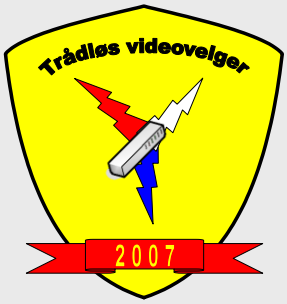
HOVEDPROSJEKT:

Trådløs styring av videovelger
Et hjelpemiddel for fjernsynsproduksjon i virtuelle miljøer



FORFATTERE:

Øivind Stuan
Anders Johansen

Dato: 29.05.07

Sammendrag av hovedprosjekt

Tittel:	Trådløs styring av videovelger -	Nr. : 3
	Et hjelpemiddel for fjernsynsproduksjon i virtuelle miljøer	Dato : 29.05.07
Deltakere:	Øivind Stuan	
	Anders Johansen	
Veileder:	Håkon Solum	
Oppdragsgiver:	Høgskolen i Lillehammer v/ Claus Knudsen	
Kontaktperson:	Claus Knudsen og Sigmund Andresen	
Stikkord (4 stk)	Fjernkontroll, basestasjon, RF-kommunikasjon, webgrensesnitt	
Antall sider: 55	Antall bilag: 18	Tilgjengelighet (åpen/konfidensiell): Åpen

Kort beskrivelse av hovedprosjektet:

Ved HiL eksperimenteres det med tv-produksjon og bruk av blant annet mediatorer for å lage virtuelle miljøer. I den sammenheng er det ønskelig for aktørene å kunne orientere seg i omgivelsene, noe som gjøres ved hjelp av kildevelgere. Frem til nå har dette vært løst ved at valget av videokilde enten har blitt styrt av produsenten eller at en stasjonær kildevelger ble plassert i umiddelbar nærhet av aktøren. Den første løsningen frarøver aktøren kontroll over situasjonen, mens den andre er problematisk fordi man får et uønsket element i scenen.

I denne hovedoppgaven skulle det lages en fjernkontroll med en tilhørende basestasjon for styring av kildevelgeren. Kommunikasjonen mellom disse to enhetene skulle foregå over radiofrekvenser. Med dette vil aktørene i tv-produksjonen kunne orientere seg i miljøet med noen enkle bevegelser på fjernkontrollen som ligger gjemt i hånden.

Valg av løsning har ingen spesielle restriksjoner med unntak av at fjernkontrollen bør være liten og ha lavt effektforbruk. Det er et viktig poeng at kontrollen ikke skal være synlig i kamera.

Forord

Hovedprosjektet er en obligatorisk og avsluttende oppgave på 15 studiepoeng som går over det siste vårsemesteret ved Høgskolen i Gjøvik. Studentene velger selv oppgave ut i fra en liste som blir presentert.

Denne hovedprosjektoppgaven startet som et oppdrag i faget Digitale Nettbaserte Produksjonssystemer under TIDE/SVR-prosjektet høsten 2006. På grunn av begrenset tid var det ikke mulig å gjennomføre oppgaven som et ledd i dette faget. Det som ble gjort var å få satt tanker og ideer om hvordan det kunne løses i system. Siden den ikke lot seg gjennomføre i løpet av høsten 2006 ble oppgaven lagt inn som et forslag til hovedprosjekt våren 2007 for medie-elektronikklassen. Oppgaven går ut på å lage et hjelpemiddel for aktører i virtuelle miljøer i en fjernsynsproduksjon.
Under TIDE-prosjektet ble det gjort forsøk med to mediatorer for at to personer på to forskjellige geografiske steder skulle kunne kommunisere med hverandre og samtidig føle et nærvær. Under disse forsøkene ble det strukket flere signalkabler til kildevelgere med mekaniske trykknapper, som ble plassert i aktørenes umiddelbare nærhet. Kildevelgerne var til tider vanskelige å gjemme bort, og aktørene som brukte dem gjorde unaturlige bevegelser som avslørte at videovelgerne var tilstede. I tillegg lagde knappene mye lyd som ble fanget opp under opptakene. Dette er noe man ikke kan leve med i en profesjonell produksjon.
For å rette fokus mot løsninger som kan brukes i profesjonell tv-produksjon har vi samarbeidet med Network Electronics ASA som leverer utstyr til tv-produksjonsselskaper som blant andre NRK.

Vi vil takke:

- HiL og Claus Knudsen for å tildele oss denne spennende oppgaven som passet godt for medieelektronikklassen.
- Håkon Solum for råd og veiledning under prosjekttiden.
- Arne Wold for lån av to stykk nRF24L01-utviklingssett.
- Network Electronics ved Geir Morten Wold og Morten Abrahamsen som har vært behjelpelige med lån av videovelger, teknisk støtte og et hyggelig møte i Sandefjord.
- John Elvesveen for bestilling av nødvendige komponenter og generell assistanse.

Gjøvik 29.05.07

Øivind Stuan

Anders Johansen

Innhold side

Figur og tabell liste

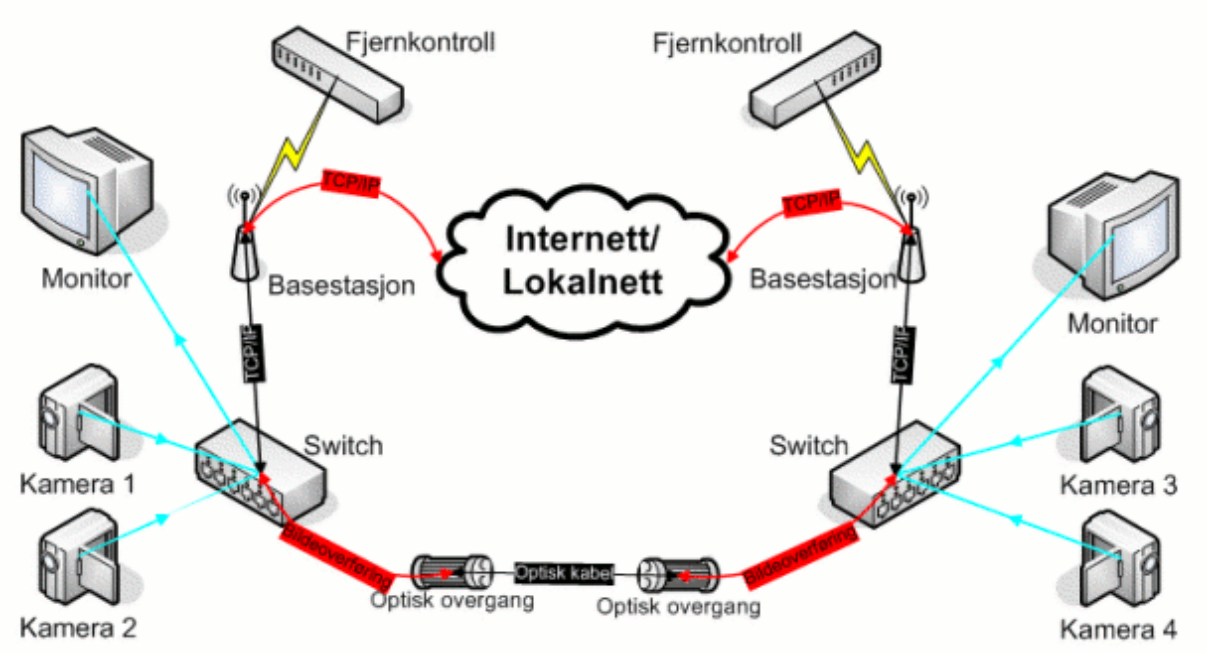
1.0 Innledning

1.1 Definering av oppgaven

Denne hovedoppgaven gikk ut på å lage en trådløs styring av videovelgere for bruk i tv-produksjon. Denne enheten skulle være et hjelpemiddel for aktører som opptrer i virtuelle miljøer. Den trådløse enheten skulle være enkel i bruk og helst liten nok til å skjules i hånden. Lengst mulig batterilevetid var selvfølgelig også ønskelig.

