette har difor ikkje blitt nytta i oppgåva.

5.3.2. Yocto-prosjektet

Det vart vurdert fleire alternative Linux-distribusjonar for bruk på Beagleboardet, men ein ende til slutt opp med å byggja ein eigen, tilpassa variant, ved hjelp av Yocto-prosjektet.

Yocto-prosjektet prøver å gjera det enklare for utviklarar å byggja sin eigen Linux-distribusjon, inkludert den verktøykjeda som skal til for å kompilera programvare og andre tilhøyrande hjelpemiddel for ein slik distribusjon. Yocto som prosjekt nyttar Beagleboard som si offisielle testplattform for ARM-arkitektur, noko som reduserer sjansen for å støyta på alvorlege problem.

Tradisjonelt har det å setja saman og kompilera ein Linux-distribusjon vore ein svært tidkrevjande og komplisert prosess, med store rom for feil. Yocto gjer dette enklare, delvis ved å implementera verktøy og oppskrifter, delvis ved å dokumentera metodar og delvis ved å vera eit nettsamfunn

der ein kan få hjelp. [56]

Yocto er eit slags paraplyprosjekt for Poky, som igjen nyttar verktøyet «Bitbake» frå OpenEmbedded-prosjektet. [57] Poky er byggjesystemet i Yocto, det som strukturerer byggjinga, medan Bitbake er sjølve verktøyet som hentar kjeldekode, legg til nødvendige patchar og kompilerer.

Bitbake er basert på oppskrifter. Ei oppskrift inneheld alt som skal til for å kompilera eit spesifikt program, til dømes kva programmet er avhengig av å ha, eventuelle spesielle tilpassingar, patchar, som skal leggjast til før kompilering og liknande.

Resultatet ein sit att med er hovudsakleg ei Linux-kjerne, eit filsystem som ein kan leggja på microSD-kortet som Beagleboardet startar frå og strukturen for ei pakkebrønn. For meir informasjon om prosessen med oppsett og bruk av Yocto og Poky, sjå vedlegg C.

5.3.3. Django

Ved å velja Linux som operativsystem for heimesentralen, får ein opnar det seg ei svært lang rekke alternative rammeverk for å byggja eit webgrensesnitt å velja blant. Fleire alternativ vart vurderte, men ein ende til slutt opp med å basera seg på Django. [58]

Django er eit webrammeverk skrive i Python, som tilbyr ei lang rekke funksjonar som vart sett på som nyttig i dette tilfellet: abstraksjon av databaseinnhald, same data presentert gjennom ulike malar, god støtte for internasjonalisering med meir. I tillegg er Python eit høgnivåspråk, som gjer det lett å utvikla i, til dømes ved å abstrahera vekk minnehandtering og tilby dynamiske lister og strengar.

5.3.4. D-Bus

D-Bus er ein kommunikasjonsbuss for å kommunisera mellom program under Linux. [59] D-Bus lar program kalla på metodar i andre program. I tillegg kan systemet nyttast til å distribuera signal, som andre program kan abonnera på. Det finst bindingar for D-Bus til ei rekke språk, inkludert Python, som blir nytta her.

Protokollen er mykje brukt i skrivebordsdistribusjonar, men òg på innebygde system, som til dømes mobiltelefonen Nokia N900. I denne oppgåva blir D-Bus nytta til å kommunisera mellom programvara som snakkar med ZigBee-kortet og andre prosessar som køyrer på Linux.

5.4. Nodeprogramvare

Programvara som køyrer på nodane er som nemnt skrive i C, basert på BitCloud.

5.4.1. Tilpassing av BitCloud

I prosjektoppgåva nytta ein BitCloud saman med nodar som er offisielt støtta, som utviklingplattformer. Dette gjeld sjølvsagt ikkje nodane som er utvikla i denne oppgåva, og det var difor nødvendig å gjera visse tilpassingar av BitCloud-stacken.

Frå Atmel si side er det lagt opp til å gjera dette arbeidet relativt enkelt, men filene må leggjast inn i sjølve BitCloud-stacken, i staden for å haldast saman med prosjektet ein arbeider med. Tilpassinga det er snakk om dreier seg om å definera kva pinnar lysdiodar, knappar og UART er tilgjengelege på, og laga relativt enkle initialiseringsfunksjonar. Dette blir svært maskinvareavhengig kode, og måtte difor gjerast individuelt for kvar av nodetypane som vart laga.

I tillegg viste det seg at BitCloud-stacken ikkje inneheldt funksjonalitet for å nytta alle dei generelle avbrotspinnane, dei såkalla `PCINT`-pinnane. Dette viste seg som eit problem, då ein hadde valt å kopla knappane på brytarane til desse pinnane. Ein måtte difor bruka ein del tid på å legga til støtte for avbrotshandtering for desse pinnane i BitCloud.

Eit anna alternativ ville vore å revidera kretsdesignen, men på grunn av at det hadde krevd å lodda komponentar av og på igjen, valde ein å heller løysa problemet i programvare i denne omgang.

5.4.2. Generelt

I delkapittel 5.1, blir «Home automation»-profilen nemnt. Denne dekker tilsynelatande mykje av det som skal utviklast i denne oppgåva. Denne profilen er likevel svært kompleks, og inneheld mykje som ikkje har interesse for denne oppgåva. Ein har heller ingen gode måtar å testa om ein faktisk implementasjon oppfyller standarden. Atmel har ikkje laga ein skikkeleg implementasjon av ZigBee Cluster Library, noko som er med på å gjera det å implementera denne profilen endå meir komplekst. Totalt sett valde ein difor å ikkje prøva å gjera ein full implementasjon av profilen.

Alle endepunkt i denne oppgåva ligg difor under ein eigen profil, men enkelte av dei støttar likevel kommandoane frå dei standardiserte klyngene. Til dømes støttar lysbrytarnoden «OnOff»-klynga som ei utgangsklynge, medan relénoden støttar «OnOff» som ei inngangsklynge. I BitCloud finst det ingen funksjonalitet for å implementera attributtane dei ulike

klyngene bruker, og det er difor ikkje gjort nokon forsøk på å støtta lesing eller skriving av attributtar.

5.4.3. Nettverk

I krav-delen er det hovudsakleg to krav som har med sjølve nettverket å gjera: kommunikasjonen skal skje kryptert, og nodar skal kunna skifta kva nett dei er tilkopla med eit knappetrykk.

På grunn av problem med konfigurasjonen, valde ein å ikkje implementera høgsikkerheitsmodusen ZigBee eigentleg skal støtta. Standard-nivå kryptering av nettverket var derimot relativt enkelt å aktivera, og kravde berre små endringar av BitCloud-innstillingane. Denne forma for sikkerheit gjer at ein ikkje kan forstå meldingar som blir sende, utan å kjenna nettverksnøkkelen. Denne nøkkelen blir automatisk generert av koordinatoren, når nettverket blir starta.

Eit punkt der dette er sårbart, er når ei ny eining skal bli med i nettverket. I slike tilfelle blir nøkkelen sendt ukryptert gjennom lufta. Ein eventuell angripar som lyttar då, vil dermed kunna få tak i nøkkelen, og deretter lytta på all trafikk.

I bruksscenariet er det skissert at ein kan nytta knappen på nodane til å tvinga noden til å finna eit nytt nettverk å kopla seg til. Dette er nyttig, om ein node til dømes koplar seg til naboen sitt nettverk i staden for sitt eige. Å implementera dette i praksis, viste seg dessverre vanskeleg. BitCloud har ingen funksjonalitet for å seia at ein ikkje vil kopla til eit spesifikt nettverk. I utgangspunktet koplar ein node seg til det sterkaste nettverket, og dermed kan ein bli fanga i ei løkke, der ein om og om igjen koplar seg til naboen sitt nett.

I delkapittel 7.2 blir begge problema som er nemnde her diskutert nærare.

5.4.4. Lysbrytarnoden

Lysbrytaren har primært to oppgåver i systemet: senda kommandoar til andre nodar, og kunna ta i mot ein konfigurasjon. Denne konfigurasjonen skal så lagrast i fast minne, og lastast inn igjen når noden startar.

Brytaren sine fire knappar er organisert i kvart sitt like endepunkt. Alle desse endepunkta blir likevel initialisert i same fil. Kvart av desse endepunkta støtter to klynger, «OnOff» som klient og «Setup» som tenar.

Når noden tek i mot ei melding gjennom oppsettsklynga, inneheld denne meldinga all informasjon som skal til for å oppretta ei binding til mottakaren. Å oppretta ei binding skjer i BitCloud ved å fylla inn ein spesiell datastruktur, og senda denne som ein parameter til ein spesiell funksjon.

Ved å lagra desse datastrukturane, kan ein dermed enkelt gjenoppretta ein konfigurasjon når noden startar igjen, ved å kalla på dei same funksjonane i BitCloud.

Når det blir trykka på ein knapp, vil desse bindingane bli nytta for å finna mottakar. Saman med bindinga, blir det òg lagra ein kommando som skal sendast, slik at ein til dømes kan velja om ein knapp alltid skal slå på, alltid slå av eller skifta tilstanden til utgangen.

For å implementera batterimåling, var som nemnt planen å bruka analog-til-digital-konverteraren (ADC) til å måla ei intern, fast spenning med drivspenninga til mikrokontrollaren som referanse. Dessverre viste det seg at denne planen ikkje let seg gjennomføra som tenkt. Databladet for mikrokontrollaren seier at ein kan nytta ein pinne kalla AVDD som referanse, og ein gjekk ut frå at denne ville vera lik drivspenninga VDD. Det er han ikkje. I staden er denne fast på 1,8 V, og metoden fungerer dermed ikkje.

5.4.5. Relénoden

Relénoden er den noden i systemet med enklast programvare. Noden har berre eitt endepunkt som er spesielt for denne, og dette endepunktet støttar ei klynge: «OnOff» som tenar.

Kommandoen frå meldinga som kjem inn blir lest, og utført. Sjølve relé-styringa er gjort i ein eigen modul, for å skilja nettverkskoden frå anna kode.

For å gjera testing enklare, valde ein å tilpassa reléprogramvara til å køyra på nodane som vart nytta i prosjektet òg. Desse har naturleg nok ikkje moglegheit til å styra faktiske relé, men funksjonaliteten blir simulert ved å slå av og på lysdiodar. Dette viste seg svært nyttig under testing.

5.4.6. Felles funksjonalitet

I tillegg til det ovannemnde, er det utvikla noko felles funksjonalitet. Til dømes har alle nodane tilgang på ein modul som gjer det mogleg å skriva ut data over UART. Denne modulen er henta frå prosjektet. Nodane har òg funksjonalitet for å blinka med eventuelle lysdiodar og ta i mot knappe-trykk.

Nettverksmessig tilbyr òg alle nodane i systemet ei id-teneste. Denne blir nytta av heimesentralen til å identifisera nodane. Id-tenesta er bygd opp slik at alle nodar i prinsippet kan vera både klient og tenar, men i bruk er det berre heimesentralen som er klient. Dei to andre nodetypane opptrer som tenarar. Denne funksjonaliteten er på mange måtar ei erstatning for den funksjonaliteten som ZigBee Cluster Library tilbyr, i forhold til å finna ut kva endepunkt, klynger og kommandoar ein node støttar.

Ein kjem nærare inn på bruken av denne tenesta i samband med heimesentralen.

5.5. Heimesentral

5.5.1. Operativsystem

Som nemnt vart det ved hjelp av Yocto laga ein eigen Linux-distribusjon for denne oppgåva. Yocto-prosjektet kjem med ferdige oppskrifter for alle grunnleggande komponentar, men i dette prosjektet måtte det lagast nokon oppskrifter i tillegg. Dette gjaldt mellom andre Django, SQLite og webtenaren som vart nytta, Cherokee.

I tillegg til oppskrifter for enkeltkomponentar, er det òg nødvendig å laga oppskrifter for systemet som heilskap. Dette blir ei form for metaoppskrifter, som ikkje seier noko om korleis pakkar skal byggast, berre kva pakkar ein vil ha med i det endelege Linux-biletet.

Alle oppskrifter som vart laga til denne oppgåva, vart lagt i det Yocto kaller eit lag. Eit lag er tenkt nettopp for å gruppera filer og oppskrifter som har med spesifikke prosjekt å gjera, nettopp som nødvendig her. Å konfigurera eit lag er heilt separat frå oppsettet av sjølve Yocto. Det einaste som må endrast i Yocto er å faktisk aktivera laget ein har sett opp.

Med Yocto kan ein enkelt la distribusjonen ein lagar nytta eit såkalla pakkesystem. Dette er ein måte å installera programvare, samt halda han oppdatert. Eit pakkesystem er bygd opp rundt pakkar, som oftast representerer eit program, eit bibliotek eller liknande.

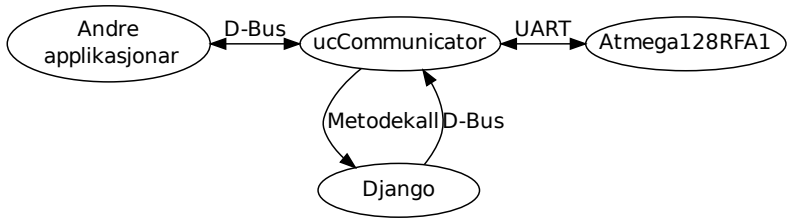
Ei oppskrift i Yocto-terminologi vil oftast laga ei pakke som ein innebygd del av bygginga. I tillegg lagar Yocto den informasjonen som skal til for å laga ei såkalla pakkebrønn. Ei pakkebrønn er ei kjelde for pakkar som kan installerast med pakkesystemet. Om ein vidareutviklar systemet, blir det difor svært enkelt å oppgradera programvara på heimesentralen.

Meir om konfigurasjon av Yocto, og korleis ein faktisk nyttar det Yocto gir ut på Beagleboardet, er skildra i vedlegg C.

5.5.2. Nodekomunikasjon

I kapittel 4.3.3 blir det grunngjeve kvifor det er ønskeleg å skilja kommunikasjon mellom mikrokontrollar og Linux ut i ein eigen prosess.

Som nemnt i kapittel 4.2, er både I^2C og UART maskinvaremessig tilgjengeleg for kommunikasjon. UART-linjene på Beagleboardet dukkar opp som ordinære serieportar, medan I^2C-grensesnittet er tilgjengeleg gjennom



Figur 5.7.: Strukturen i programvara på heimesentralen.

eit C-grensesnitt. Delvis på grunn av manglande støtte for I^2C mastermodus både i Linux og BitCloud, og delvis fordi UART viste seg enklare å ha med å gjera, valde ein å nytta UART til denne kommunikasjonen.

Sjølve programmet som les inn data vart kalla ucCommunicator og implementert i Python. Python har god støtte for å lesa data frå serieportar, og er eit svært behageleg språk å bruka. Strukturen i programmet er vist i figur 5.7, og blir gått nærare gjennom i det følgande.

Kommunikasjonen mellom noden og ucCommunicator er bygd på at kvar linje i UART-kommunikasjonen er ei melding, på følgande format:

```
MESSAGE_TYPE/NWK_ADDRESS/DATA_LENGTH/DATA0/.../DATAN
```

Her er `MESSAGE_TYPE` eit 16-bit tal, `NWK_ADDRESS` ein 16-bits nettverksadresse, til den eininga meldinga har med å gjera, `DATA_LENGTH` eit 16-bit tal som gir talet på `DATA`-element som følger. Meldinga er avslutta av linjeskift. Døme på meldingstypar kan vera at det har kome ei ny eining i nettverket, eller at heimesentralnoden har fått svar på ei id-spørjing.

Dei ulike meldingstypane er definert i kjeldekoda for sentralnoden. Ein del meldingstypar er i tillegg implementert som eigne objekt i Python, for å gjera behandlinga enklare. Eit slikt meldingsobjekt blir laga når programmet les inn ei melding frå UART.

Dei fleste meldingstypar har definert eit tilhøyrande D-Bus-signal som blir distribuert når meldinga kjem inn. Dette gjer det mogleg å laga eksterne program som kan motta data frå ZigBee-nettverket. Motsatt veg tilbyr programmet ei rekke D-Bus-metodar for å senda meldingar ut i nettverket.

I tillegg til å senda D-Bus-signal, er ucCommunicator òg integrert med databaseløysinga web-grensesnittet brukar. Dette blir gjort ved å nytta databaseabstraksjonslaget frå Django, som er relativt fleksibelt, og enkelt å nytta utan å trekka inn resten av Django. Dermed blir det automatisk oppretta ei ny eining i databasen når heimesentralen får melding om det.

Prinsipielt sett kan ein argumentera for at alt som har med Django å gjera burde ha køyrt i sin eigen prosess, men det vart slått saman for å

redusera talet på prosessar som trengst. Alt som har med Django å gjera er likevel skilt ut i sin eigen modul. Om ønskeleg, kan det dermed enkelt òg skiljast ut i ein eigen prosess.

5.5.3. Nodeidentifikasjon

For at webgrensesnittet skal kunna presentera dei riktige moglegheitene for dei ulike nodane, må ein vita kva type dei ulike nodane er. Om ein her skulle følga ZigBee Home Automation, seier denne som nemnt at dette skal skje ved å implementera ei teneste på nodane, som skal gi informasjon om type og tal på endepunkt, kva klynger dei støttar og så vidare.