Basestasjonen skulle ha et nettsidebasert brukergrensesnitt for å gjøre lagring og endring av oppsett enklest mulig uten å måtte ha tilgang på spesielle verktøy. Enklere operative innstillinger skulle kunne gjøres på selve basestasjonen ved hjelp av et enkelt menysystem. I tillegg var det ønskelig å kunne bruke flere fjernkontroller til å kommunisere med samme basestasjon.

Figur 1 viser hvordan systemet var tenkt integrert i et større audiovisuelt kommunikasjonssystem.



Figur 1: Et eksempel på oppsett ved bruk av vår trådløse løsning

1.2 Målgruppe

Denne oppgaven er ment for ingeniører innen elektronikk og for dem som eventuelt skulle få en oppgave om utbedring av den løsningen som representeres her. For å få utbytte av denne rapporten er det en fordel å ha grunnleggende kunnskap innen elektronikk og programmering av mikroprosessorer. Kretsene som ble laget er veldig enkle.

De som studerer denne rapporten skal få et detaljert innblikk i hvordan vi gikk frem for å løse oppgaven og se hva slags resultat som ble oppnådd. På den måten skal det oppnås høy grad av forståelse rundt virkemåte og oppbygning av produktet, i tillegg til oversikt over potensielle forbedringspunkter.

1.3 Faglig bakgrunn hos studentene

Begge studentene har gått Medieelektronikk (04HBINEM) de to siste årene ved HiG. Øivind tok sitt første studieår på Høgskolen i Sør-Trøndelag og har hatt elektronikk på yrkesfaglig videregående skole. Anders har gått alle tre årene på HiG med allmennfaglig bakgrunn fra videregående skole.

1.4 Valg av arbeidsformer

Arbeidsformen i gruppen har vært at hvert medlem har hatt sine ansvarsområder. Anders har hatt hovedansvaret for fjernkontrollen og Øivind har jobbet med basestasjonen. Ved problemer har begge forsøkt å hjelpe hverandre videre. Arbeidet har hovedsakelig forgått på elektronikklaben på HiG ved hjelp av datamaskiner, da mye av oppgaven har bestått i programmering av både fjernkontroll og basestasjon. Oppdateringer om status er lagt ut på gruppens hjemmeside underveis i prosjekttiden. Videre har vi vært i kontakt med Network Electronics ved Geir Morten Wold og Morten Abrahamsen. I starten av prosjektet ble det avholdt et møte i Network Electronics' lokaler i Sandefjord hvor vi presenterte oppgaven og fikk diskutert mulige løsningsalternativer.

2.0 Teoretisk og praktisk grunnlag

Gruppens teoretiske og praktiske grunnlag som har vært vesentlig for denne oppgaven er tre år med studier innen elektronikk på høgskolenivå. De mest relevante fagene fra tidligere semestre har vært elektronikk 1 og 2, programmering og mikroprosessorteknikk 2. Det tredje året var det en nyttig erfaring å ha faget Digitale Nettbaserte Produksjonssystemer slik at gruppen fikk litt innsikt i hvordan en tv-produksjon fungerer, og ut i fra dette kunne danne et bilde av hvordan sluttresultatet helst burde fungere. Dessverre var det ikke mulig å få lagt opp en fagplan med faget elektronikkonstruksjon for oss, siden det kolliderte med vårt undervisningsforløp ellers.

3.0 Tekniske kravspesifikasjoner

3.1 Generell begunnelse

Kravspesifikasjonen til fjernkontrollen var at den skulle være såpass liten at den kunne gjemmes i hånda til aktøren som skulle bruke den. Den burde bruke radiofrekvenser (rf) for kommunikasjon med basestasjonen. Dette for å hindre problematikk med infrarød overføring der kravene til fri sikt ville gjøre det vanskelig å skjule fjernkontrollen og samtidig beholde kommunikasjonen. Fjernkontrollen burde ha et enkelt og intuitivt brukergrensesnitt, da aktøren ikke skulle behøve å se på kontrollen for å velge riktig kilde. Levetiden på batteriene burde også være lang for å unngå at batteriene kunne gå tomme i løpet av én produksjon.

Basestasjonen måtte ha et TCP/IP-grensesnitt for å kommunisere med videovelgeren. Den måtte også ha et display og bryterpanel slik at man kunne velge ferdige profiler på en lettvint måte. Det skulle være mulig å lagre flere oppsett i basestasjonen. Den mest praktiske løsningen var å gjøre dette via et nettsidebasert brukergrensesnitt, hvor man kunne vise gjeldende oppsett og slette eller endre etter behov. Basestasjonen måtte også kunne stå utenfor synsrekkevidden til kameraene, noe som igjen satte krav til rekkevidden på den trådløse kommunikasjonen.

Grunnleggende kravsspesifikasjon for fjernkontroll:

- Radiobasert overføring for å slippe sikteproblematikk
- Kompakt og logisk utforming for å kunne brukes uten å se på den
- Lavt strømtrekk
- Rekkevidde > 10 meter

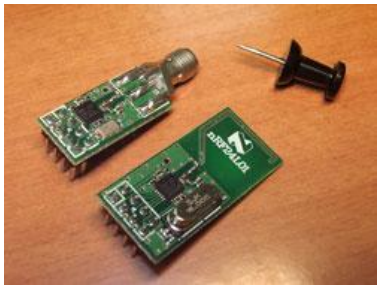
Grunnleggende kravsspesifikasjon for basestasjon:

- Radiomottager med mulighet for "paring" slik at man slipper forstyrrelser fra andre fjernkontroller
- Ethernettilknytning for styring av signalvelgere og endring av konfigurasjon
- Relativt enkel konfigurering via nettleserbasert brukergrensesnitt
- Kunne håndtere flere fjernkontroller samtidig.

4.0 Utvikling av fjernkontroll

4.1 Komponentvalg

4.1.1 RF-komponenten



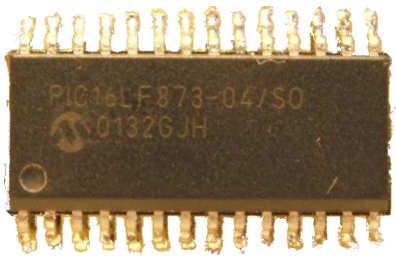
Figur 2: nRF24L01 uten og med integrert antenne

I og med at det var ønskelig at den som brukte fjernkontrollen skulle kunne gjemme den og ikke kreve fri sikt til basestasjonen valgte vi å gå for en RF-basert overføring i stedet for infrarødt.

Det ble tidlig informert om at det fantes et nRF24L01 (L01) utviklingssett fra Nordic Semiconductor på skolen. Denne kretsen kan drives av et spenningsnivå på mellom 1.9V og 3.6V. De trådløse overføringene blir utført i 2,4GHz-båndet i likhet med trådløse tastatur, mus og lignende. Ved sending er det maksimale strømtrekket 11mA med en utgangseffekt på 0dBm (1mW) og i dvale 0.9μA.

L01 hadde også mer enn god nok overføringshastighet da det eneste som skulle overføres var fjernkontrollens ID-nummer, batteristatus og hva slags retning joysticken ble trykket inn. Det eneste denne løsningen manglet var en innebygd mikrokontroller. Nordic Semiconductor kunne levere en annen brikke, nRF24E1, med integrert mikrokontroller og ellers identiske egenskaper som L01. Derfor ble også denne løsningen vurdert i en liten periode, men vi fant snart ut at et utviklingssett for denne modellen ville ta halvparten av det totale budsjettet og vi gikk derfor tilbake til modellen som var tilgjengelig på skolen.

4.1.2 Mikrokontroller



Figur 3: PIC16LF873

Siden vi endte opp med å ikke bruke nRF24E01 trengte fjernkontrollen en mikrokontroller. Her var både størrelse og strømtrekk av stor betydning. Valget falt på en PIC16LF873 fordi den var tilgjengelig på skolens lager og kunne kjøres på 3V drivspenning. For å senke strømtrekket maksimalt kunne vi benytte oss av SLEEP-mode i mikrokontrolleren. Den ville da

trekke minimalt med strøm (<1μA) når den ikke trengte å utføre noen oppgaver. Den er overflatemontert og passet vårt formål perfekt. Denne mikrokontrolleren har 128byte EEPROM noe som er mer enn nok for vår bruk.

Selve programmeringen kunne forgå i C, noe som gjorde programmeringsjobben litt lettere enn om den måtte bli gjort i assemblykode. Programmeringen ble gjort i MPLAB IDE og overført med en MPLAB ICD2 enhet. Kompilatoren vi brukte var studentutgaven av CC5X fra B Knudsen Data.

4.1.3 Bryterfunksjon

Hvordan bryterfunksjonen skulle være var lenge et stort spørsmål da det var ønskelig at denne delen skulle være så liten som mulig slik at den ikke ble ødeleggende med tanke på størrelsen på brukergrensesnittet. Det ble brukt en del tid på å finne en god løsning på dette, men vi endte opp med en meget elegant og kompakt joystick-løsning. Med en grunnflate på 7,5mm x 7,5mm og en høyde på 5mm ble ikke denne delen lenger utslagsgivende for størrelsen på fjernkontrollen.

Figur 4: Joystick på fjernkontrollen

4.1.4 Batteri

For å holde størrelsen på fjernkontrollen til et minimum ble det bestemte at det skulle brukes 3V knappcellebatteri da disse er tynne og lette. De forskjellige størrelsene som var oppe til vurdering var 16mm og 20mm med henholdsvis 125mAh og 210mAh kapasitet. Begge størrelsene ble forsøkt på forskjellige utgaver av fjernkontrollens kretskort, men i den endelige fjernkontrollen endte vi opp med å bruke den minste varianten for å forenkle konstruksjonen.

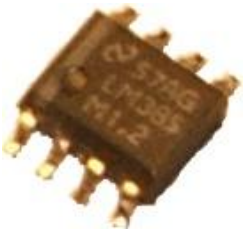
Figur 5: Knappcellebatterier som blir brukt i fjernkontrollen

4.1.5 Krystall

Figur 6: 32kHz krystall

I og med at den laveste hastigheten mikrokontrolleren kan jobbe på er 32kHz valgte vi i utgangspunktet en krystall på 32,768kHz. Dette senker strømtrekket i kretsen til et minimum. I løpet av konstruksjonsperioden oppdaget vi imidlertid at vi ikke fikk brukt debuggingsmulighetene med denne krystallen og vi valgte derfor å bytte til 2MHz klokkefrekvens. Valget av denne baserte seg hovedsakelig på krystallens fysiske dimensjoner, da 1MHz krystallene som var tilgjengelige på skolens lager var for store til å få plass i fjernkontrollen.

4.1.6 Spenningsreferanse



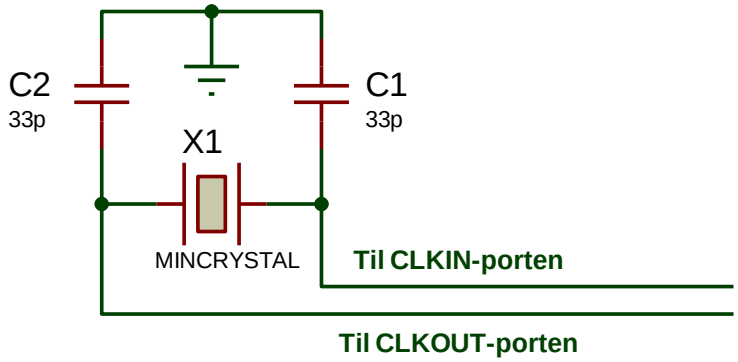
Figur 7: Spennignsreferansen LM385

For å kunne ta i bruk A/D-konverteringen i mikrokontrolleren måtte vi ha en spenningsreferanse. Til dette formålet planla vi å bruke en LM385 spenningsreferanse som kunne gi ut en referansespenning på 1.24V – 5.30V.

4.2 Konstruksjon

Når valg av komponenter var fullført kunne fokus settes på selve konstruksjonen av kretsen. Det skulle brukes en drivspenning på 3V fra batteriet til å drive hele kretsen.

Tilkoblingen av mikrokontrollerens krystall var veldig rett frem da oppskriften sto i databladet. Det samme gjorde kondensatorverdiene som skulle brukes for de forskjellige frekvensene. I vårt tilfelle der det skulle brukes en klokketakt på 2MHz måtte det velges kondensatorer på 33pF (se Figur 8: Kretsskjema for tilkobling av krystall).



Figur 8: Kretsskjema for tilkobling av krystall

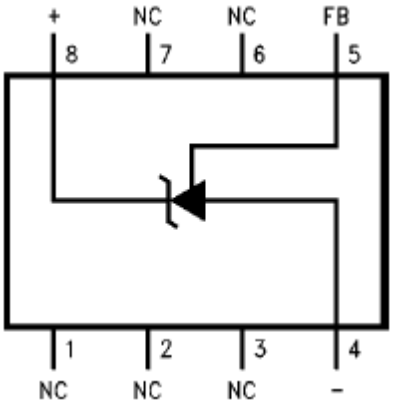
Joysticken vi valgte kunne beveges i alle fire himmelretninger samt inntrykk. Siden debuggingsenheten ICD2 har interne pull-down motstander var vi nødt til å bruke samme type løsning på fjernkontrollkortet for å at debuggingen skulle fungere. Hvis vi hadde brukt pull-up hadde vi endt opp med en spenningsdeling på inngangene som ble brukt til programmering og debugging. Funksjonen for weak-pull-up på mikrokontrollerens PORTB ble skrudd av.