Fordi dette er ei svært kompleks teneste å implementera, utan tilgang på eit skikkeleg klyngebibliotek, valde ein i denne oppgåva å nytta ei alternativ løysing. I staden for at nodane gir all informasjon, gir nodane berre to identifikasjonsnummer: ein produsent-id og eit produkt-id. Basert på dette, slår heimesentralen opp i sin database, og finn den nødvendige informasjonen der. Dermed kan web-grensesnittet presentera berre funksjonalitet som faktisk finst.

Foreløpig vil webgrensesnittet feila om ein node presenterer seg med ukjente id-ar. Moglege løysingar på dette, blir presentert i delkapittel 8.1.4.

5.5.4. Web-grensesnitt

Django kjem med sin eigen webtenar, som kan nyttast i utvikling. I tillegg har Django eit godt databaseabstraksjonslag, som abstraherer vekk sjølve databasebruken. Dette laget støttar mange databasetypar, men ein har valt å nytta SQLite i denne omgang. SQLitet er ei enkel, filbasert databaseløysing, som ikkje treng oppsett av noko slag. Dette gir eit svært fleksibelt utviklingsmiljø, der ein kan utvikla lokalt, utan å installera ekstra programvare utanom Django sjølv.

Seinare, i produksjon, kan ein enkelt skifta ut både web- og databaseløysing, utan å endra eigen kode. Særleg på webtenar-sida er dette nødvendig, då den innebygde webtenaren i Django på ingen måte er meint for anna enn utvikling. På sjølve Beagleboardet vart difor ein web-tenar kalla Cherokee nytta. [60] Denne er kjent for å krevja lite ressursar, og blir generelt sagt å vera enkel i bruk.

I Django-terminologi har ein eit prosjekt som toppnivå for alt arbeid, og dette kan innehalda fleire applikasjonar. Oppgåva her vart difor laga som eitt prosjekt, kalla «domatikaSuite», med ein applikasjon for styring og ein for oppsett, kalla «webcontrol» og «webconfig».

Django inneheld mykje funksjonalitet for å handtera databaseinnhald.

Det vart difor valt å laga ein databasestruktur som gjenspeglar ZigBee-strukturen. Dette gav strukturen vist i figur 5.8.

Kvar fysisk eining har då eit tilhøyrande objekt i databasen, ein Device. Det er `ucCommunicator` sitt ansvar å leggja inn einingane i databasen. Ei eining i databasen, er eit produkt, Product, tilhøyrande ein produsent, Manufacturer. Til eit produkt høyrer eit sett endepunkt, Endpoint, som har sine tilhøyrande klynger, Cluster, som igjen har sine tilhøyrande kommandoar, Command. To endepunkt kan knyttast saman med ei binding, Binding, som inkluderer ein kommando, Command.

Denne strukturen gjer det mogleg å gjera intelligente val og avgrensingar i web-grensesnittet. Dette viser seg til dømes i opprettinga av ei binding. Å gjera dette startar frå lista over einingar, der ein vel «Legg til binding». Dette valet er berre tilgjengeleg for inngangseiningar, og gir ein vegvisar som let brukaren velja kva eining denne inngangen skal knyttast til. Gjennom heile denne vegvisaren blir berre val som faktisk er gyldige vist fram, og ein unngår dermed at brukaren prøver å gjera noko som eigentleg er meiningslaust. Dermed heng alle val her nøye saman, og det er grunnen til at det er organisert som ein vegvisar.

Frå styringsgrensesnittet kan ein med denne strukturen for det første sørga for å visa berre einingar som kan styrast, og for det andre visa alle kommandoar som er gyldige for kvar av inngangsklyngene ei eining støttar.

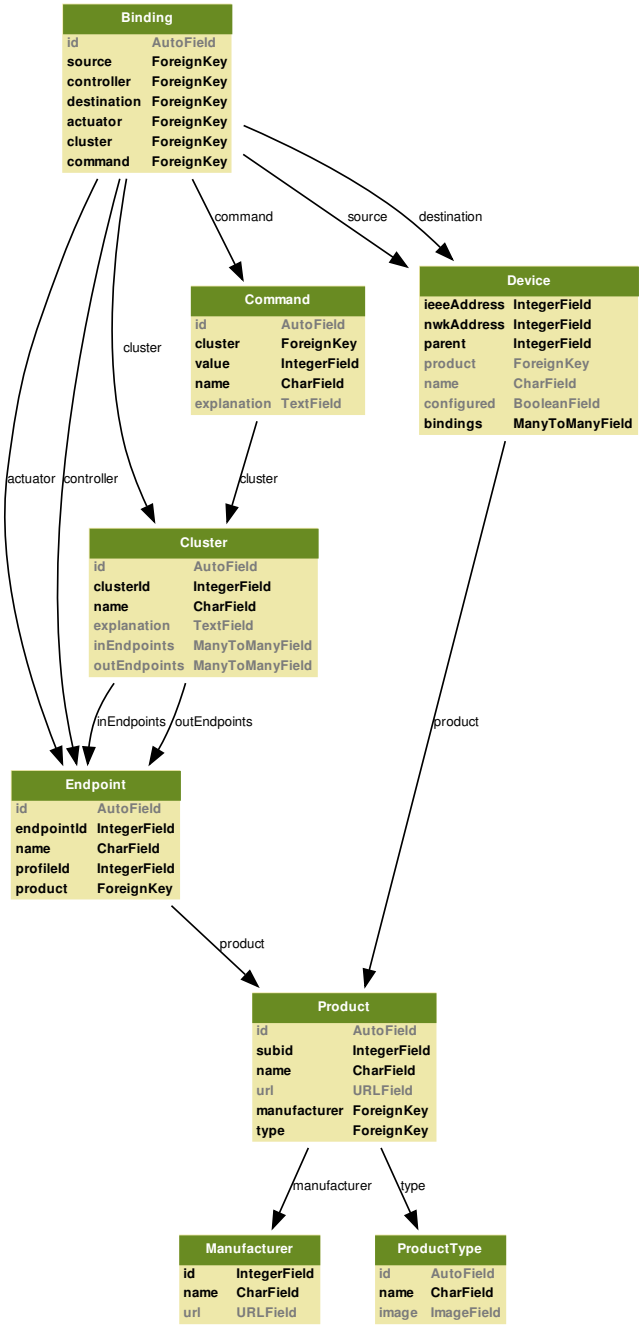
5.6. Deksel for lysbrytarnoden

Som nemnt i delkapittel 4.2.2, håpa ein å kunna nytta dekselet til ein vanleg Elko-brytar som deksel for lysbrytarnoden.

Då ein faktisk fekk tak i eit slikt deksel, viste det seg dessverre at dette var laga i svært tjukk plast. Dette gjorde det umogleg å trykka på knappane gjennom plasten. Ein prøvde difor å tilpassa dekselet, ved å fresa ut delar av plasten, men dette viste seg å vera svært vanskeleg. Resultatet vart heller ikkje tilfredsstillande, verken i form eller funksjon. Dekselet var framleis vanskeleg å trykka gjennom, og var heller ikkje så fint som ein kunne håpa.

Å ha eit deksel til brytaren vart i utgangspunktet sett på som svært lite viktig, og at tilpassinga av Elko-dekselet ikkje gjekk som tenkt vart difor ikkje teke særleg tungt. Etterkvart fekk ein likevel lyst til å prøva å laga eit deksel frå botnen av, då dette var ei fin avkopling frå programmering og skriving.

Etter litt tenking på alternative metodar, kom ein fram til at kryssfiner truleg var eit eigna materiale som ramme, kombinert med aluminium som sjølve dekselet.

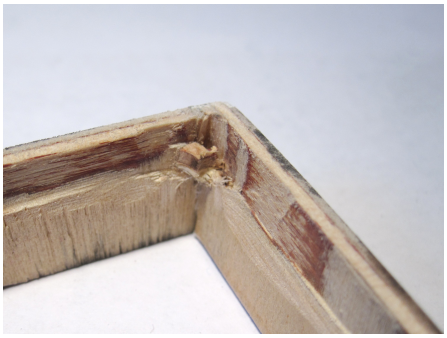


Figur 5.8.: Databasemodellen som vart nytta for Django-prosjektet.

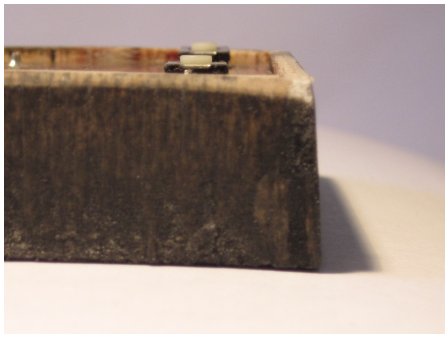
Rammen vart difor laga ved å laga fire kryssfinerbitar på om lag 59 mm · 17 mm, med spor i hjørnene og lima dei saman. På innsida vart det frest ut ein kant, slik at brytaren vart liggande noko forseinka inni rammen. Høgda vart tilpassa med små bitar som vart limt fast, slik at sjølve knappane stikk over ramma. Dette gjer det mogleg å trykka på han gjennom dekselet som blir lagt oppå.

Dekselet vart laga ved å klippa ut hjørner i ein aluminiumsplate på om lag 69 mm · 69 mm, slik at kvar langside kunne knekkast ned rundt hjørnet. Brytarnoden har ein lysdiode i midten, og der vart det difor frest ut eit hol i dekselet, som det vart limt ein pleksiglasbit i. Dermed kan ein sjå lysdioden gjennom dekselet.

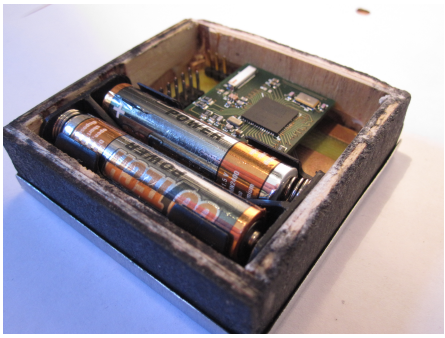
I figur 5.9 kan ein sjå det endelege resultatet, inkludert ein del detaljbilete. Ein fordel med den designen som er gjort her, er at han er relativt enkel, og såleis truleg eigna for masseproduksjon.



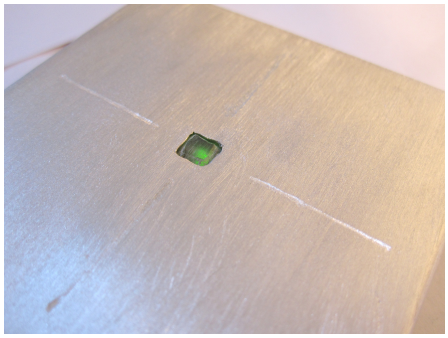
(a) Innvendig hjørne i rammen.



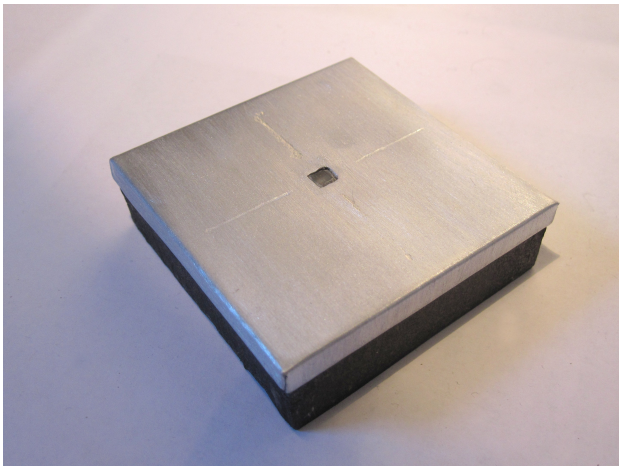
(b) Knappane stikk litt opp over rammen.



(c) Rammen gir god tilgang til batteria.



(d) Dioden er lett synleg.



(e) Den komplette brytaren.

Figur 5.9.: Rammen og dekselet som vart laga til lysbrytarnoden.

6. Testing

I dette kapitlet blir det prøvd å i størst mogleg grad testa dei resultata og implementasjonane som er gjort opp mot krava som vart laga i kapittel 3. Ein del av krava er ikkje testbare i form av målbare resultat, og ein startar difor testing med å visa nokon brukseksempel frå systemet.

6.1. Generell bruk

6.1.1. Oppstart

For å starta systemet, må ein først kopla eit ZigBee-kort til dei tre andre korta, lysbrytaren, relénoden og heimesentralen. Heimesentralkortet må òg koplast til Beagleboardet. Det er ingen fysiske hindringar for å setja ZigBee-korta feil veg. Dette ville vore mogleg å gjort for ZigBee-kortet, ved å forskyva dei to kontaktane i forhold til kvarandre, men vart oversett. Riktig veg for alle nodane er vist i figur 6.1.

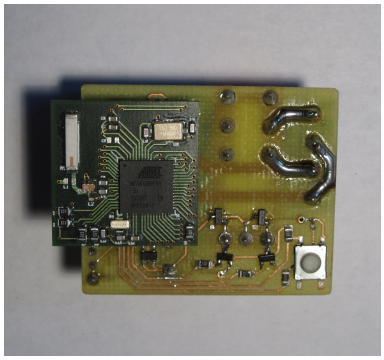
Beagleboardet har ingen av/på-brytar, og startar med ei gong det får straum. Dessverre er det tilsynelatande noko rart med samspelet mellom Beagleboardet og ZigBee-kortet, som gjer at ZigBee-kortet som oftast ikkje startar opp på riktig måte. I staden blir det hengande, utan å faktisk starta eit ZigBee-nettverk, og utan å vera tilgjengeleg over UART. Den einaste løysinga ein har funne er å gi kortet ein reset-puls, til dømes ved hjelp av JTAG-programmeraren.

For å kunna bruka web-grensesnittet til heimesentralen, må ein finna IP-en til denne. Foreløpig kan dette berre gjerast gjennom serieporten på Beagleboardet, noko som er nærare skildra i vedlegg C.3.3.

Når ein har IP-adressen, kan ein gå til denne i ein nettlesar for å få opp web-grensesnittet, som vist i figur 6.2. Her er det naturleg nok ingen einingar i lista, sidan systemet nettopp er starta.

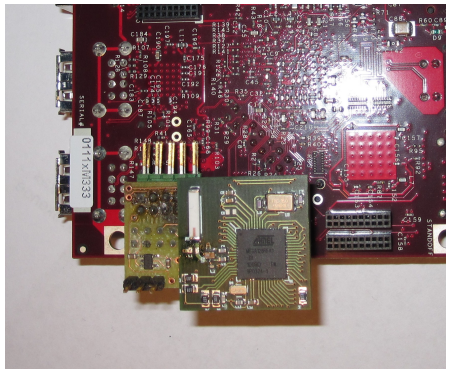
6.1.2. Nye einingar

Når nye einingar blir starta, sender dei ei melding til heimesentralen. Dette gjer at dei dukkar opp i lista, neste gong web-grensesnittet blir oppdatert.



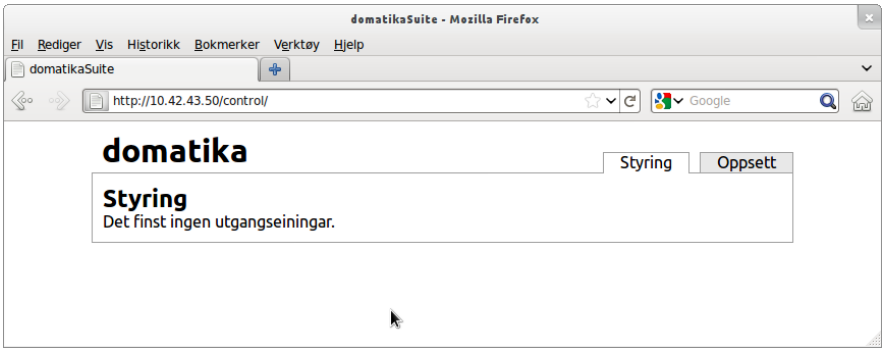
(a) Lysbrytarnoden.

(b) Relénoden.

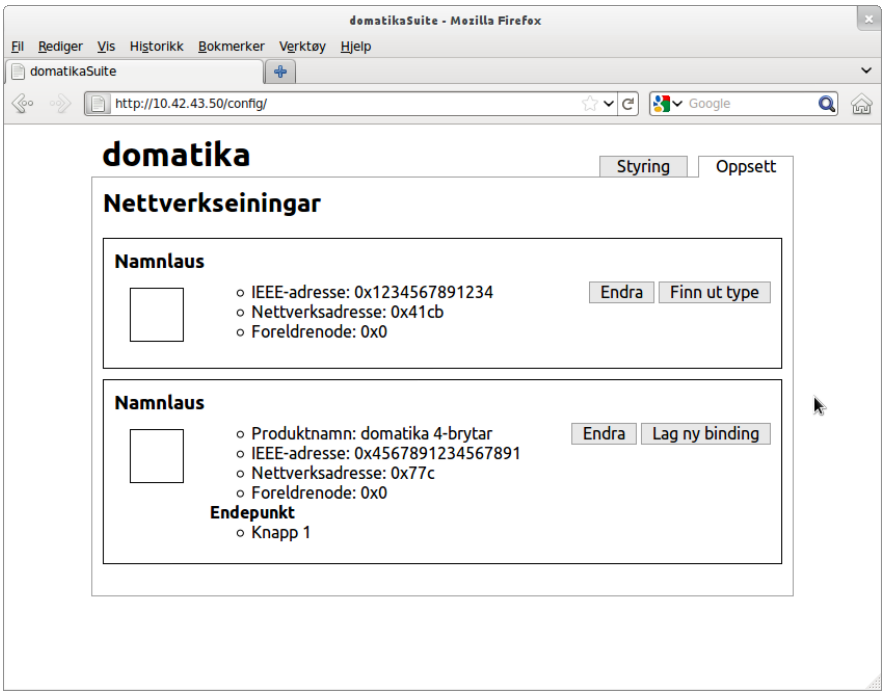


(c) Heimesentralen. Legg òg merke til kva veg sjølve heimesentralkortet står på Beagleboardet.