Siden vi skulle benytte oss av interrupt for å vekke mikrokontrolleren fra SLEEP koblet vi inntrykk-funksjonen til den eksterne interrupt-inngangen (RB0/INT). De andre bevegelses punktene i joysticken ble tilkoblet de fire siste pinnene på PORTB (RB4 - 7) fordi disse pinnene støtter funksjonen interrupt-on-change.

Når det gjaldt $\overline{\text{MCLR}}$ brukte vi en pull-up motstand slik at vi unngikk problemer med uønskede resettinger av mikrokontrolleren.

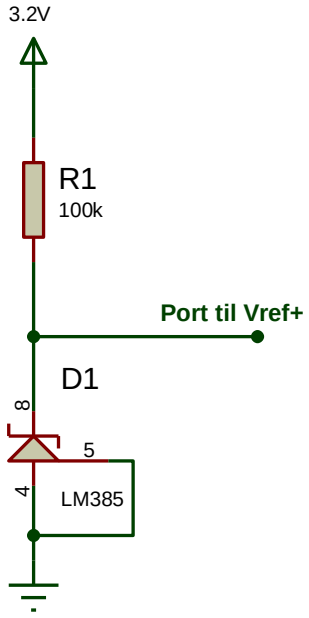
Kommunikasjon over SPI ga ingen større problemer med tegning av kretsen da det bare var å trekke baner direkte mellom de riktige pinnene på mikrokontrolleren og tilkoblingspunktene på L01-kortet.

Det var ønskelig å kunne følge med på spenningsnivået hos batteriet. For å kunne gjøre det var tanken at man sendte batteristatusen sammen med joystickretningen hver gang man valgte en videokilde slik at statusen kom opp displayet på basestasjonen. Til dette måtte A/D-konverteren i mikrokontrolleren tas i bruk.



Figur 9: LM385 - Pinne 5 er feedback, 4 er inngang og 8 er utgang

For å kunne ta i bruk A/D-konverteren, kan man lage en spenningsreferanse ved hjelp av en LM385 for å ha noe å sammenligne batterispenningen med. Inngangen og feedback kobles til jord mens utgangen kobles til Vref+ på mikrokontrolleren. Strømmen kan begrenses ved å koble en 100kΩ motstand mellom utgangen og batterispenningen (se Figur 10: Koblingsskjema for referansespenning). Dette gjør at man ender opp med en referansespenning på 1,24V.

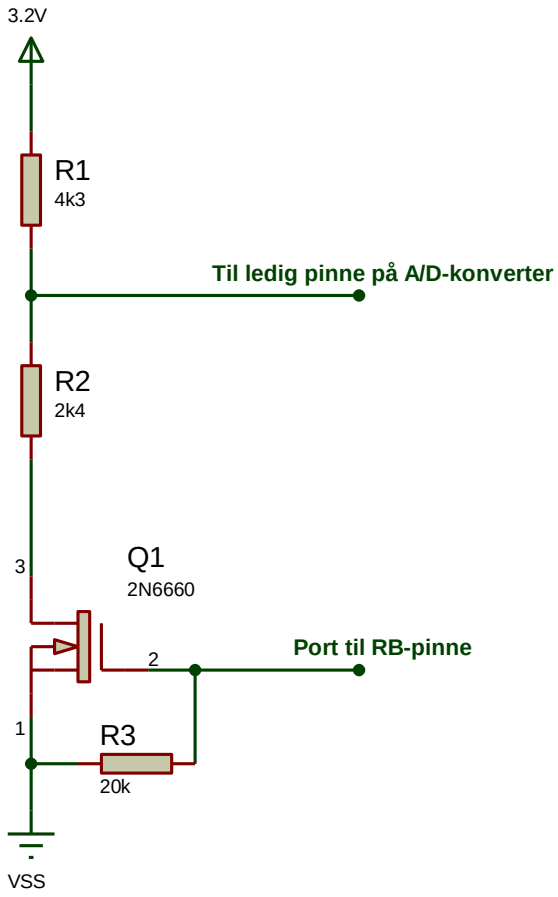


Figur 10: Koblingsskjema for referansespenning

For å løse problemet med at spenningsdelingen ikke skal trekke for mye strøm, kan man lage en enkel kobling som baserer seg på bruken av to porter på mikrokontrolleren (se figur 11). Den ene tilkoblingen må være på en av de tilgjengelige A/D pinnene og den andre må tilkobles en ledig digital utgangspinne på mikrokontrolleren. Ved å koble to motstander, som vist i figur 11, vil man kunne styre strømtrekket ved å sette utgangen høy eller lav. Dette kan enkelt styres i programkoden og på denne måten kan koblingen begrenses til å trekke strøm kun i det øyeblikket man skal lese av batterinivået.

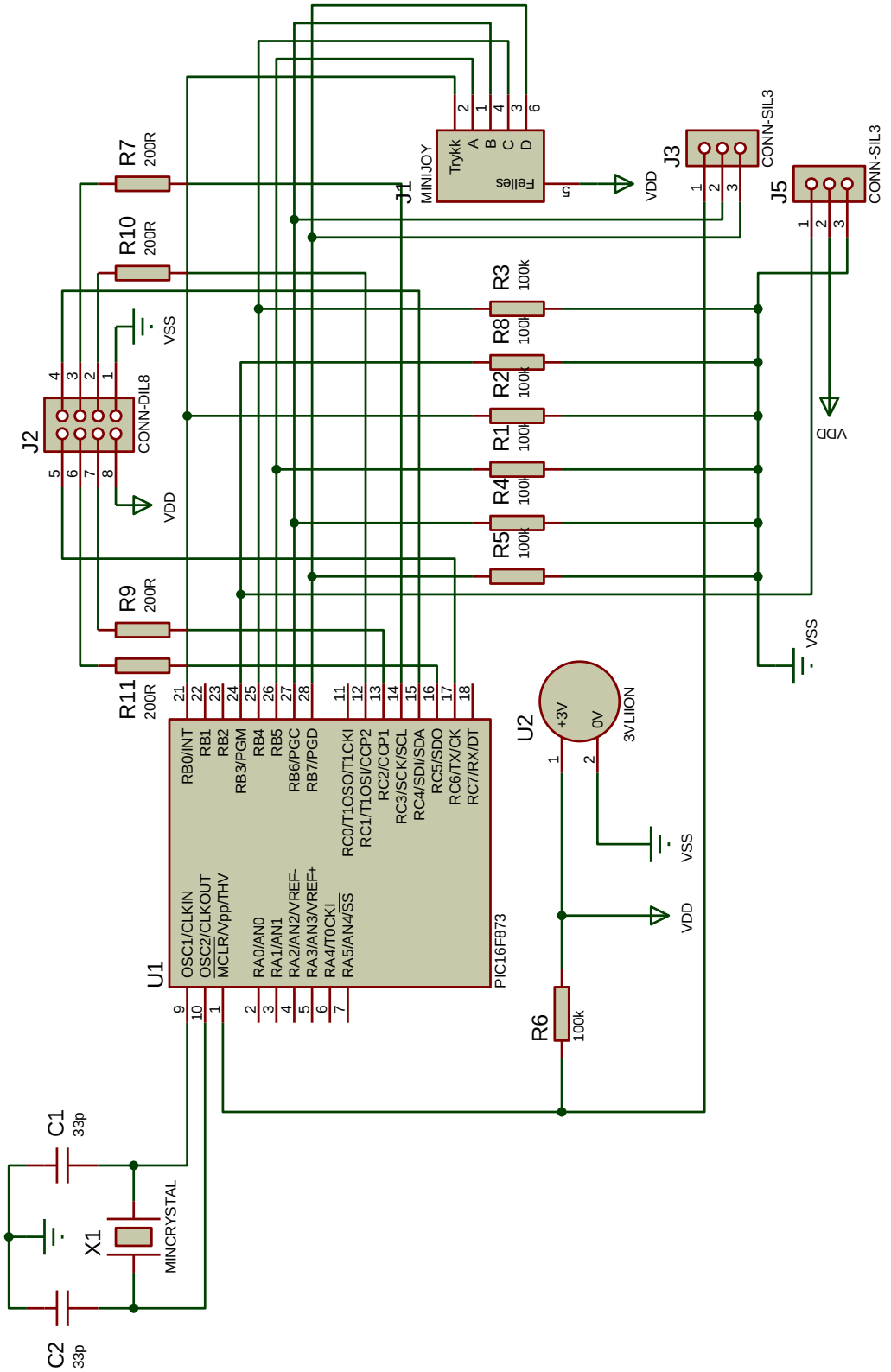
Det man imidlertid må passe på er at man ikke bruker en for høy motstandsverdi mellom batteri og A/D-pinnen (maksimalt 10kΩ), samt at verdien inn på A/D-pinnen er mellom 0V og nivået til referansespenningen (her 1.24V) som kommer inn på Vref+.

A/D-konverteringsfunksjonen er ikke i bruk på denne prototypen, men vi har her gitt et forslag til løsning som skal fungere. Noe av programkoden som skal til for å få denne funksjonen til å fungere er allerede på plass (se kode for fjernkontroll i vedlegg B).



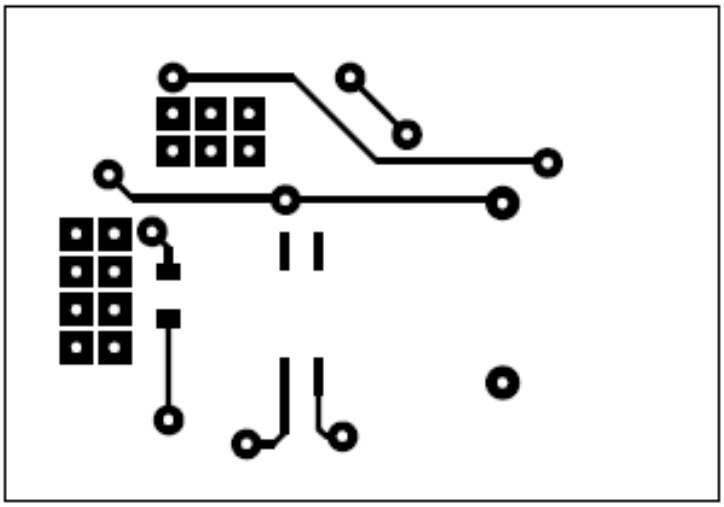
Figur 11: Forslag til spenningsmåling med lavt strømtrekk

Koblingsskjema for fjernkontrollen. Laget i ISIS 6 Professional. Løsningen for spenningsmåling med lavt strømtrekk er ikke implementert i disse tegningene.

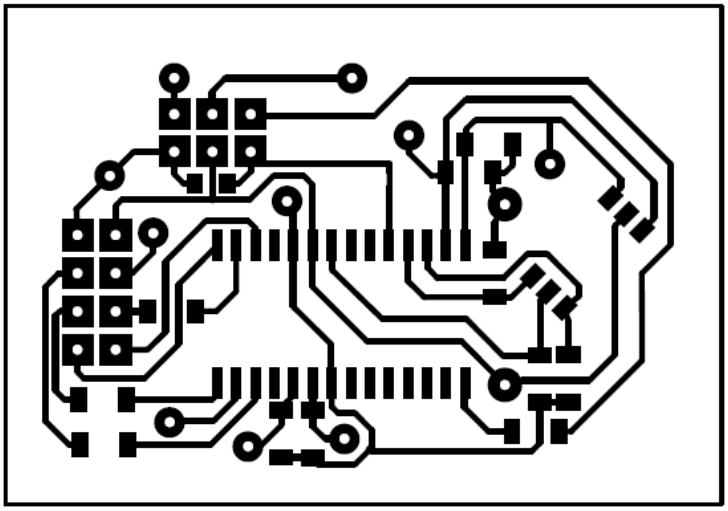


Figur 12: Kretsskjema for fjernkontrollen

Denne siden viser kretsutlegget for fjernkontrollen slik det ser ut på den gjeldende prototypen. Kretsutleggene er forstørret til det dobbelte av reell størrelse. Bildene på neste side viser størrelsen på tidligere utgaver av kretsutlegget i forhold til en gjennomsnittlig hånd og den valgte boksen. Det gjeldende kortet har beholdt de samme fysiske dimensjonene og har kun endringer i selve kretsen.

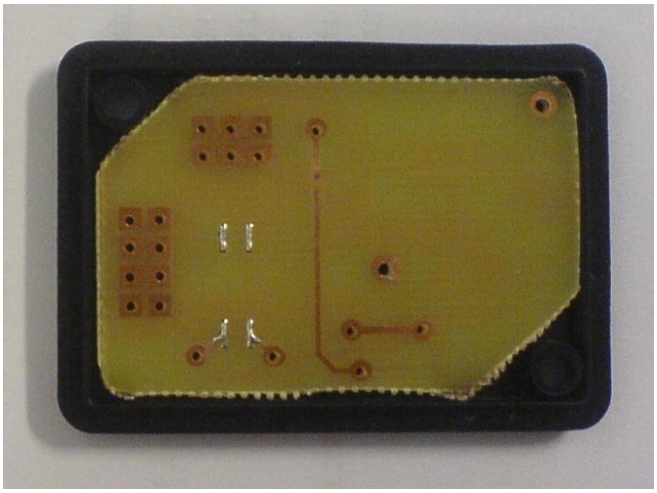


Figur 13: Oversiden av kretskortet (sett ovenfra)



Figur 14: Underside av kretskort (sett ovenfra)