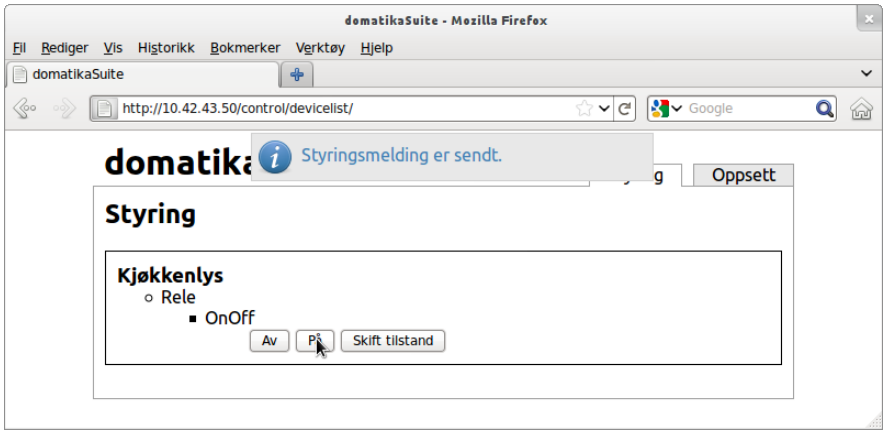
Figur 6.1.: Tilkopling av ZigBee-kortet til dei ulike andre korta.



Figur 6.2.: Web-grensesnittet umiddelbart etter oppstart.



Figur 6.3.: Web-grensesnittet etter oppstart av to nye nodar, ein som foreløpig er uidentifisert og ein brytarnode.



Figur 6.4.: Web-grensesnittet gir mogleghet til å styra utgangseiningar direkte, med alle kommandoar eininga støttar.

I utgangspunktet blir ein ny node lagt til som eit ukjent produkt. Ein må manuelt trykka på knappen «Finn ut type» i web-grensesnittet for å senda ein forespurnad om produkttype. Figur 6.3 viser web-grensesnittet med to aktive nodar, den øverste uidentifisert medan den nedste er identifisert som ein brytar.

6.1.3. Styring av einingar

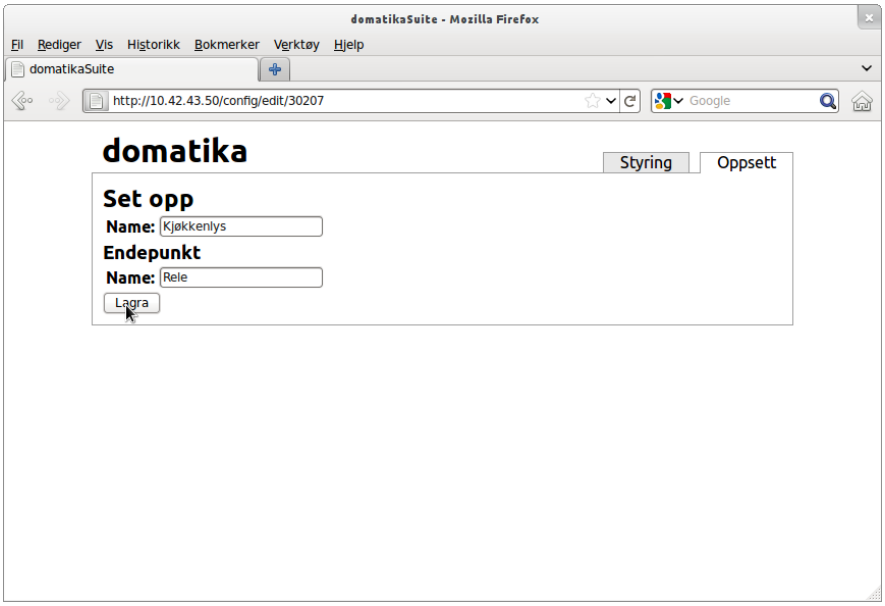
For utgangseiningar tilbyr styringfana direkte moglegheit til å senda kommandoane klynga støttar. Dette er vist for ein relénode i figur 6.4.

6.1.4. Oppsett av einingar

Kvar eining i web-grensesnittet har ein tilknytta knapp kalla «Endra». Denne gir foreløpig moglegheit for å gi ei eining eit namn, som vist i figur 6.5. Her er det òg aktuelt å implementera støtte for å velja kva rom ei eining høyrer til, eventuelt setja gruppe eller liknande.

I tillegg til «Endra»-knappen, har inngangseiningar ein knapp kalla «Lag ny binding». Dette valet gjer det mogleg å oppretta ei binding mellom ein inngang og ein utgang, som skildra i delkapittel 5.1.6. Bindingsprosessen er organisert som ein vegvisar, med steg som vist i figur 6.6.

Ein vegvisar vart nytta fordi valmoglegheitene i kvart steg i stor grad avheng av kva som er valt tidlegare. Det er lagt ned ein stor innsats i å



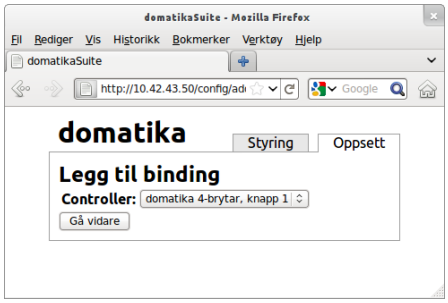
Figur 6.5.: Web-grensesnittet gir mogleghet til å endra namn på einingar.

presentera berre val som faktisk er gyldige, basert på kva klynger og endepunkt dei ulike einingane tilbyr. I mange tilfelle har ein difor berre eit val tilgjengeleg, til dømes når ein skal velja kva klynge ei binding skal opprettast gjennom. Dei nodane som er laga i denne oppgåva, lysbrytarnoden og relénoden, har berre ei klynge felles, «OnOff»-klynga. I desse tilfella burde systemet eigentleg ha gjort valet for brukaren, og ikkje spurt han i det heile. Dette er foreløpig ikkje implementert.

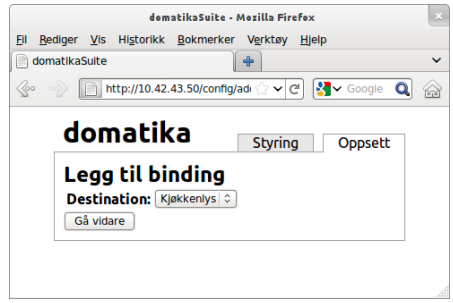
6.1.5. Bruk av lysbrytarnoden

Lysbrytarnoden er designa med fire knappar, som skal kunna styra einingar uavhengig av kvarandre. Dessverre viste det seg under testing at det hadde vorte gjort ein designfeil som gjorde at berre to av knappane faktisk er brukbare.

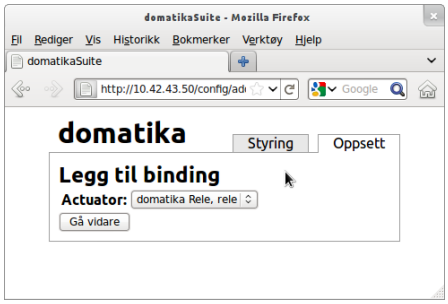
I design-fasen valde ein å kopla knappane til dei såkalla `PCINT`-pinnane på mikrokontrollaren. Tilsynelatande blir nokon av desse pinnane brukt internt i BitCloud-stacken, og å lesa tilstanden på desse fungerer difor ikkje som forventa. Resultatet er at den fysiske brytaren berre har to fungerande knappar. Dette problemet kan ein enkelt kome rundt ved å flytta knappane



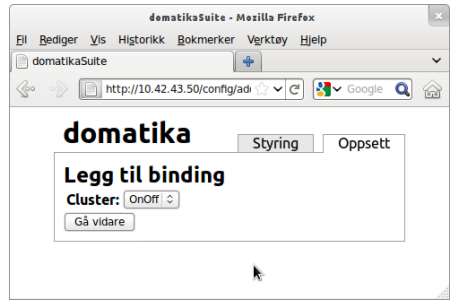
(a) Vel inngangskanal.



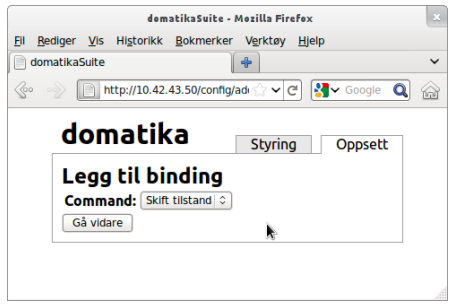
(b) Vel mottakar.



(c) Vel mottakar-kanal.



(d) Vel klynge.



(e) Vel kommando.

Figur 6.6.: Web-grensesnittet for å oppretta binding mellom einingar. Merk at ein i kvart steg berre får opp val som faktisk er gyldige, basert på endepunkt og klynger støtta av sendar og mottakar.

til andre pinnar på mikrokontrollaren.

6.2. Testar

6.2.1. Rekkevidd

Denne testinga vart gjort svært enkelt, ved å bruka lysbrytarnoden og heimesentralen. Lysbrytarnoden vart starta i ulike avstandar frå sentralen, og det vart vurdert kor langt vekke han kunne vera og framleis klara å knytta seg til nettverket.

Testen viste at den maksimale avstanden ein kunne ha, var om lag 1,5 meter.

6.2.2. Batterilevetid

For å gi eit godt estimat på batterilevetida, må ein vita straumforbruket. Resultata frå prosjektarbeidet viste at det utan tvil er straumbruk i sovande tilstand som har størst innverknad på batterilevetida. [2] Ein valde difor å måla dette spesifikt, ved å nytta eit multimeter.

Multimeteret vart kopla i serie med ei straumforsyning, slik at straumforbruket kunne lesast av i displayet. Oppsettet er vist i figur 6.7, med eit svært representativt resultat i displayet: 40,4 µA. Resultatet varierte noko, opp mot 40,9 µA i periodar, men låg for det meste på 40,4 µA.

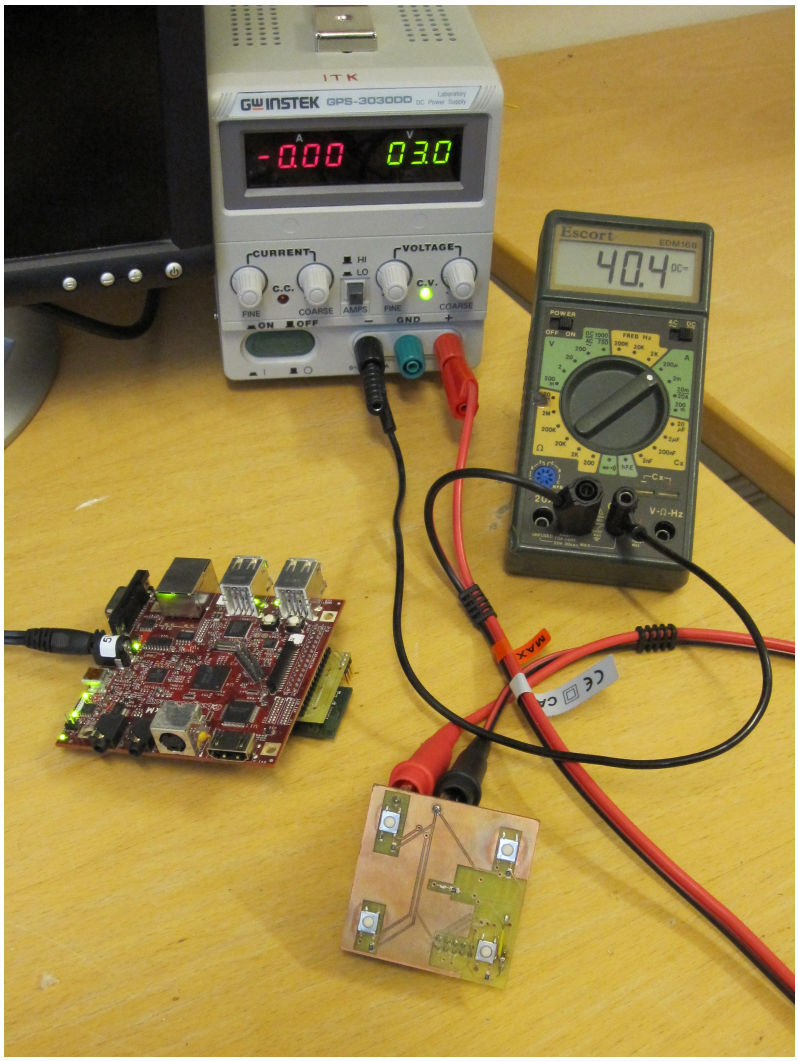
Ein ting som kan vera verdt å leggja merke til her, er at når lysdioden lyste, auka forbruket til 0,72 mA.

6.2.3. Responstid

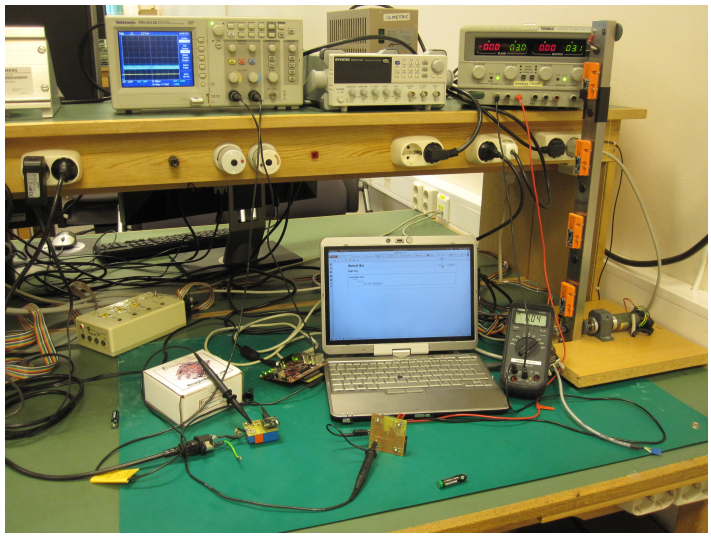
I denne testen var det nytta eit oppsett som vist i figur 6.8. Straumtrekket til brytaren vart kontinuerleg overvaka, for å sikra at han faktisk sov utanom knappetrykka. Som figuren viser, var det ikkje stor avstand mellom nodane, og det var heller ingen anna trafikk i ZigBee-nettverket under testen. Den bærbare datamaskinen i figuren hadde aktivert trådlaust nettverk, men det var ikkje i bruk under testen.

Knapp 1 på brytaren vart gjennom webgrensesnittet bunde til «Skift tilstand»-kommandoen reléet tilbyr. Eit oscilloskop med to kanalar vart nytta, med ein kanal tilkopla den fysiske knappen, og den andre kanalen kopla til styringa av reléet. Dermed kunne ein sjå på tida mellom dei to flankane.

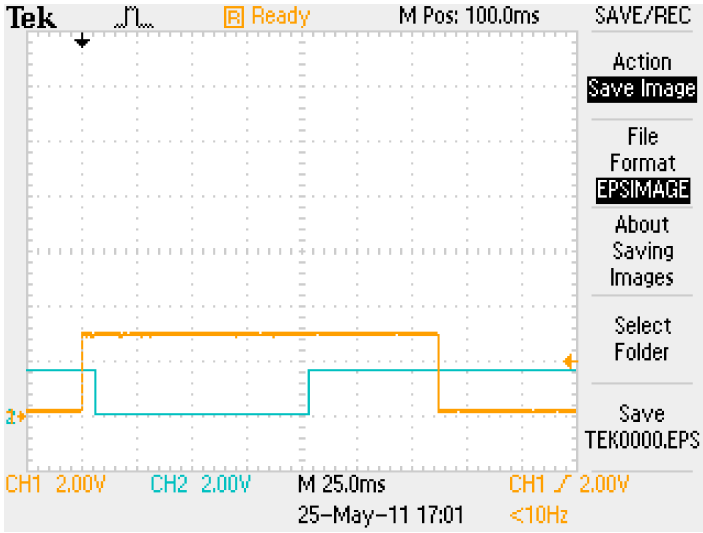
Det vart gjort fleire testar, men resultata var tilsynelatande konstante, og låg på i overkant av 5 ms. Berre ein figur er difor vist her, i figur 6.9.



Figur 6.7.: Oppsettet for testing av straumforbruk i sovemodus, og dermed batterilevetid. Multimeteret viser òg eit svært representativt resultat, 40,4 µA.