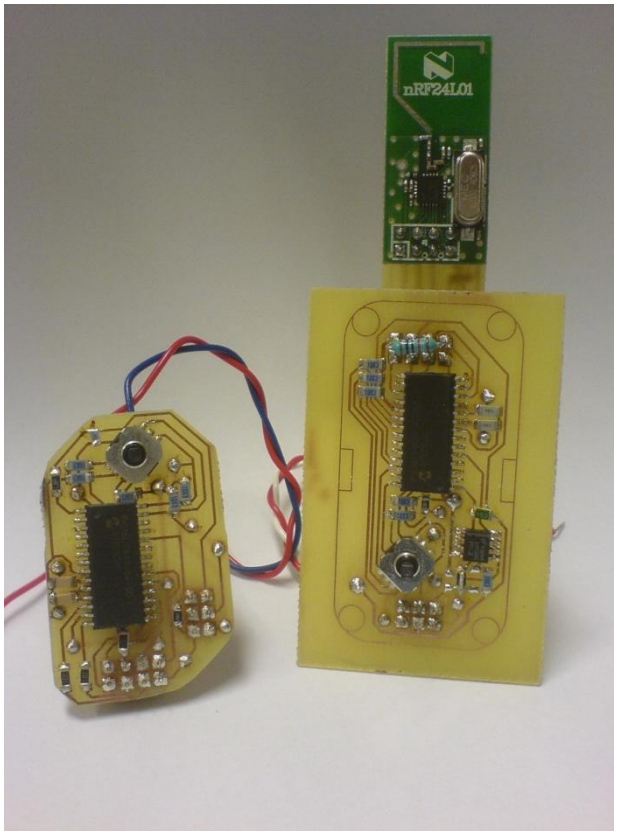
Figur 16: Første utgave av kretskortet til fjernkontrollen



Figur 15: Første utgave av kretskortet tilpasset boksen som skal brukes

Vi fant to aktuelle bokser til fjernkontrollen, og lagde kretsdesign til begge to. Nedenfor ser man de to versjonene av designet, hvor versjonen til høyre ikke har blitt kuttet og tilpasset boksen sin. Man kan se konturene som angir maksimal størrelse på kortet. Ledningene i bildet er brukt til spenningstilførsel under testing, før montering av batteriholderene.

Fjernkontrollen til høyre har også påmontert et datterkort som korrigerer pinnenummereringen på tilkoblingen til L01-kortet. Med riktig nummerering ville dette stått motstatt vei og holdt seg innenfor hovedkretsens yttermål.



Figur 17: Kretsdesign for de to boksene som har vært til vurdering

5.0 Utvikling av basestasjon

5.1 Komponentvalg

5.1.1 PICDEM.net

Siden det ikke var noen spesielle krav til den fysiske utformingen av basestasjonen sto vi relativt fritt i valg av løsning.

Det første reelle alternativet som ble presentert var å bruke Network Electronics (NE) sitt kontrollerkort. Dette er samme enhet som sitter i VikinX Sublime-routerene.

Fordeler med NE-kontrollerkortet:
- God og gratis kompilator (GNU C Compilator) kan brukes.
- Kompakt utforming.

Ulemper med NE-kontrollerkortet:
- Manglende utviklingsverktøy og erfaring med denne prosessortypen.
- Manglende kode for grunnfunksjonalitet.

Konklusjonen ble at det var for mange ukjente variable til at denne løsningen kunne velges.

Det andre alternativet var å bruke et av de mange test- og utviklingskortene fra Microchip. De to kandidatene var PICDEM.net og PICDEM.net 2. Ved starten av prosjektperioden var det ikke mulig å skaffe PICDEM.net 2 og dermed falt valget på PICDEM.net.

Fordeler med PICDEM.net:
- Grunnfunksjonene på plass i form av nettverksstøtte og nettsidetjener.
- Ekstrafunksjoner som display og serieportkommunikasjon for debugging på plass.
- Støttepersoner på HiG med erfaring fra PIC-programmering.

Ulemper med PICDEM.net:
- Ikke fullt så kompakt utforming som alternativet fra NE.

PICDEM.net ble valgt hovedsakelig på grunn av lett tilgjengelig kildekode for grunnfunksjonene. Den tilgjengelige koden er relativt godt kommentert og skaper et godt grunnlag for å utvikle og tilpasse de nødvendige funksjonene. Koden ble skrevet ved hjelp av MPlab v.7.50 og kompilert med studentutgaven av Microchip C18.

De viktigste spesifikasjonene for PICDEM.net i standardutførelse:
- PIC18F452 mikrokontroller klokket til 20MHz med 32kiB FLASH-minne og 256B EEPROM.
- TCP/IP grensesnitt gjennom en Realtek RTL8019AS kontrollerbrikke.
- Ekstern EEPROM på I2C-buss, 32kiB lagerplass.
- 16x2 LCD display uten bakgrunnsbelysning.
- Tilkobling for Microchip ICD2 debugging- og programmeringsenhet
- Prototypeområde på kretskortet for tilkobling av eksterne kretser.

Figur 18: PICDEM.net utviklingskort

5.1.2 PIC18F4620

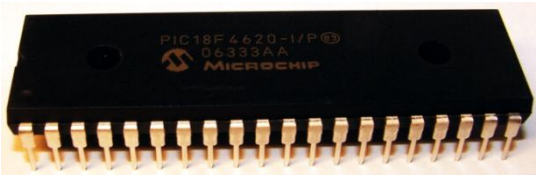
For å kunne gjennomføre prosjektet i henhold til planen ble PIC18F452 erstattet av PIC18F4620. Tabell 1 lister opp de viktigste forskjellene mellom de to prosessorene.

Tabell 1: Sammenligning av 18F4620 og 18F452

Prosessor	4620	452
Flash-minne	64kiB	32kiB
EEPROM	1024B	256B
RAM	3986B	1536B

Det dobbelt så store Flash-minnet er nødvendig fordi vi valgte å ikke bruke den eksterne EEPROM-brikken. Dette valget ble gjort for å lette utviklingen av SPI-grensesnittet, siden SPI og I2C bruker de samme pinnene på prosessoren.

Det fire ganger så store EEPROM-området er nødvendig for å kunne lagre et tilstrekkelig antall oppsett.



Figur 19: PIC18F4620

Arbeidsminnet var ikke noe stort problem og det hadde vært nok med samme mengde

minne som 452-utgaven hadde. Å ikke bli begrenset av arbeidsminnet var uansett en fordel under kodingen, selv om det selvfølgelig ble lagt vekt på å holde koden relativt kompakt.

5.1.3 LCD-display og bryterpanel

Innstillingene i basestasjonen gjøres ved hjelp av et femveis bryterpanel og et enkelt menysystem som vises i LCD-displayet. Lagring, endring og sletting av koblingsoppsett gjøres via nettsiden, men når dette er på plass kan enheten betjenes uten å ha tilgang på en datamaskin.

Figur 20: Bryterpanelet

Valg av bryterpanel falt på en Navimec-enhet fra danske mec. Valget ble gjort på bakgrunn av en god fysisk utforming med tydelige tilbakemeldinger på knappetrykk. LCD-displayet som satt på kortet ble brukt slik det var.

5.1.4 Trådløs kommunikasjon

Basestasjonen ble utstyrt med samme type rf-kort som i fjernkontrollen. Mikrokontrolleren kommuniserer med nRF24L01 over en SPI-buss som henholdsvis master og slave. Når rf-kortet mottar en ny datapakke fra fjernkontrollens rf-sender gir den et avbruddssignal (interrupt) til mikrokontrolleren som deretter henter ut de mottatte dataene.

5.2 Konstruksjon

5.2.1 Bryterpanel

For å få koblet til de ønskede ekstraenhetene var det nødvendig å koble ut noen av de originale tilbehørene på PICDEM.net-kortet. Som det fremgår av tabell 2 var det få ledige pinner på mikroprosessoren. Enhetene som ble fjernet var den eksterne EEPROM-brikken og spenningsdeleren som var tilkoblet RA0.