Figur 6.8.: Oppsett som vart brukt til å testa responstida. Øvst til venstre står oscilloskopet, som er flanketrigga på knappetrykket. Nedst til venstre er relénoden, framfor datamaskinen ligg brytarnoden.



Figur 6.9.: Resultat frå responstidtesten. Den gule linja er knappetrykket, den blå reléstyringa. Forseinkinga er i overkant av 5 ms, heile styringspulsen 100 ms og knappetrykket omtrent 175 ms.

7. Diskusjon

I dette kapitlet blir det sett nærare på område ein i ettertid har oppdaga at burde vore gjort annleis. I tillegg blir alternative bruksområde for eit nettverk som dette skissert.

7.1. Kretskortproduksjon

Denne oppgåva er det første store kretskortprosjektet underteikna har gjennomført, og dette har gjort at det har gått med meir tid på denne delen enn nødvendig. Fleire av korta som vart designa måtte gå gjennom fleire revisjonar for å fiksa feil som vart oppdaga undervegs. Truleg kunne dei fleste av desse vore unngått om ein hadde hatt noko meir erfaring i utgangspunktet.

Til dømes vart det i utgangspunktet nytta for små hol til via-ane. Dermed var det ikkje andre alternativ enn å laga korta på nytt. I og med at det å slå i via-ar er ein manuell operasjon, og ikkje noko kretskortfresen gjer, vart det òg talet på via-ar forsøkt redusert. Eit verkemiddel for å få til det, var å ikkje setja via i alle hol i dei ulike kontaktane, men i staden føra signala til baksida av kortet utanom kontakten. På den måten blir det god kontakt mellom pinne og bane i loddinga av kontakten.

Bruken av via-ar var likevel ikkje heilt uproblematisk. For nokon via-ar opplevde ein at det ikkje var tilstrekkeleg kontakt mellom sjølve via-en og banen som leia inn til via-en. Dermed fekk ein ikkje kontakt i det heile.

Dette gjaldt til dømes jordpinnen i UART-kontakten. Fordi UART som protokoll er definert med signal i forhold til jord, fekk ein dermed ikkje overført data i det heile. Med oscilloskop såg signala likevel fine ut, og det tok difor relativt lang tid å finna denne feilen. Løysinga er likevel enkel, og etter å ha lagt på litt tinn mellom via-en og banen, fungerte kommunikasjonen bra. Denne erfaringa bidrog til å løysa seinare problem raskare.

Vidare vart det for fleire av komponentane etterkvart oppdaga feil i symbol og fotavtrykk. Som nemnt i delkapittel 4.1, har ikkje gEDA-pakka særleg store bibliotek for symbol og fotavtrykk, og dei fleste vart laga sjølv. Feila i desse gjorde at komponentane anten ikkje passa fysisk, eller at dei ikkje fungerte som tiltenkt. I enkelte tilfelle vart dette fiksa ved å lodda på

ledningar på strategiske plassar, medan ein i andre tilfelle måtte laga nye kort.

Alt i alt gjorde manglande erfaring at arbeidet med kretskortdesign tok mykje lenger tid enn det ein hadde rekna med, noko som har redusert tida tilgjengeleg til dei andre delane av oppgåva.

7.2. Nettverk

Under implementasjonen vart det oppdaga problem med å oppfylla krava som går på nettverksfunksjonalitet, både i forhold til kryptering og i forhold til å bli med i riktig nettverk. På mange måtar kan desse problema seiast å henga saman.

Standard-sikkerheit i ZigBee er som nemnt sårbar om nettverket blir avlytta i det ein ny node blir med i nettverket, sidan nettverksnøkkelen då blir transportert ukryptert gjennom lufta. Denne nettverksnøkkelen blir aldri oppdatert, og blir nytta for all kryptering i nettverket.

I høgsikkerheitsmodus nyttar ZigBee ulike nøklar mellom alle nodar, i tillegg til at nettverksnøkkelen av og til blir skifta. Dette, saman med kryptert nøkkelutveksling gjer denne modusen eigentleg å føretrekka.

Det er likevel enkelte problem her. Å bli med i eit høgsikkerheits nettverk, krev at noden som blir med kjenner ein av nøklane som blir brukt i nettverket. Dette kan vera anten nettverksnøkkelen eller ein spesiell linknøkkel til eininga som er ansvarleg for sikkerheit, tillitssenteret.

I ZigBee Home Automation seier ein at alle einingar skal ha den same, førehandsprogrammerte link-nøkkelen, og nytta denne for å kommunisera med tillitssenteret for å få vita gjeldande nettverksnøkkel. Dette kunne fungert svært bra for å utelukka angriparar, om ein til dømes kunne seia at tillitssenteret ikkje tillet nye nodar å bli med utan at brukaren godkjenner det. Dessverre ser ikkje dette ut til å vera mogleg med BitCloud slik løysinga framstår i dag.

Heller ikkje problemet med å tvinga nodar til å bli med i eit nytt nettverk er enkel å løysa med dagens versjon av BitCloud. Ein kan spesifisera at ein ønskjer å bli med i eit spesielt nettverk, basert på nettverksnamnet, men ikkje at ein ikkje vil bli med i eit spesielt nettverk. I og med at det òg er slik at ein node blir med i det nettverket som har sterkast signal der noden er, kan ein godt risikera at noden blir med i same nettverk om og om igjen.

Truleg kan ein redusera dette problemet ved å flytta noden fysisk, men for til dømes relénodar, montert inni elektriske boksar på veggen, kan dette vera svært problematisk å få til praksis. Det er heller ikkje ein metode som garantert fungerer.

Ein måte å løysa dette på er difor å kunna fortelja noden på førehand kva nettverk han skal bli med i. I og med at nodane berre har knappar og lysdiodar som brukargrensesnitt, er det lite aktuelt å la brukaren gi slik informasjon på førehand. Eit alternativ er difor å kopla ein ny node fysisk til sentralen, før ein startar han. På den måten kan all interessant informasjon overførast, eventuelt òg inkludert nettverksnøklar eller liknande, og heimesentralen kan bli klar over at denne noden no kjem til å starta opp.

Ei slik løysing er likevel ikkje enkel å innføra, og vil kreva både programvare og maskinvare for å kunna gjennomførast. Det framstår likevel som ein interessant tanke.

7.3. Testar

Nokon av testane som vart gjort, krev ein viss diskusjon av resultata som vart oppnådd.

7.3.1. Rekkevidd

Som resultata i delkapittel 6.2.1 viser, har RF-korta som vart utvikla her svært dårleg rekkevidd. Når ein designar RF-kort, er det ønskeleg med maksimal effektoverføring til antenna. Dette blir gjort ved å sørga for at impedansen i antenna er lik impedansen i kretskortlinja frå mikrokontrollaren til antenna. Som kjent gir dette maksimal effektoverføring til antenna. [35]

For Atmega128RFA1, som nyttar ein balun, skal den differensielle impedansen i linjene frå mikrokontrollaren til balunen vera 100 ohm, og impedansen frå balunen til antenna vera 50 ohm. Dette vart det, som nemnt i delkapittel 4.2.1, rekna på før kortet vart designa, men resultata var altså ikkje gode.

I databladet for balunen, er det foreslått å nytta ein ekstra kondensator på mellom 0,5 og 1,5 pF for å betra denne tilpassinga. Det vart difor prøvd å lodda på ein kondensator på 1 pF, utan at dette gav nokon merkbar betring i rekkevidd.

Det kan vera fleire årsaker til det dårlege resultatet. I utrekninga av linjeimpedans, inngår, som likning 4.1 viser, fleire variablar som ein ikkje kan fastslå med sikkerheit, til dømes den elektriske permittiviteten til kretskortmaterialet og kor stor avstand det er ned til referanselaget. Dette er begge tal som er spesifisert frå kretskortleverandøren, men dei seier samtidig at dei kan variera frå produksjonsserie til produksjonsserie.

Det er mogleg å måla linjeimpedans, men dette krev visstnok svært dyrt utstyr og er ikkje trivielt. Då rekkevidda vart vurdert å vera god nok til

Aktivitet	Forbruk	Tal per dag	Sum per dag
Soving	40,5 µA/time	24 timar/dag	0,97 mAh/dag
Trykk	10,0 µAh/trykk	20 trykk/dag	0,20 mAh/dag
ZigBee	0,40 mAh/vakning	1440 vakningar/dag	0,58 mAh/dag
Sum			1,75 mAh/dag

Tabell 7.1.: Oversikt over straumforbruket for lysbrytarnoden per dag, kor mykje som går med til soving, brukartrykk og oppvakningane ZigBee krev.

testformål, vart det ikkje prioritert å gjera dette i denne omgang.

7.3.2. Batterilevetid

I samband med testing av straumforbruk, skal det nemnast at ein i utgangspunktet fekk svært dårlege resultat, omtrent 1,8 mA i sovande tilstand. Etter litt nøyare lesing av databladet, viste det seg at dette kunne koma av at avlusingsfunksjonaliteten på brikka var aktivert. [34] Denne blir styrt gjennom ein såkalla «fuse» ved namn OCDEN. Ein «fuse» er ei form for permanente innstillingar for mikrokontrollaren. Ved å deaktivera OCDEN, sank straumforbruket til det som er skissert i delkapittel 6.2.2.

Her vart det altså målt ei straumtrekk på om lag 40,5 µA i sovande tilstand. Basert på dette, får ein følgande sovetid på to 1200 mAh AAA-batteri som følgjer:

$$2400\ \text{mAh}/40{,}5\,\mu\text{A} \simeq 6,8\ \text{år} \tag{7.1}$$

ZigBee som teknologi krev at endeklientar tek kontakt med foreldrenoden sin ei gong av og til, for å få levert eventuelle meldingar til seg. I tillegg vil brytaren vakna og bruka ekstra straum kvar gong det blir trykka på knappen. I denne oppgåva vart det ikkje gjort målingar av kor mykje straum dette krev, men dersom ein nyttar verdiane frå prosjektet, kan ein setja opp tabell 7.1.

Denne viser altså at ein dag, med 20 knappetrykk og oppvakning ei gong i minuttet, gir eit totalt straumforbruk på 1,75 mAh. Dersom ein nyttar to 1200 mAh AAA-batteri som referanse, gir dette ei batterilevetid på nesten 3,8 år, og må seiast å vera eit ganske godt resultat.

$$2400\ \text{mAh}/1,75\ \text{mAh/dag} \simeq 3,8\ \text{år} \tag{7.2}$$

7.3.3. Responstid

Resultatet ein fekk her, er svært bra. Ei forseinking på 5 ms blir opplevd som at endringa skjer umiddelbart. Dette står i sterk kontrast til ein tilsvarande test i prosjektet, der forseinkinga vart målt til over 2 sekund. [2]

Det er vanskeleg å vita kva som gjer at ein har ei så stor endring i resultat, sidan testane som vart gjort er svært like. Begge vart gjort med nodane nær kvarandre, og begge vart gjort utan annan trafikk i ZigBee-nettverket eller i tilstøytande trådlause nettverk.

I testen frå prosjektet vart det nytta ein ferdig node, frå dresden elektronik. Denne er som tidlegare nemnt basert på separat mikrokontrollar og RF-brikke. I tillegg vart det i prosjektet nytta ein tidlegare versjon av BitCloud, 1.10 mot 1.11 her. Om ein eller begge av desse faktorane bidreg, er vanskeleg å seia.

Det er mogleg at ein ville fått andre resultat om ein hadde prøvd med eit ZigBee-nettverk med mykje trafikk i. På grunn av tidsaspektet vart dette ikkje testa i denne omgang.

7.4. Masseproduksjon

Oppgåveteksta seier at systemet skal vera basert på eit «lavpris trådløst nett», og at systemet som skal designast skal kunna masseproduserast. I forhold til masseproduksjon, er det interessant å sjå nærare på to aspekt: komponentkostnad og kortdesign.

7.4.1. Kostnadsanalyse

Dersom ein ser på komponentkostnaden for eit heilt system, basert på designen nøyaktig som skissert i denne oppgåva, endar ein opp med kostnadane vist i tabell 7.2[1]. Denne tabellen viser kostnaden for eit system med berre ein inngangsnode og ein utgangsnode, og gir dermed eit skeivt kostnadsbilete i forhold til eit system med fleire titals inngangar og utgangar.

Tabellen er likevel interessant som eit utgangspunkt, og viser til dømes at i eit lite system er Beagleboardet ein stor del av kostnaden. Som nemnt i kapittel 4.2 er det ikkje anbefalt å nytta Beagleboardet integrert i eit sluttbrukarprodukt. I tillegg inneheld kortet ein god del funksjonalitet som ikkje blir brukt i denne oppgåva, mellom anna skjermutgang, inn- og utgangar for lyd og ei rekke USB-portar. Ein kunne difor tenkt seg å på sikt utvikla ei eiga erstatning for Beagleboardet. Ein slik jobb har sjølvsagt ein

[1] Merk at det heller ikkje her er teke med kostnadar for sjølve produksjonen av kretskort.

Nodetype	Tal	Med kontaktar	Utan kontaktar
Heimesentral	1	7,46	1,74
Relénode	1	11,82	6,10
Brytarnode	1	7,57	1,86
ZigBee-kort	3	45,40	27,84
Delsum		72,24	37,53
Beagleboard	1	149,00	149,00
Sum		221,25	186,53

Tabell 7.2.: Komponentkostnad for eit heilt system, men med berre ein inngangsnode og ein utgangsnode, med og utan bruk av kontaktar mellom ZigBee-kort og funksjonskort. Tala er i amerikanske dollar, og er henta frå tabellane i kapittel 4.

ikkje ubetydeleg kostnad, men kan truleg redusera produksjonskostnadane ein del.

Nest etter Beagleboardet er det ZigBee-korta som representerer den største kostnaden. Dette kjem sjølvsagt delvis av at ein har fleire av dei, men biletet er det same om ein ser på kostnad for eitt av kvar av dei fire korta; ZigBee-kortet er dyrast.

Dersom ein ser nærare på tabell 4.1, ser ein at størstedelen av kostnaden for eit ZigBee-kort er mikrokontrollaren. Atmega128RFA1 er faktisk den dyraste enkeltkomponenten som er nytta i oppgåva, til over 6 amerikanske dollar per stykk, og er dermed ein stor del av årsaka til den høge kortkostnaden. Dessverre er denne kostnaden svært vanskeleg å koma unna, og kan neppe reduserast noko særleg.

Det som likevel kan endrast, og som er ein stor del av kostnaden for alle korta, er sjølve kontaktane som er nytta mellom ZigBee-korta og funksjonskorta. Desse er relativt sett svært dyre, og representerer meir enn 75 % av kostnaden for både heimesentralenkortet og brytarnoden.

Dersom ein kunne unngå å nytta desse, ville det, som det går fram av tabell 7.2, redusera kostnaden med nesten 40 %, om ein ser vekk frå Beagleboardet. I eit stort system, med mange nodar, er dette ei svært naturleg tilnærming, då ein berre treng ein heimesentral, uavhengig av talet på nodar. I neste delkapittel blir det gått nærare inn på korleis korta kan endrast for å gjera dette mogleg.

Tabell 7.2 viser òg at relénoden er ganske mykje dyrare enn dei to andre nodetypane. Dersom ein ser nærare på tala, gitt i tabell 4.3, blir det tydeleg

at dei to største kostnadane i denne noden er transformatoren og releét. Utvalet av bistabile relé, som kan slå av og på 16 A, er lite, og det er difor lite truleg at det er mogleg å spara noko særleg her. Derimot kan det vera mogleg å spara i forhold til transformatoren. For å avgrensa tidsbruken, vart det laga ei svært enkel, heilt tradisjonell straumforsyning i denne oppgåva. Vissnok finst det moglegheiter for å byggja slike forsyningar på andre måtar, og dette kan potensielt redusera både kostnad og straumforbruk.

I designa slik dei er gjort i denne oppgåva, er det òg teke med kostnadar for både programmeringskontakt og UART-kontakt. Ingen av desse grensesnitta vil vera nødvendige i ein endeleg design, og fjerninga av desse kontaktane vil difor redusera kostnaden med ytterlegare nokon prosent.

7.4.2. Kortdesign

Delvis basert på kostnadsanalysen ovanfor, og delvis av andre årsaker, er det visse aspekt ved kortdesignen som bør endrast for å vera veleigna for masseproduksjon. Dette går delvis på det å unngå kontaktane som er nytta mellom ZigBee-kortet og funksjonskorta, og delvis på å gjera designen sikrare i bruk.

Å fjerna kontaktane mellom ZigBee-kortet og funksjonskorta har ikkje berre økonomiske fordelar. Slike kontaktar kan òg gi mekaniske problem, i det at dei kan løsna og gi brudd på signallinjer. Dessutan vil ein del av grunngjevinga for å skiljet som vart gjort, i forhold til fleksibilitet, vera unødvendig i eit masseproduksjonsscenario.

For å unngå kontaktane må ZigBee-kommunikasjonen dermed tilpassast til det kvar enkelt nodetype treng. Til dømes kan ein dermed tenka seg eit brytarkort, med eit eige, avsett område til mikrokontrollaren og tilhøyrande komponentar, men der kortet òg inneheld alle nødvendige knappar, lysdiode og batterihaldar.