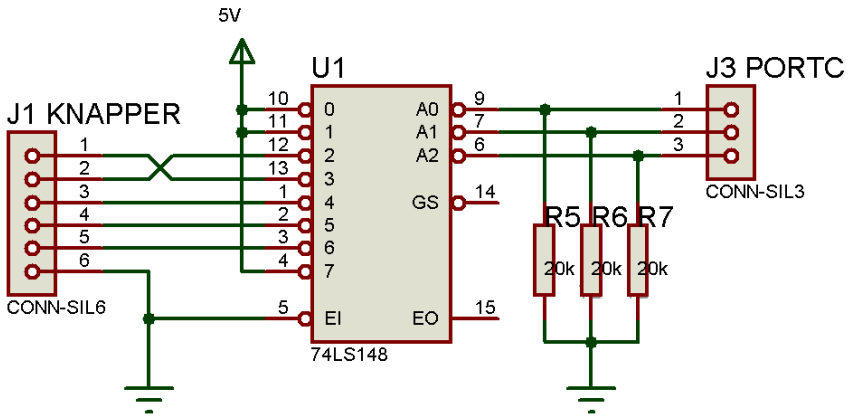
Tabell 2: Pinneoversikt på PIC18F4620

Pinne	Funksjon	Pinne	Funksjon
RA0	nRF24L01 interrupt	**RC3**	SPI klokkesignal
RA1	nRF24L01 chip select	**RC4**	SPI data inn
RA2	LED0	**RC5**	SPI data ut
RA3	LED1	**RC6**	TX USART (til RS232)
RA4	LED2	**RC7**	RX USART (til RS232)
RA5	LCD enable	**RD0**	Display og Ethernett databuss
RB0	RTL8019AS Adressebuss	**RD1**	Display og Ethernett databuss
RB1	RTL8019AS Adressebuss	**RD2**	Display og Ethernett databuss
RB2	RTL8019AS Adressebuss	**RD3**	Display og Ethernett databuss
RB3	RTL8019AS Adressebuss	**RD4**	Display og Ethernett databuss
RB4	RTL8019AS Adressebuss	**RD5**	Display og Ethernett databuss
RB5	RTL8019AS Adressebuss	**RD6**	Display og Ethernett databuss
RB6	RTL8019AS Adressebuss	**RD7**	Display og Ethernett databuss
RB7	RTL8019AS Adressebuss	**RE0**	RTL8019AS Les data
RC0	Bryterpanel-enkoder LSB	**RE1**	RTL8019AS Skriv data
RC1	Bryterpanel-enkoder	**RE2**	RTL8019AS Reset
RC2	Bryterpanel-enkoder MSB		

Bryterpanelet består av fem brytere men det er ikke nødvendig med mer enn tre bit for å kunne identifisere alle mulige tasteretninger. Det ble forutsatt at kun én knapp skulle trykkes om gangen. For å løse dette ble det brukt en enkel 8-3 enkoder fra Texas Instruments (74LS148). Denne kretsen foretar prioritert enkoding slik at det uansett kombinasjon av tastetrykk kun blir sendt én gyldig verdi til mikrokontrolleren.

Inngangene og utgangene i enkoderen opererer med invers logikk (lavt nivå for logisk 1). For å sørge for stabile signalnivåer på utgangen ble det montert 20kOhm pull-down motstander mellom hver linje og jord. Utgangene A2-A0 ble koblet til mikroprosessorens innganger RC2-RC0 (msb-lsb).

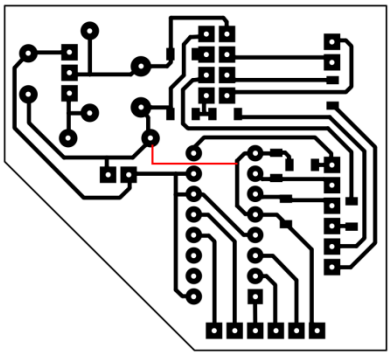
Knappene ble tilkoblet enkoderens innganger 2-6 mens inngangene 0, 1 og 7 ble koblet konstant høye. Med denne koblingen ble tallkombinasjonene på utgangen i området 1-5 for de fem retningene og 7 når ingen brytere hadde blitt trykket. I programmet sjekkes bryterpanelet ved å se om tallverdien på mikroprosessorens innganger er forskjellig fra 7 i hver runde av hovedløkka.



Figur 21: Koblingsskjema for enkoder

Pinne 1 og 2 på bryterpanelets tilkoblingspunkter er krysskoblet for å forenkle kretskortutlegget. Kretskortutlegget har rom for forbedringer siden det ble nødvendig å foreta en strapping, men fungerer ellers upåklagelig. Alle tilkoblinger til PICDEM.net med unntak av +5v og jord gjøres gjennom faste pinner slik at kortet er lett å koble til og fra.

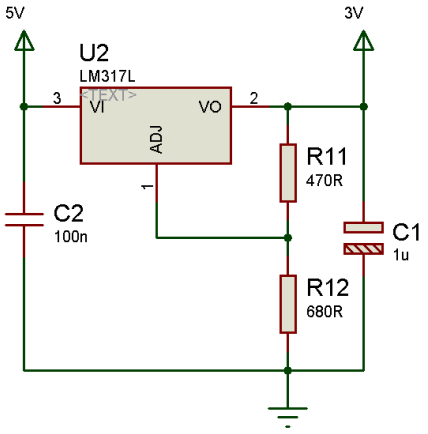
Den røde streken på printutlegget angir hvor det ble gjort en strapping for å fullføre kretsen.



Figur 22: Printutlegg for basestasjonens datterkort

5.2.2 Spenningsregulator

Siden PICDEM.net forsyner mikroprosessoren og de andre eksisterende enhetene på kortet med 5V drivspenning ble det nødvendig å konstruere en enkel reguleringskrets for å mate rf-kortet med en spenning på rundt 3V. For å gjøre det enklest og billigst mulig ble det valgt en LM317 spenningsregulator. Denne ble koblet opp i henhold til anbefalt oppkobling fra databladet med 100nF og 1uF avkoblingskondensatorer på henholdsvis inngang og utgang. Motstandene R1 og R2 ble valgt til 470Ohm og 680Ohm.



Figur 23: Koblingsskjema for spenningsregulator

Hvis vi ignorerer spenningsbidraget fra justeringsstrømmen blir utgangsspenningen beregnet til Vo = 1,25V * (1+R2/R1) ≈ 3,06V. Denne beregningen ble bekreftet med målinger. Det ble ikke brukt beskyttelsesdioder i kretsen.