For relénoden er det derimot truleg ønskeleg å halda på skiljet mellom ZigBee-kort og funksjon, for å skapa eit tydeleg skilje mellom høgspenning (230 V) og lågspenning. Kostnadsreduksjon er likevel mogleg, fordi ein ikkje lenger treng å fokusera på å ha mest mogleg funksjonalitet tilgjengeleg i kontaktane. Dette gjer at ein kan redusera talet på pinnar i kontaktane.

Det å halda på skiljet mellom ZigBee-kort og funksjonskort, gjer det òg mogleg å kapsla inn høgspenningskomponentane i forhold til brukargrensesnittet, representert av knappen og lysdioden. I den designen som er presentert i denne oppgåva, ligg knappen berre nokon få millimeter frå pinnar med 230 V, noko som truleg ikkje er i samsvar med gjeldande regelverk. I staden kan ein tenka seg eit kort med ZigBee-kommunikasjon,

knapp og lysdiode, som blir kopla til eit anna kort med sjølve straumforsyningsdelen, releét og temperaturfølarar med berre seks leiarar; to straumleiarar, to reléstyringsleiarar og to temperaturfølarleiarar.

Eit anna aspekt som gjer det nødvendig å gjera om på noko av kortdesignen, er eit eventuelt skifte i heimesentralplattform. Det heimesentralkortet som her er designa, er tilpassa til utvidingskontakten på Beagleboardet. Ved å nytta eit anna kort enn Beagleboard, må dette naturlegvis tilpassast det nye kortet.

7.5. Andre bruksområde

Hovudtyngda av denne oppgåva fokuserer på systemet som ei plattform brukt i styring av lys og varme. Oppgåveteksten inneheld likevel òg eit krav om å demonstrera at plattforma er eigna for andre bruksområde. Her er det sjølvsagt mykje å velja i, og ein har her valt å gå nærare inn på to av dei.

7.5.1. Adgangskontroll

Adgangskontroll er eitt av områda som blir nemnt i oppgåveteksta. Ein tolkar det her som å avgrensa talet på personar som har tilgang til eit spesielt område, til dømes som erstatning for dørlåsar i eit vanleg hus.

Ein dørlås er på mange måtar eit svært enkelt konsept: Har du nøkkel, kjem du inn, utan nøkkel, blir du utanfor. Ein kunne tenkt seg eit tilsvarande system, basert på plattforma i denne oppgåva. I så fall kunne ein hatt ei batteridriven eining, som inneheldt ein nøkkel, som når han vart kommunisert til låsen, gjorde at døra opna seg.

Den einaste fordelen med ei slik løysing, er likevel berre den trådlause funksjonen, ein slepp å putta nøkkelen i låsen og vri om. Du må framleis ha med deg ein dings, og om du mister han, kan nokon som finn han likevel koma seg inn. I tillegg kan ein få problem med batterilevetid. Det er difor interessant å gjera noko meir intelligent.

Ein kunne då til dømes tenka seg at ein kombinerer ei slik eining med eit passord eller liknande. Dermed unngår ein problemet med at ei mista eining automatisk gir tilgang, og i tillegg er det her kommunikasjonsplattforma blir interessant. Med ei plattform som skissert her, kan sjølve godkjenning av dette passordet gjerast på sentralen, og ein oppnår dermed enklare endenodar. I tillegg kan ein då få same passord og funksjonalitet i fleire dørlåsar, til dømes ei kjellerdør.

Sjølvsagt kan ein òg tenka seg meir avanserte samansettingar. Ei mogleg utviding kunne vera moglegheiten for å overstyra dørlåsen, slik at bruka-

ren til dømes frå arbeidsplassen sin kan sleppa inn nokon som står utanfor. Ei slik løysing krev truleg at ein kan sjå kven som er der, og dermed kanskje eit kamera. Eit kamera bør truleg nytta ein annan kommunikasjonskanal for overføring av sjølve biletet, men kan lata seg styra gjennom systemet det her er snakk om.

7.5.2. Lyd og bilete

Styring av lyd og bilete i heimar er prega av ei lang rekke fjernkontrollar, som kvar styrer si eiga eining. Dei fleste av desse er basert på infrarød stråling (IR), og krev dermed at ein har fri sikt til det ein skal styra.

Ved å til dømes laga ei bru mellom ZigBee og IR, kan ein få moglegheiten til å styra slike einingar frå andre stader enn der eininga faktisk er. Dette kan gjera det mogleg å gøyma vekk einingar som ein kanskje helst ikkje vil ha framme.

På lenger sikt, kan ein sjølvsagt òg sjå for seg einingar som i utgangspunktet er styrbare over ZigBee, til dømes gjennom profilen ZigBee Remote Control. [61]

Uansett er noko form for styring av lyd- og bileteiningar eit krav for å kunna verkeleggjera dei tyngste heimekinoløysingane enkelte nok drøymer om: Eitt trykk på ein knapp, for å slå av lyset, på prosjektoren og stereoanlegget, og trekka føre gardina.

8. Vidare arbeid

Ein kan sjå for seg svært mange ulike retningar ein kan ta eit prosjekt som skissert her vidare. I det følgande kapitlet blir nokon av desse skissert.

8.1. Forbetringar

Sjølv om det er lagt ned mykje arbeid i systemet som det er, er det utan tvil rom for forbetringar.

8.1.1. Fjernstyring

Systemet slik det er no er berre i teorien mogleg å fjernstyra. I prinsippet er det ingenting i vegen for å opna web-grensesnittet mot internett, og på den måten få fjernstyring. Dette er likevel ikkje på nokon måte å anbefala, i og med at grensesnittet ikkje implementerer nokon form for autentisiering.

Det er difor stort rom for vidareutvikling her. Det finst minst to måtar å gjera dette mogleg på. Eitt alternativ kan ein vera å gjera webgrensesnittet som her er utvikla meir robust, ved å implementera ei form for autentisiering. Ein bør sørga for at ei slik utvikling ikkje gjer lokal styring vanskelegare, anten ved å la alle lokale einingar nytta grensesnittet utan, eller ved å implementera ei form for kvitelisting av kjente einingar.

Eit anna alternativ er å laga ei sentralisert løysing for styring. Dette kan til dømes gjerast ved å laga ei abonnementsteneste, køyrande på ein felles tenar. All tilgang til systemet kan då skje gjennom denne, og ein slepp noko oppsett av lokalnettverket hos brukaren.

8.1.2. Programvareoppgradering

Programvareoppgradering av heimesentralen i systemet er relativt enkelt, sidan denne nyttar pakkesystemet skildra i delkapittel 5.5.1. Dermed kan ein ganske enkelt publisera ein ny pakke for å oppgradera programvara på denne.

For dei enkelte nodane i systemet finst det i denne oppgåva derimot ikkje nokon måte å oppgradera programvara, utan fysisk tilkopling gjen-

nom JTAG. For eventuell masseproduksjon kan dette visa seg å vera eit problem, om det til dømes blir oppdaga feil etter produksjon.

BitCloud-stacken, som er nytta i denne oppgåva, inneheld visstnok funksjonalitet som gjer det mogleg å gjera såkalla «Over the air»-oppgraderingar av programvara på nodane. Dette vil seia å senda ei programveoppgradering til noden trådlaust, som noden så skriv til minne og nyttar etterpå. Dette krev noko ekstra maskinvare og programvare, men er interessant for å kunna tilby feilrettingar til sluttbrukarar, utan at dei har eller kan bruka ein JTAG-programmerar.

8.1.3. Standardisering

For å gjera det mogleg å nytta systemet saman med andre produsentar sine einingar, kan det vera interessant å implementera heile eller i alle fall større delar av ZigBee Home Automation-profilen. Som nemnt i delkapittel 5.4, vart dette ikkje gjort i denne omgang, primært på grunn av tidsaspektet. Ei slik løysing vil gjera det mogleg å nytta heimesentralen som skissert her til å setja opp og styra andre produsentar sine einingar.

Som ei slags standardisering, kunne det òg vera svært interessant å sjå på korleis ein del av forskinga som er referert i kapittel 2.2 kunne integrerast. Det mest interessante konkrete dømet, er kanskje Smart-M3, frå DIEM-prosjektet. Som tidlegare nemnt er dette meint som ein generell informasjonsdatabase, noko som utan tvil kan ha stor nytteverdi i eit system som det her er snakk om.

Òg enkelte av dei andre prosjekta har aspekt som det kunne vore verdt å sjå nærare på. Dette gjeld til dømes OPEN meter sin kommunikasjon med leverandørar, eller eDIANA sine forsøk på å hjelp folk å redusera straumutgiftene sine.

8.1.4. Nodeidentifikasjon

Som nemnt i delkapittel 5.5.3, valde ein å basera seg på eigne produsent- og produkt-identifikasjonsnummer for å vita kva funksjonalitet ein viss node tilbyr. Dette er ei løysing som fungerer bra i seg sjølv, men det må gjerast eit arbeid for å støtta ukjente nodar. Her kan ein tenka seg ulike alternativ.

Eitt av dei er å gjera som i føregåande avsnitt, og implementera dei relevante delane av ZigBee Home Automation-profilen. Dette er truleg nødvendig for å kunna bruka andre produsentar sine einingar på ein fornuftig måte.

Eit anna alternativ er å laga ein eigen web-teneste, som inneheld informasjon om alle einingar som kan nyttast med systemet. Når ein heime-

sentral då får eit ukjent id-par som svar frå ein node, kan han spørja webtenesta etter informasjon. Ei slik teneste er nok enklare å utvikla enn å ta inn profilen, men vil gjera det vanskelegare å nytta saman med andre einingar.

8.1.5. Straummåling

Som det er nemnt tidlegare, finst det komponentar som kan nyttast for å gjera straummåling på ein veldig enkel måte. Dette vart likevel ikkje gjort i denne omgang, dels av storleiksomsyn og dels av prisomsyn.

Begge desse omsyna er ikkje nødvendigvis konstante, og det kan godt henda det kan gjerast meir intelligente ting for å få til denne funksjonaliteten. Kan henda kan ein laga nodane på ein slik måte at ein enkelt kan produsera nokon nodar med måling og nokon utan. Dermed kan ein for kvar utgang velja om ein vil kunna måla straumbruk eller ikkje.

Eit slikt konsept er nok mest interessant om alle utgangar faktisk er med i ei måling, og då kan ein tenka seg å laga ein heilt eigen, liten nodetype for straummåling. På den måten kan ein til dømes sjå forbruket skilt på til dømes rom eller kategori, som lys, varme, varmtvatn eller liknande.

Uansett er dette funksjonalitet som er svært nyttig, og som er naturleg å ha med i eit system som det her er snakk om.

8.1.6. Nettverksoppsett

Som nemnt i delkapittel 7.2, finst det utfordringar i forhold til oppsett av nettverket, kryptering og oppstart av nye einingar. Her er det absolutt rom for vidare arbeid, først for å finna ei god løysing på dei omtalte problema, og deretter faktisk implementera denne løysinga.

8.1.7. Batterimåling

For lysbrytarnoden, som er batteridriven, er det svært nyttig å kunna lesa av batterinivået frå heimesentralen. Dette var eigentleg planlagt implementert, men på grunn av feiltolking av databladet, vart det ikkje mogleg å implementera som planlagt.

Den enklaste måten å gjera dette mogleg på, er truleg å gjera batterispenninga tilgjengeleg som referanse for analog-til-digital-konverteraren. Dette kan gjerast gjennom pinnen på mikrokontrollaren kalla AREF.

8.1.8. Maskinvare

Som det går fram av kapittel 7.3.1, bør rekkevidda i systemet betrast mykje. Dette krev truleg bruk av dyrt og komplekst utstyr, men vil forhåpentleg gjera at ein får resultat meir i retning av det som vart sett i arbeidet med prosjektoppgåva. Om prosjektet skal bli praktisk brukbart, er dette heilt nødvendig.

I tillegg vart det nytta pinnar som burde vore ubrukte i designen av brytarnoden. Dette gjorde at ein måtte gjera ekstra tilpassing av stacken, forklart i delkapittel 5.4.1, i tillegg til at to av knappane ikkje fungerte som tiltenkt, nemnt i delkapittel 6.1.5. Knappane bør difor flyttast til pinnar som ikkje er i bruk av BitCloud, og som det er betre innebygd funksjonalitet for i BitCloud. Eit godt val her, er truleg `INT`-pinnane, som BitCloud har god funksjonalitet for å handtera.

8.1.9. Programvare

I krav- og design-kapitla er det snakk om å styra einingar i grupper, eller til eit bestemt scenario. På grunn av tidsaspektet vart ikkje dette faktisk implementert, men det er forhåpentleg laga eit godt grunnlag for å gjera dette som vidare arbeid.

I bruksscenariet er det òg skissert at systemet kan henta ned ekstra programvare for å nytta andre produsentar sine maskinvareeiningar. Ingenting av programvare som trengst for å få til dette er implementert, verken på sentralen eller på tenarsida.

8.2. Produktutvikling

Om ein skal kommersialisera systemet som er designa her, er mange av forbetringane nemnt ovanfore potensielt svært nyttige å fiksa. I tillegg dukkar det opp nokon problemstillingar som ikkje har verdi for eit laboratoriesystem, men som er essensielle for eit system som faktisk kjem i kontakt med sluttbrukarar.

8.2.1. CE-merking

Alt elektrisk utstyr som skal nyttast i Noreg, skal vera CE-merka. Dette er i utgangspunktet ei merking som seier at produsenten har følgt dei reglane som gjeld for utstyret det er snakk om. For å setja dette merket på eit produkt, må det lagast ei såkalla samsvarserklæring, der ein som

produsent erklærer at produktet følgjer gjeldande regelverk. Dette må òg kunna dokumenterast, til dømes med ein testrapport. [62]

Gjennom arbeidet med denne oppgåva har ein prøvd å i størst mogleg grad følgja dei regelverka ein veit om. Likevel er det sjølvsagt mogleg at ting har blitt oversett, og at einingane difor ikkje er i samsvar med regelverket. I tillegg har den designen som er nytta for relénoden neppe godt nok skilje mellom 230 V-sida og brukargrensesnittet. Til saman gjer dette at eventuell kommersialisering bør inkludera ein nærare kikk på kva krav som er gjeldande for dei ulike einingane og korleis einingane er i forhold til krava.

8.2.2. Nye nodetypar

I denne oppgåva har ein primært fokusert på systemet som ei plattform for meir intelligent styring av lys og varme. Dette er eit område som gir rom for ei lang rekke fleire einingar. Dei kanskje mest tydelege einingstypane som manglar i systemet er ein eller anna form for termostat, i tillegg til ein dimmbar utgang, primært for lys.

Men systemet stoppar på ingen måte der. I bruksscenariet blir det skissert å nytta systemet som vekkarklokke. I så fall kan ein kanskje tenka seg ein vekkarklokkenode, som liknar på tradisjonelle vekkarklokker, kanskje med ein høgtalar og enkel tidsinnstilling.

Liknande kan ein tenka seg ei eining som heng ved utgangsdøra, og som let deg fortelja systemet når du kjem tilbake. Slik informasjon kan til dømes nyttast til å slå ned varmen medan ein er ute, utan at det går ut over komforten når ein kjem tilbake.

I samband med dette er det kanskje mogleg å laga oppvarmingssystem som gjer intelligente val i forhold til når varmen blir slått på. I dag er dei fleste system tilsynelatande basert på tid. Truleg kunne ein gjort meir energieffektiv oppvarming om ein såg på kor fort temperaturen endrar seg når ein slår på, og justerer starttidspunkt basert på det.

Elles kan ein sjå for seg ei form for sensorar, som kan merka når eit vindauge går opp, og som då slår av alle omnar i rommet. Slike sensorar kan òg fortelja deg om du går ut av huset utan å lukka alle vindauge.

Ein kan òg tenka seg einingar som merkar om det er folk til stades i eit rom, helst utan at brukaren treng å ha med seg spesielle einingar. Andre styringseiningar er sjølvsagt òg mogleg, anten som reine visningar av webgrensesnittet frå heimesentralen, eller meir spesialiserte einingar med knappar.

8.2.3. Andre bruksområde

Eit nettverk som dette gir moglegheiter for å implementera mange ulike konsept. Oppgåveteksten seier det skal lagast ei plattform som skal kunna vera eit grunnlag for fleire bruksområde, og skisserer nokon: «lys og varme, energioptimalisering, adgangskontroll». I delkapittel 7.5 blir bruk innan adgangskontroll og styring av lyd og bilete gått noko nærare inn på.

Lista som er nytta i oppgåveteksten er sjølvsagt ikkje ei uttømande liste. Ein kan òg tenka seg å nytta plattforma til å styra andre ting. Det er vanskeleg å sjå for seg akkurat kor grensene går, men innan kvart av områda er det i alle fall rom for mykje arbeid, både i forhold til å utvikla fornuftige brukargrensesnitt og å finna gode nodetypar.

9. Konklusjon

Gjennom denne oppgåva har det blitt laga eit system for styring av tekniske installasjonar, basert på ZigBee som kommunikasjonsteknologi. Systemet er laga slik at det òg kan nyttast som ei plattform for andre applikasjonar seinare.

Arbeidet vart gjort på bakgrunn av eit studie av både eksisterande, kommersielle system og aktuelle forskingsprosjekt innanfor området. Forskingsaktiviteten er stor, og det er relativt stort spenn i prosjekta som er omtalt, frå reine tekniske prosjekt, til prosjekt med fokus meir på dei sosiale konsekvensane av meir intelligente heimar.

Basert på delar av dei eksisterande systema, forskingsresultat og eigne idear, vart det laga eit bruksscenario som har fungert som ein referanse i det vidare arbeidet. Dette scenariet skildrar korleis eit slikt system kan fungera, både i seg sjølv og som ei plattform for andre applikasjonar.

Som grunnleggjande konsept valde ein å laga systemet desentralisert, i det at konfigurasjonen ligg lagra i dei enkelte einingane i systemet. Likevel har systemet ein sentral, for å kunna gjera automatisert, sentral styring. Sentralen skal òg tilby styring og konfigurasjon gjennom eit webgrensesnitt.

Av fysiske einingar vart det i denne omgang laga ein sentral, ein brytarnode og ein relénode. Desse kan saman nyttast til å slå av og på alle typar elektriske einingar, opp til 16 A ved 230 V. Ein valde å skilja alle komponentar som har med ZigBee-kommunikasjon ut på eit eige kort. Alle dei tre andre nodane nyttar difor same type kort her. Å ha dette som eit eige kort, gir betre fleksibilitet, og gjer det enklare å nytta dette kortet i andre prosjekt.

Tidlegare arbeid har gjort ein kjent med Atmel sine komponentar, og ein valde å nytta desse òg no, i form av Atmega128RFA1. Dette er ein mikrokontrollar med innebygd kommunikasjonsbrikke.

Heimesentralen vart i tillegg til Atmel-brikka bygd rundt Beagleboard -xM, ein kraftig ARM-basert enkelt-kort-datamaskin, med nettverksfunksjonalitet, USB-portar og microSD-kort for lagring. Ein spesielt tilpassa Linux-distribusjon med tilhøyrande pakkesystem vart laga ved hjelp av Yocto-prosjektet. Dette gir eit system som startar svært raskt, og er enkelt å utvida med nye applikasjonar.

Web-grensesnittet vart implementert på toppen av dette, basert på Python-rammeverket Django. Bruken av eit fullverdig operativsystem vil gjera det enkelt å integrera andre applikasjonar i denne sentralen seinare.

Under utvikling og testing vart det tydeleg at rekkevidda til nodane ikkje var så god som ho burde vore. Truleg kjem dette av at impedansen i kretskortbanane mellom mikrokontrollaren og antenna ikkje er heilt riktig. Dette kan difor truleg fiksast i ein ny revisjon av ZigBee-kortet.

Til gjengjeld viste batterilevetida seg å vera svært god, med håp om å få levetid på over 3 år i faktisk bruk ut av 2 AAA-batteri. Òg responstida i systemet var god, med tider på om lag 5 ms frå ein trykkar på ein knapp til reléet skifta tilstand.

Å laga maskinvare tok dessverre svært mykje lenger tid enn ein hadde rekna med. Ein har difor ikkje rukke å implementera den funksjonaliteten ein hadde håpa. I tillegg finst det utfordringar i forhold til sikkerheit i nettverket, og det å få inn nye nodar i nettverket på ein trygg måte. Det er difor stort rom for vidare arbeid.

Totalt sett kan systemet ikkje seiast å vera komplett, slik det framstår i dag. Primært er det noko manglande funksjonalitet, samt den dårlege rekkevidda som er dei største problema. Arbeidet som er gjort her, er likevel eit svært godt grunnlag.

For det første kan nettverket og sentralen nyttast til å implementera nye applikasjonar innanfor husstyring i vid forstand. For det andre burde eit system som her òg ha eit visst kommersielt potensiale, om ein fullfører ein del av den manglande funksjonaliteten og finpussar på brukargrensesnitta.

Referansar

[1] *Energifakta - Vannkraft og elektrisitet i historisk perspektiv*. URL: www.energifakta.no/documents/Miljo%20og%20velferd/Samfunn/Historie.htm (sjekka 12.05.2011).

[2] Ole Morten Haaland. «Lågpris trådlause nettverk for heim- og kontorautomasjon». Prosjektoppgåve. Norges teknisk-naturvitenskaplige universitet, 17. des. 2010.

[3] Control4. *Control4 User Interfaces*. URL: www.control4.com/files/products/data-sheets/Control4-UI.pdf (sjekka 02.02.2011).

[4] *Control4 - 4Store*. URL: www.4store.com/ (sjekka 02.02.2011).

[5] *Abt Electronics and Appliance store*. URL: www.abt.com/product/49526/Control4-C4HC200BEB.html (sjekka 11.05.2011).

[6] Glen Dimplex Nordic. *Bruksanvisning Orion 700*. URL: www.glendimplex.no/filarkiv/2009/05/08/14a04263c53062.pdf (sjekka 21.09.2010).

[7] Glen Dimplex Nordic. *Bruksanvisning GSM Control Plus*. 23. aug. 2007. URL: www.glendimplex.no/filarkiv/2009/05/08/14a03fbb35cb67.pdf (sjekka 21.09.2010).

[8] *elhandel.no*. URL: www.elhandel.no/catalog/index.php?cPath=90 (sjekka 11.05.2011).

[9] NEXA. *Bruksanvisning NEXA LCMR-300*. URL: www.nexa.se/res/Word/se_lcmrmanual.doc (sjekka 02.02.2011).

[10] NEXA. *Operating manual NEXA LBST-604*. URL: www.nexa.se/res/Word/lbst604_eng_se_no_fi.doc (sjekka 02.02.2011).

[11] NEXA. *Grundläggande kännedom om System Nexa*. URL: www.nexa.se/res/pdf/Grundlaeggande-kaennedom-om-System-Nexa.pdf (sjekka 02.02.2011).

[12] *Amigo - Ambient Intelligence for the Networked Home Environment*. URL: www.hitech-projects.com/euprojects/amigo/amigo.htm (sjekka 25.01.2011).

[13] *INRIAGForge: Amigo: Project Filelist*. URL: https://gforge.inria.fr/frs/?group_id=160 (sjekka 25.01.2011).

[14] *Amigo Challenge - Software*. URL: www.hitech-projects.com/euprojects/amigo/software.htm (sjekka 25.01.2011).

[15] *Report on User Requirements - Summary and Conclusions*. URL: www.hitech-projects.com/euprojects/amigo/deliverables/Deliverable%20D1-2-Vol-I_Summary_v10_final.pdf (sjekka 30.05.2011).

[16] *SM4ALL Project*. URL: www.sm4all-project.eu (sjekka 02.02.2011).

[17] Roberto Baldoni, Carola Aiello og Massimo Mecella. *Project Presentation*. Forskningsrapport. SM4ALL-prosjektet, 31. des. 2008. URL: www.sm4all-project.eu/images/stories/deliv/d11.pdf.

[18] Massimo Mecella med fleire. *The SM4All Architecture – 2nd iteration*. Forskningsrapport. SM4ALL-prosjektet, 30. nov. 2009. URL: www.sm4all-project.eu/images/stories/deliv/d21b.pdf.

[19] *eDIANA - Embedded Systems for Energy Efficient Buildings*. URL: www.artemis-ediana.eu/ (sjekka 25.01.2011).

[20] Riccardo Ukmar med fleire. *eDIANA Reference architecture*. Forskningsrapport. eDIANA-prosjektet, EU, jan. 2010. URL: www.artemis-ediana.eu/documents/D21B_eDIANA_Reference_Architecture_m12_STRU.pdf.

[21] Chiara Buratti. *Communication protocol specification*. Forskningsrapport. eDIANA-prosjektet, EU, jul. 2010. URL: www.artemis-ediana.eu/documents/COMMUNICATION_PROTOCOL_SPECIFICATION_M18_UNIBOCB.pdf.

[22] Julen Ugalde med fleire. *In-Home energy management strategies development*. Forskningsrapport. eDIANA-prosjektet, EU, jun. 2010. URL: www.artemis-ediana.eu/documents/HOME_ENERGY_MANAGEMENT_STRATEGIES_DEVELOPMENT_V1.pdf.

[23] *OPEN meter*. URL: www.openmeter.com (sjekka 01.02.2011).

[24] Auguste Ankou med fleire. *Design of the overall system architecture*. Forskningsrapport. OPEN meter-prosjektet, 8. feb. 2010. URL: openmeter.com/files/deliverables/Open%20Meter_D3%201_Architecture_v6_.pdf.

[25] Jose Miguel Arzuaga med fleire. *Specification of OPEN meter OSI layers and multi- metering networking interfaces*. Forskningsrapport. OPEN meter-prosjektet, 19. jun. 2009. URL: openmeter.com/files/deliverables/Open%20Meter_D3.2_v1-1.pdf.

[26] *SOFIA project*. URL: www.sofia-project.eu/ (sjekka 11.05.2011).

[27] Juha-Pekka Soininen med fleire. *Device interoperability: Emergence of the smart environment ecosystems*. Forskningsrapport. Tivit/DIEM-prosjektet, 12. feb. 2010. URL: www.tivit.fi/fi/dokumentit/64/DIEM+whitepaper.pdf.

[28] *Videos - SmartProducts*. URL: www.smartproducts-project.eu/mainpage/videos (sjekka 27.01.2011).

[29] CENELEC The European Comittee for Electrotechnical Standardization. *SmartHouse - The way forward*. Forskningsrapport. SmartHouse Roadmap-prosjektet, URL: ftp://ftp.cenorm.be/CENELEC/SmartHouse/SmartHouseBrochure.pdf.

[30] CENELEC The European Comittee for Electrotechnical Standardization. *Production of a Roadmap for an integrated set of standards for SmartHouse and systems related to it and an Open Event*. Forskningsrapport. SmartHouse Roadmap-prosjektet, 9. feb. 2011. URL: ftp://ftp.cenorm.be/CENELEC/SmartHouse/SmartHouseRoadmap.pdf.

[31] Geir Mathisen. «Fra ide til produkt - Utvikling i henhold til V-modellen». TTK4125 Datastyring. 2008.

[32] *PlantUML*. URL: plantuml.sf.net (sjekka 23.03.2011).

[33] Atmel Corporation. *AVR2005: Design Considerations for the AT86RF230*. Aug. 2007. URL: www.atmel.com/dyn/resources/prod_documents/doc8092.pdf (sjekka 10.03.2011).

[34] Atmel Corporation. *ATmega 128RFA1 Datasheet, preliminary version*. 2010. URL: www.atmel.com/dyn/resources/prod_documents/doc8266.pdf (sjekka 10.03.2011).

[35] Chris Bowick, John Blyler og Cheryl J Ajluni. *RF circuit design*. Amsterdam: Newnes, 2007.

[36] Missouri University of Science og Technology. *Microstrip Impedance Calculator*. URL: emclab.mst.edu/pcbtlc2/microstrip.html (sjekka 12.05.2011).

[37] *Programing Atmega128RFA1 EK with JTAG*. URL: www.avrfreaks.net/index.php?name=PNphpBB2&file=viewtopic&p=742707 (sjekka 10.03.2011).

[38] *Digikey.com*. URL: www.digikey.com (sjekka 16.05.2011).

[39] Elko AS. *Grossistinformasjon, Elko RS16*. URL: www.elko.no/wsp/elko2_nor/frontend.cgi?func=catalog.show&table=PRODUCT&func_id=&prod_id=15135&template=grossist (sjekka 10.03.2011).

[40] OSRAM Opto Semiconductors GmbH. *SmartLED® 0603 Hyper-Bright Low Current LED - L29K*. URL: catalog.osram-os.com/catalogue/catalogue.do;jsessionid=32CB61882C089EDD4B6D0A54E7E1F117?act=downloadFile&favOid=020000020000138e000100b6 (sjekka 05.05.2011).

[41] Panasonic. *6 mm Square Thin Type SMD Light Touch Switches*. URL: industrial.panasonic.com/www-data/pdf/ATK0000/ATK0000CE28.pdf (sjekka 05.05.2011).

[42] Elko AS. *S57/100 6-løps veggboks*. URL: elko.no/wsp/elko2_nor/frontend.cgi?func=catalog.show&template=product&table=PRODUCT&prod_id=11698&search=yes&main_id=1156&l1exp=&l2exp=&l3exp= (sjekka 10.03.2011).

[43] Elko AS. *RS1091 PT stikk m/j I PH*. URL: elko.no/wsp/elko2_nor/frontend.cgi?func=catalog.show&template=product&table=PRODUCT&prod_id=11426&search=yes&main_id=1156&l1exp=1181&l2exp=&l3exp= (sjekka 10.03.2011).

[44] Tyco electronics. *Power PCB Relay RT1 bistable*. URL: documents.tycoelectronics.com/commerce/DocumentDelivery/DDEController?Action=srchrtrv&DocNm=RT1_bistable&DocType=DS&DocLang=EN (sjekka 10.03.2011).

[45] Inc On-Shore Technology. *OSTTA - 5.0 mm series 1-Interlock*. URL: www.on-shore.com/sites/default/files/manuals/osttaxx0161.pdf (sjekka 05.06.2011).

[46] Block. *Short circuit proof PCB transformer VB Range*. URL: https://www1.elfa.se/data1/wwwroot/assets/datasheets/avBlock-VB_data_d-e.pdf (sjekka 05.05.2011).

[47] Micro Commercial Co. *MB05S THRU MB10S 0.5 Amp Single Phase Glass Passivated Bridge Rectifier 50 to 1000 Volts*. URL: 61.222.192.61/mccsemi/up_pdf/MB05S-MB10S(MBS-1).pdf (sjekka 05.05.2011).

[48] Micrel. *MIC5205 - 150mA Low-Noise LDO Regulator*. URL: www.micrel.com/_PDF/mic5205.pdf (sjekka 05.05.2011).

[49] Allegro. *ACS758xCB - Thermally Enhanced, Fully Integrated, Hall Effect-Based Linear Current Sensor IC with* 100 µΩ *Current Conductor*. URL: www.allegromicro.com/en/Products/Part_Numbers/0758/0758.pdf (sjekka 20.05.2011).

[50] beagleboard.org. *Beagleboard -xM System Reference Manual*. URL: beagleboard.org/static/BBxMSRM_latest.pdf (sjekka 06.06.2011).

[51] Kristoffer Rist Skøien og Håvard Vermeer. «General Platform for Unmanned Autonomous Systems - Hardware, Operating System and Peripheral Interfacing». Prosjektoppgåve. Norges teknisk-naturvitenskaplige universitet, 19. des. 2010.

[52] Microchip Technology Inc. *TC1014/TC1015/TC1185 50 mA, 100 mA and 150 mA CMOS LDOs with Shutdown and Reference Bypass*. URL: ww1.microchip.com/downloads/en/DeviceDoc/21335e.pdf (sjekka 05.05.2011).

[53] *AVR: Monitor power supply voltage, for free*. URL: ikalogic.com/band_gap_avr.php (sjekka 26.05.2011).

[54] *PCB Configuration - PCB-POOL.COM*. URL: pcbpool.com/ppuk/order_productconfiguration_js.html (sjekka 09.05.2011).

[55] *BitCloud Profile Suite – ZigBee PRO Public Profiles*. URL: www.atmel.com/products/zigbee/bitcloud.asp?family_id=676 (sjekka 26.05.2011).

[56] *About the Yocto Project*. URL: www.yoctoproject.org/about (sjekka 09.05.2011).

[57] *OpenEmbedded.org*. URL: www.openembedded.org/index.php/Main_Page (sjekka 09.05.2011).

[58] *Django - The web framework for perfectionists with deadlines*. URL: www.djangoproject.com (sjekka 09.05.2011).

[59] *D-Bus*. URL: dbus.freedesktop.org (sjekka 10.05.2011).

[60] *Cherokee Web Server*. URL: www.cherokee-project.com/ (sjekka 30.05.2011).

[61] *ZigBee Remote Control*. URL: www.zigbee.org/Standards/ZigBeeRemoteControl/Overview.aspx (sjekka 19.05.2011).

[62] Direktoratet for samfunnssikkerhet og beredskap. *Import og omsetning av elektriske produkter*. URL: www.dsb.no/no/Ansvarsomrader/EL-sikkerhet/Elektriske-produkter/Import-av-elektriske-produkter/Import-og-omsetning-av-elektriske-produkter/ (sjekka 20.05.2011).

[63] *Tragesym tutorial*. URL: www.geda.seul.org/wiki/geda:tragesym_tutorial (sjekka 13.05.2011).

[64] *PCB Footprint generator*. URL: dlharmon.com/geda/footgen.html (sjekka 13.05.2011).

[65] IBM. *Booting Linux on the BeagleBoard-xM*. URL: www.ibm.com/developerworks/linux/library/l-beagleboard-xm/index.html (sjekka 16.05.2011).

A. Vedleggs-CD

Med denne oppgåva skal det følga ein CD, som inneheld alt som er utvikla i denne oppgåva: skjema, symbol, utlegg, kode og sjølve rapporten. I tillegg er ein del av kjeldene som er nytta i kapittel 2 lagt ved. Filene ligg i ein struktur som vist i figur A.1.

datablad Inneheld datablad for alle komponentar som er nytta i oppgåva.

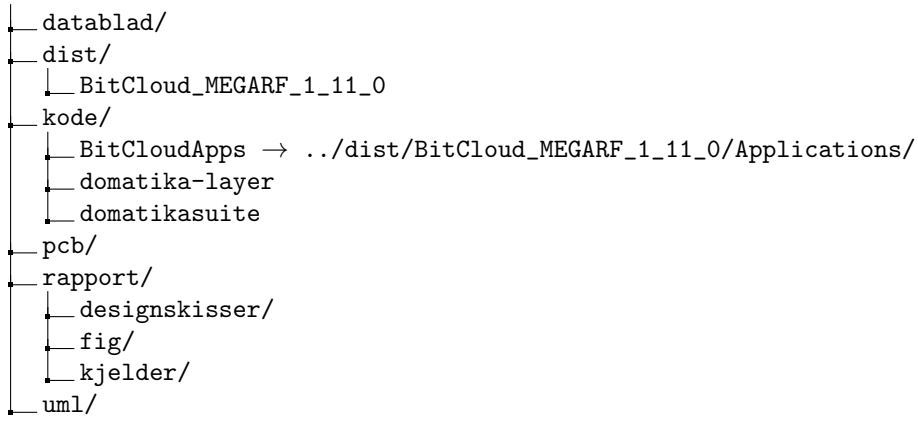
dist Inneheld BitCloud, med dei modifikasjonane som er gjort, inkludert tilpassing av maskinvarelaga og støtte for `PCINT`-pinnane.

kode Inneheld program som er utvikla, inkludert Yocto-laget for heimesentralen og webgrensesnittet domatikasuite.

pcb Inneheld alle skjema og utlegg som er gjort. Nærare skildra i neste delkapittel.

rapport Inneheld Latex-filene for sjølve rapporten, inkludert figurar og dei fleste kjelder ein har referert til.

uml Inneheld UML-diagramma som vart nytta i kapittel 4.



Figur A.1.: Mappestrukturen på CD-en.

B. Oppsett av gEDA

Som nemnt i delkapittel 4.1, vart den relativt laust samanknytta gEDA-pakka nytta for å gjera elektronikkdesign. I dette delkapitlet, blir ein del teknikkar som gjorde arbeidet med pakka lettare skissert.

B.1. Filstruktur

I oppgåva vart det designa fire forskjellige kretskort, men dei nytta ein god del felles komponentar. Ved å nytta ein filstruktur som vist i figur B.1, vart dette langt enklare.

I lista er mapper merka med «/» på slutten, «→» representerer ein såkalla symlink og alt anna er vanlege filer. Med denne strukturen har ein dermed dei filene som er spesielle for kvart kort i sine respektive undermapper, medan alle symbol og fotavtrykk ligg i fellesmapper.

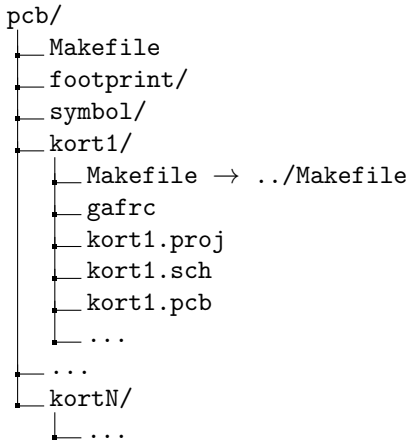
Filene `gafrc` og `kortN.proj` inneheld plassering for prosjektspesifikke symbol og fotavtrykk.

B.2. Symbol og fotavtrykk

Som nemnt i delkapittel 4.1.1, fanst det ikkje fotavtrykk for mange av komponentane som vart nytta i dette prosjektet. Dette gjaldt alt frå sjølve mikrokontrollaren, til transformatoren, releet, kontaktar med fleire. Både for å laga symbol og fotavtrykk finst det små verktøy, i tillegg til kokebok-aktige rettleiingar.

For symbol-laging vart verktøyet `tragesym` nytta. [63]. Dette tek ei reknearkliste med pinnar, namn og type, og konverterer til eit symbol som så kan spesialtilpassast i gSchem. Desse reknearka ligg i mappa `pcb/symbols/`, som `.csv`-filer.

Til fotavtykk-laging vart verktøyet `footgen` nytta. [64]. Dette lagar fotavtrykk-filer, basert på ei tekstfil i eit spesielt format. Desse fotavtrykka kan så nyttast frå gSchem. Råfila som vart nytta i denne oppgåva, ligg i `pcb/footgen/master-footprints.src`.



Figur B.1.: Filstrukturen som vart nytta for kretskortarbeidet.

B.3. Filoverføring

Skjema- og utleggsprogramma i gEDA-pakka nyttar kvar sine filformat, gSchem nyttar `.sch` og PCB `.pcb`. For å overføra data frå det skjema til utlegg, må ein nytta eit eige verktøy, kalla `gschem2pcb`. Dette nyttar `.proj`-fila til å finna dei nødvendige symbol og fotavtrykk, og gjer så konverteringa.

Å gjera ei slik konvertering gir tre filer: Sjølve `.pcb`-fila, ei `.net`-fil og ei `.cmd`-fil. `.pcb`-fila er sjølve utlegget, `.net`-fila er ei oversikt over netta som finst i prosjektet, medan `.cmd`-fila er ei liste med kommandoar som kan køyrast frå PCB for å namngje alle pinnar.

Dersom det allereie finst ei `.pcb`-fil, vil ikkje `gschem2pcb` oppdatera denne med nye komponentar av seg sjølv. Ein må frå PCB spesifikt gå inn og velja «File» -> «Load layout data from file» for å henta inn nye komponentar. Gamle komponentar blir automatisk fjerna.

B.4. Materialliste

For å generera ei materialliste, såkalla «Bill of materials», finst det eit eige verktøy, kalla `gnetlist`. Dette gir ut ei liste over komponentar som kommaseparert tekst, som enkelt kan behandlast i rekneark.

B.5. Makefil

For å gjera køyring av ein del av desse verktøya enklare, vart det laga ei felles Makefil for kretskorta. Denne ligg i rotmappa for kretskort, og er derfrå tenkt symlenka inn i kvar kortmappe.

Denne fila tilbyr følgande kommandoar:

bom Lag materialliste, som skildra i B.4.

clean Fjern alle midlertidige filer.

cleanpcb Fjern alle utleggsfiler.

pcb Lag kretskortfiler, som skildra i B.3.

pdf Lag ein pdf av skjemateikninga, ved hjelp av `gschem` og `ps2pdf`. Fila blir lagt i undermappa pdf.

C. Oppsett og bruk av Yocto med Beagleboard

C.1. Starta eit Yocto-prosjekt

For å starta eit Yocto-prosjekt, må ein først henta ned Poky frå nettsida deira. [56]. I denne oppgåva vart versjon 5.0 nytta, som ein del av Yocto 1.0. Når ein pakkar ut denne fila, får ein ei mappe kalla `poky-bernard-5.0`. Inni denne mappa ligg det eit skript kalla `poky-init-build-env`. Ved å køyra dette på følgande måte, får ein ei arbeidsmappe, der konfigurasjon og resultat frå Poky blir liggande.

```
$ source poky-bernard-5.0/poky-init-env namn-på-arbeidsmappe
```

For å fortelja Poky at ein ønskjer å kompilera for Beagleboard, er det nok å endra `MACHINE`-variabelen i `conf/local.conf` i arbeidsmappa. Denne skal vera sett til `beagleboard` for å få eit resultat som kan nyttast med Beagleboardet.

C.2. Laga eit eige lag

Som nemnt i delkapittel 5.5.1, vart den konfigurasjonen av Poky som var nødvendig skilt ut i eit eige lag. Eit lag i Poky-samanheng tyder eigentleg berre ei samling oppskrifter, meint for ein spesifikk bruk.

Det einaste kravet Poky set til eit lag, er at det i base-mappa for laget skal finnast ei undermappe `conf`, som skal innehalda ei `layer.conf` som gir konfigurasjonen for laget sjølv. Poky set ingen avgrensingar på korleis strukturen av oppskrifter i eit lag skal vera, men etter å ha sett på korleis Poky sjølv gjer det, vart det nytta ein struktur som følger:

Med denne strukturen er det forsøkt å dela oppskriftene etter kva dei er for. I Poky-terminologi er eit «image» det høgste nivået ein opererer på. Eit bilete spesifiserer eit sett med «tasks», som skal installerast i biletet. Ein «task» definerer ei moglegheit eller ein funksjonalitet systemet har. Til dømes inneheld «task-domatika-dev» pakkar som var nyttige under utvikling, men ikkje i eit produksjonssystem.

```
domatika-layer/
├── conf/
│   └── layer.conf
├── recipes-images
│   └── domatika-image-coordinator.bb
├── recipes-kernel
│   ├── linux-yocto_git.bbappend
│   └── add_gpio.cfg
├── recipes-server
│   ├── files
│   │   └── cherokee.init
│   ├── cherokee.inc
│   └── cherokee_1.2.2.bb
├── recipes-tasks
│   ├── task-domatika-webui.bb
│   ├── task-domatika-dev.bb
│   └── task-domatika-base.bb
└── recipes-utils
    ├── python-django_1.3.bb
    ├── django-south_0.7.3.bb
    ├── nano_2.2.6.bb
    └── python-pyserial_2.4.bb
```

Både «image» og «task» er abstrakte oppskrifter, som nesten berre avheng av andre pakkar, utan å vera faktiske installasjonsoppskrifter. Dei andre oppskriftene, for Django, Cherokee og Nano, spesifiserer derimot korleis pakkene skal installerast.

I sjølve Poky-pakka finst det òg ei undermappe `conf/`, og i denne ligg fila `layers.conf`. Her vel ein kva lag som skal nyttast ved kompilering, og det er her ein må aktivera eigne lag.

C.3. Bruk på Beagleboardet

C.3.1. Partisjonering

Beagleboardet startar frå eit MicroSD-kort. Dette må delast i to partisjonar for å kunna startast frå, ein liten (til dømes 70 megabyte) FAT[1]-partisjon i starten av kortet. Denne partisjonen er oppstartspartisjonen, og skal innehalda sjølve Linux-kjerna, oppstartslastaren U-boot, samt ei binærfil kalla

[1] File Allocation Table - eit filsystem tradisjonelt brukt av DOS

MLO, som er ansvarleg for å lasta U-boot.

Resten av kortet kan nyttast som ein stor partisjon for rotfilsystemet til Linux-distribusjonen, typisk i ext3[2]-format. Det finst skript tilgjengelege på nett for å automatisera oppsett av kortet. [65]

C.3.2. Installasjon

I mappa `arbeidsmappe/tmp/deploy/images/`finst dei interessante filene i forhold til oppstart av Beagleboardet. I ei faktisk mappe vil desse filene ha versjonsnummer eller dato som ein del av filnamnet, men det endrar ikkje funksjonen til filene.

MLO Binærfil som er ansvarleg for å lasta U-boot.

u-boot.bin Oppstartslastaren.

uImage Sjølve Linux-kjerna.

domatika-image-coordinator.tar.bz2 Komprimert versjon av rotfilsystemet.

Dei tre første, `MLO`, `u-boot.bin` og `uImage` skal kopierast inn på oppstartspartisjonen på SD-kortet. I arbeidsmappa har dei versjonsnummer som ein del av filnamnet. Dette bør fjernast under kopieringa.

Rotfilsystemet må pakkast ut på Linux-partisjonen på SD-kortet. Dette kan gjerast ved å vera i den rette mappa på SD-kortet, og køyra følgande:

```
$ tar -jxvf ~/bane/til/domatika-image.coordinator.tar.bz2
```

C.3.3. Oppstart av Beagleboardet

Beagleboardet startar så fort det får straum, og som ein del av oppstarten tek det i mot ein IP-adresse ved hjelp av DHCP [3]. Det er ikkje utvikla nokon enkle måtar å finna ut kva IP-adresse kortet har fått på, og ein må difor nytta ein serie-terminal for å finna denne første gong.

På Beagleboardet finst det ein fast serieport, som fungerer som terminal for Linux-installasjonen. Ved å kopla denne til ein PC, til dømes ved hjelp av eit USB-til-serieport-adapter, kan ein i ein terminalemulator få opp den grunnleggjande konsollen Linux tilbyr. Beagleboardet sin serieport sender og mottek med 115200 bps, 8 bit per byte og 1 stoppbit.

Det er ikkje sett opp andre brukarar på Beagleboardet enn `root`-brukaren, og denne har tomt passord.

IP-adressen blir skrive til konsollen under oppstart, men om ein ikkje får han med seg, kan han òg finnast ved å køyra kommandoen `ifconfig` frå konsollen.

[2] «third extended filesystem», det foreløpig mest brukt filsystemet for Linux-distribusjonar
[3] Dynamic Host Configuration Protocol

C.3.4. Pakkesystem

Ein har valt å nytta pakkesystemet `opkg`, som er meint nettopp for den typen innebygde system som det her er snakk om. `opkg` nyttar pakkeformatet `.ipk`.

I mappa `arbeidsmappe/tmp/deploy/ipk` finst alle filene som trengst for å laga ei pakkebrønn. Denne er delt i tre, pakkar som fungerer på alle arkitekturar, pakkar som fungerer på ARM og pakkar som fungerer berre på Beagleboardet. For å nytta denne pakkebrønna frå Beagleboardet, må denne sjølve `ipk`-mappa først publiserast ved hjelp av ein webserver. Deretter må banen til alle dei tre delane leggjast inn i `/etc/opkg/feed.conf` på sjølve Beagleboardet med følgande format:

```
src/gz all http://url-til-pakkebrønn/all/
src/gz armv7a http://url-til-pakkebrønn/armv7a/
src/gz beagleboard http://url-til-pakkebrønn/beagleboard/
```

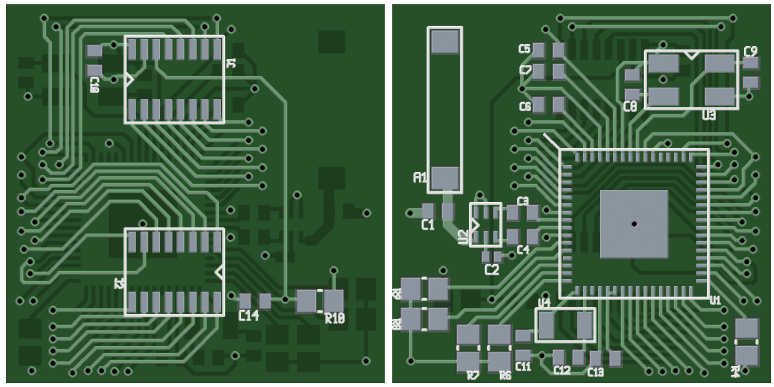
I det ferdige biletet er dette sett opp på førehand.

Dermed kan nytta standard opkg-kommandoar, som å oppdatera pakkelista med `opkg update`, installera pakkar med `opkg install pakkenamn`, eller oppgradera alle pakkar med `opkg upgrade`.

Merk at for å få Poky til å laga listene over kva pakkar som er tilgjengelege i pakkebrønna på nytt, må ein køyra `bitbake package-index` på maskinen der ein har bygd pakkebrønna. Dette er nødvendig til dømes etter å ha bygd nye pakkar eller nye versjonar av gamle pakkar.

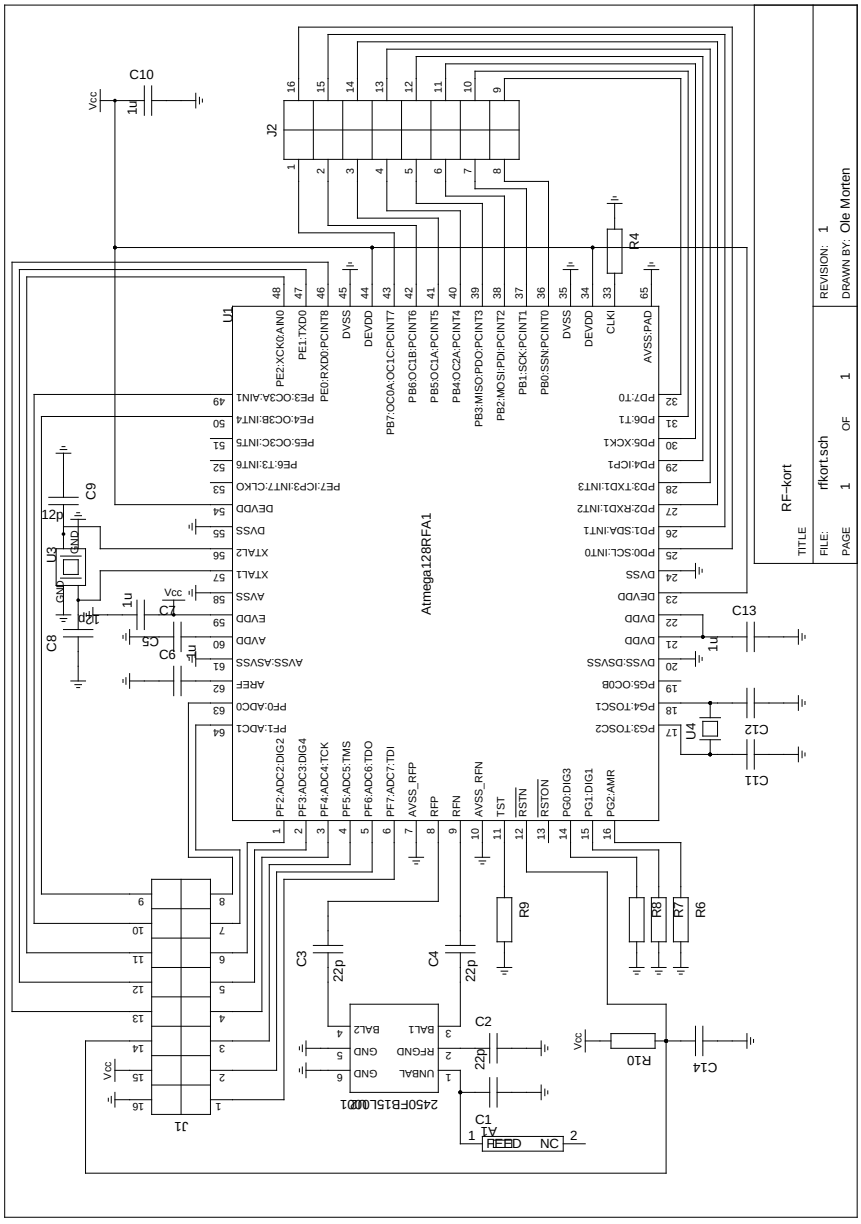
D. Skjema og utlegg

Skjema og utlegg er vist som figurar på dei følgande sidene, med rekkefølge ZigBee-kort, lysbrytarnode, relénode og heimesentral.

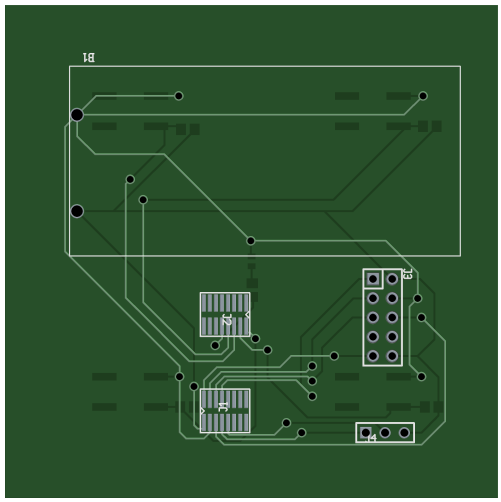


(a) Baksida (b) Framsida

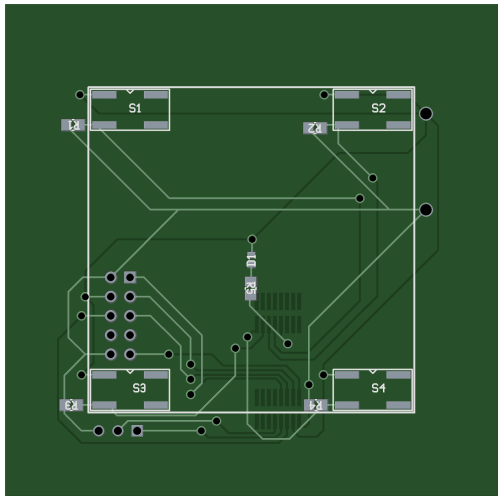
Figur D.1.: Utlegget av ZigBee-kortet. Dobbel storleik.



Figur D.2.: Skjemateikning for ZigBee-kortet.

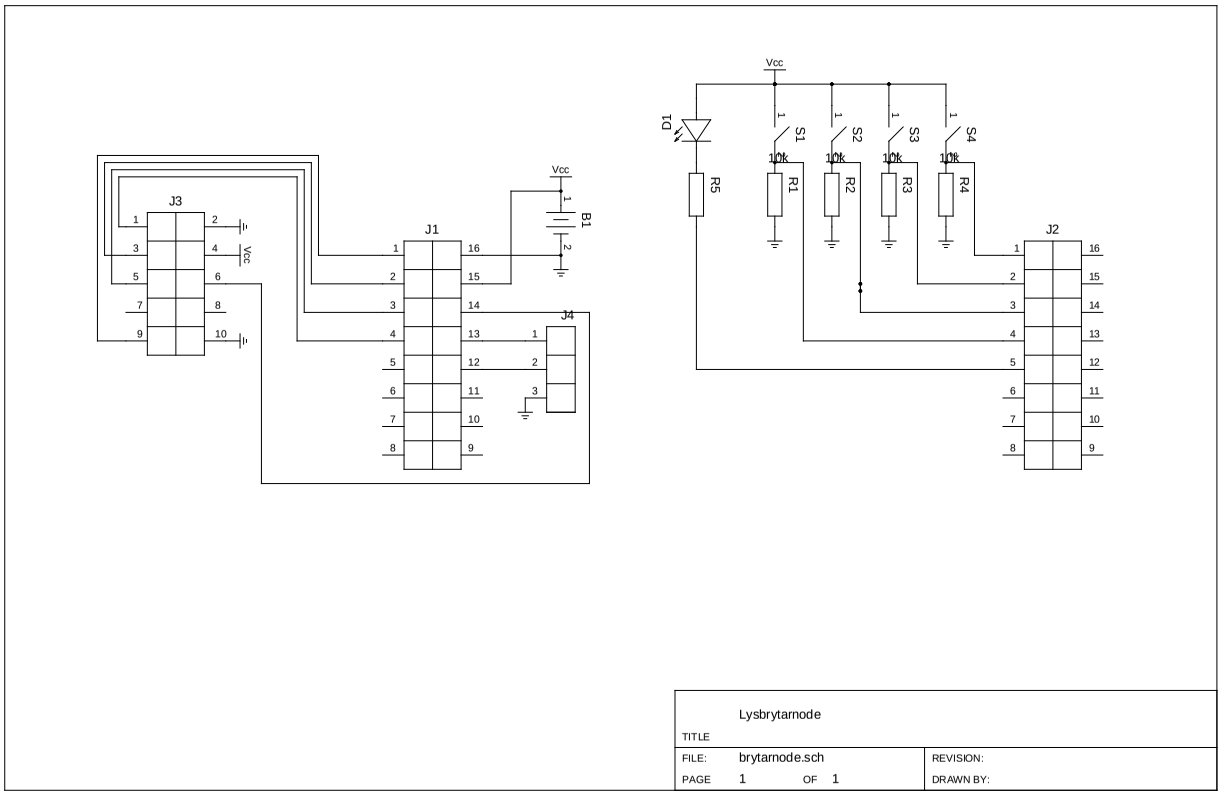


(a) Baksida

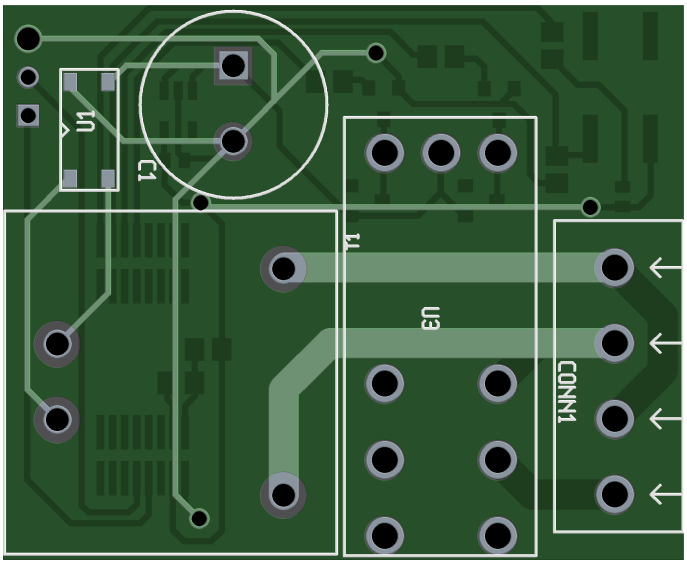


(b) Framsida

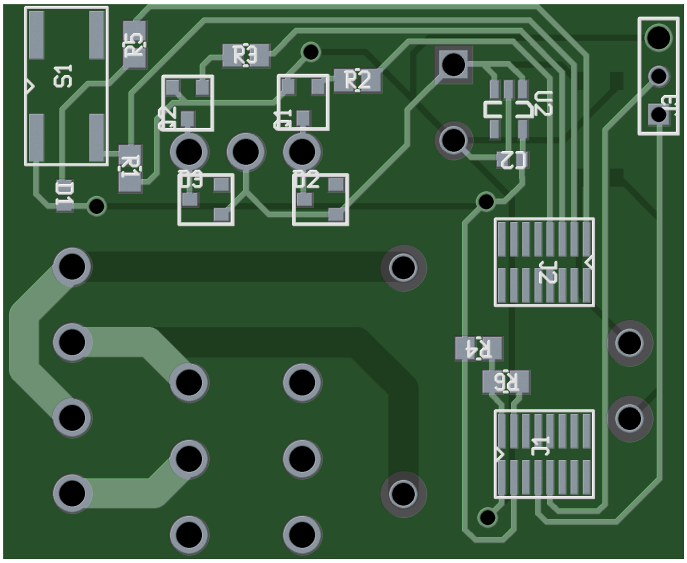
Figur D.3.: Utlegget av lysbrytarnoden.



Figur D.4.: Skjemateikning for lysbrytarnoden.

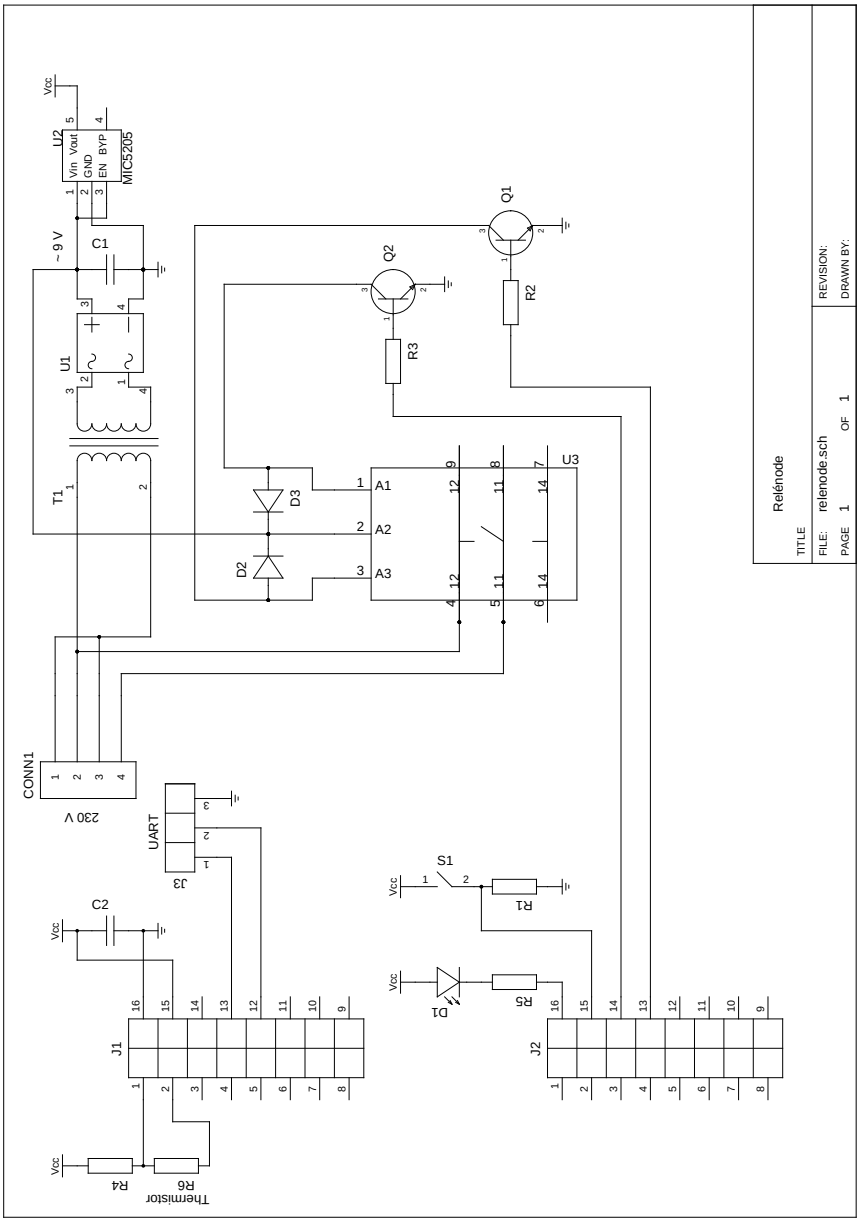


(a) Baksida

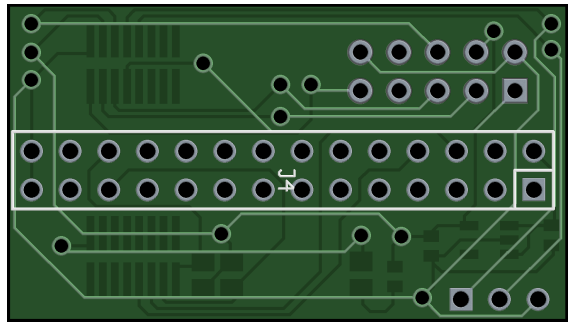


(b) Framsida

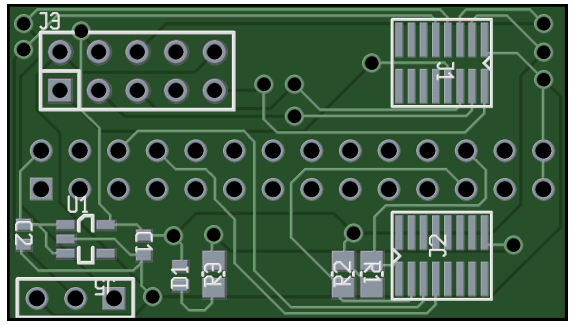
Figur D.5.: Utlegget av relénoden. Dobbel storleik.



Figur D.6.: Skjemateikning for relénode.

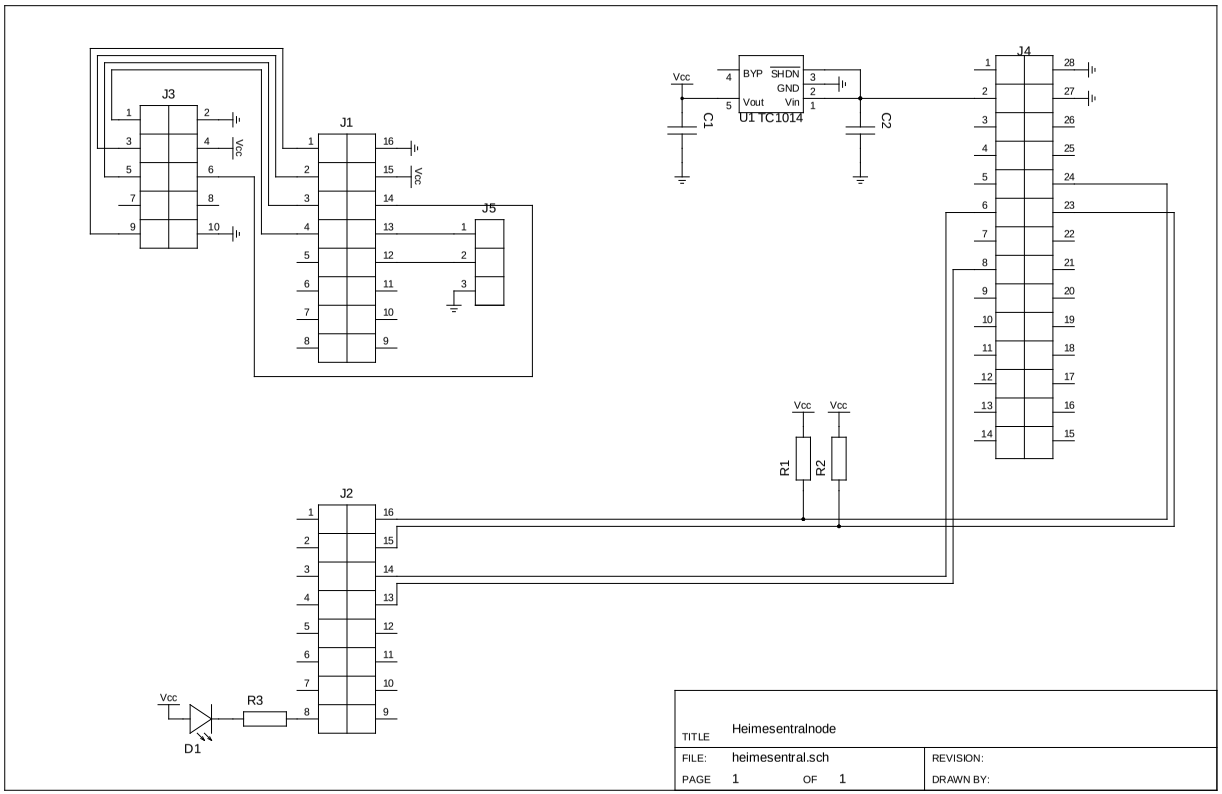


(a) Baksida



(b) Framsida

Figur D.7.: Utlegget av heimesentralnoden. Dobbel storleik.



Figur D.8.: Skjemateikning for heimesentralnoden.