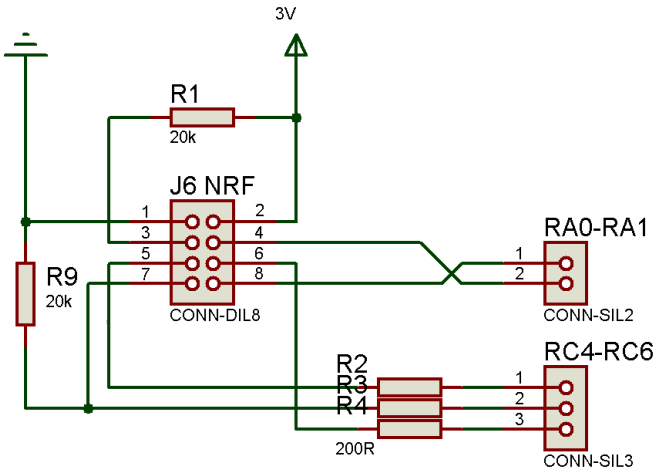
5.2.3 SPI-buss

Kommunikasjonen mellom nRF24L01 og mikroprosessoren foregår over en SPI-buss. Siden den synkront-serielle kommunikasjonsenheten i PIC18-prosessorene bruker de samme pinnene til SPI og I2C var det en stund usikkert hvordan dette skulle løses.

En løsning som ble vurdert var å programmere et eget SPI-grensesnitt som sørget for at den serielle datalinjen på pinne RC4 (SPI data inn og I2C datalinje) kun endret tilstand når klokkelinja lå lav. Siden I2C-enheter kun er aktive mellom START- og STOPP-tilstandene (signalisert ved henholdsvis negativ og positiv flanke på SDA-linja når klokkelinja har høyt nivå), kunne man på denne måten foretatt SPI-kommunikasjon uten å måtte gi slipp på I2C-bussen.

Denne løsningen ble forkastet da det ble klart at vi kunne bruke deler av Flash-minnet til å lagre de sammme dataene som i utgangspunktet lå lagret på den eksterne EEPROM-brikken. Siden denne brikken var den eneste enheten på PICDEM.net kortet som kommuniserte over I2C kunne vi dedikere den serielle kommunikasjonskontrolleren til SPI-modus og forenkle programmeringsarbeidet betraktelig.

Som det fremgår av Figur 24: Koblingsskjema for nRF24L01 på basestasjonen er rf-brikken koblet til mikroprosessoren via pinnene RA0 (IRQ), RA1 (CSN), RC4 (SCK), RC5 (SDI) og RC6(SDO). Det hadde helt klart vært ønskelig at også nRF24L01 kommuniserte over I2C for å forenkle konstruksjonen og spare pinner.

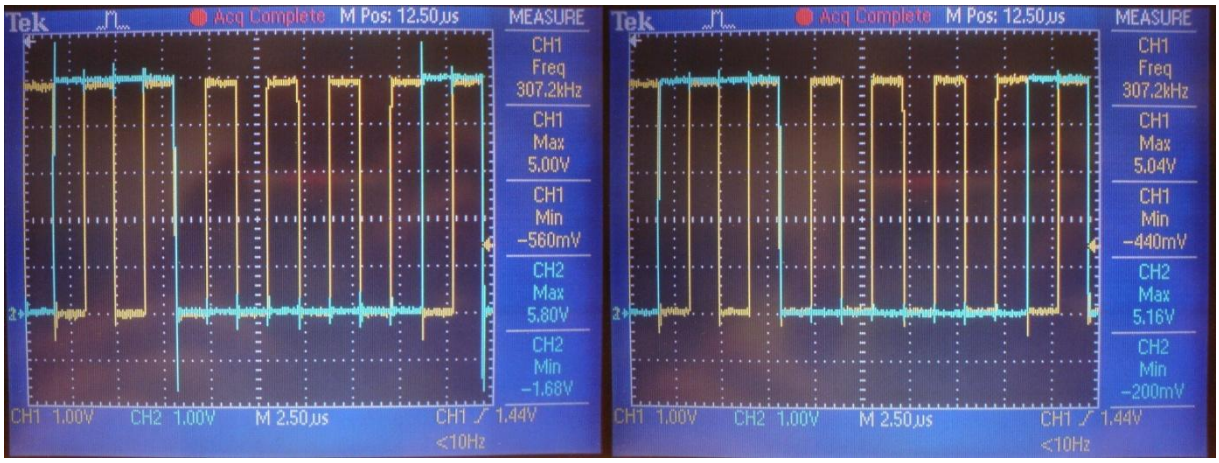


Figur 24: Koblingsskjema for nRF24L01 på basestasjonen

Tabell 3: Pinner på nRF24L01

Pinne	Navn
1	GND
2	RF_VDD
3	CE
4	CSN
5	SCK
6	MOSI
7	MISO
8	IRQ

Siden basestasjonen kun skulle fungere som mottager er CE-pinnen lagt høy via R1 til 3V. R2-R4 har verdien 200Ohm. Formålet med disse er å dempe oversvinget i kantene av firkantpulsene fra basestasjonen (CSN, MOSI og SCK). Denne verdien ble valgt etter litt prøving og en avveining av oversving i forhold til falltid.



Figur 25: SDO direkte koblet og med 200Ohm seriemotstand

6.0 Utstyr

Tabell 4: Oversikt over utstyr brukt fra elektrolaben

Navn	Type	Lab.nr
Tektronix TDS2002	Digitalt oscilloskop	EL 5006
Oltronix B200	Spenningsforsyning	SV 06001
Thurlby PL320	Spenningsforsyning	RE 06215 RE 06222
MPLAB ICD2	In-cicuit Debugger	EL 7025 EL 7017 EL 7003
FLUKE 45	Multimeter	RE 02361
Thurlby Thandar Instruments	Signalgenerator	EL 3000 EL 3001 EL 3002 EL 3005 EL 3008
nRF24L01	Utviklingssett	Ikke nummerert[1]
VikinX SL-V1616	Videovelger	Ikke nummerert[2]

Det ble blitt brukt mye datautstyr under denne oppgaven. Til programmering har vi brukt MPLAB v.7.50 sammen med MPLAB ICD2-enhetene nevnt i tabell 4.
For å teste kommunikasjonen mellom nRF24L01-enhetene benyttet vi oss av et program (nrf24l01ec) fra Nordic Semiconductor som fulgte med uviklingssettet.

Til å lage koblingsskjema og kretsutlegg har vi brukt henholdsvis ISIS 6 Professional og ARES 6 Professional.

Vi har også brukt videosvitsjen fra Network Electronics under utviklingen av basestasjonen for å sjekke at kommunikasjonen fungerte etter hensikten.

[1] Dette settet ble utlånt av Arne Wold
[2] Denne enheten er lånt fra Network Electronics i Sandefjord

7.0 Utførelse

7.1 Forklaring av programkoden til fjernkontrollen

Se vedlegg E, flytskjema for fjernkontrollprogrammet og vedlegg B, komplett kildekode for fjernkontrollen.

Programmet vårt starter med å gi brukeren mulighet til å velge et identifikasjonsnummer (av fem mulige) på fjernkontrollen (jfr. punkt 1 i flytskjemaet). Dette gjøres ved å holde joysticken i en av de 5 forskjellige retningene mens man resetter fjernkontrollen.

Dersom fjernkontrollen registrerer at joysticken holdes i en retning ved oppstart (punkt 2) vil programmet lete opp ID-nummer avhengig av hvilken retning som ble trykt og lagre dette på posisjon 0-4 i EEPROM før den tar det i bruk. De tilgjengelige identifikasjonsnumrene ligger forhåndslagret i mikrokontrollerens EEPROM på posisjon 5-29. Dersom joysticken ikke røres bruker fjernkontrollen den ID'en som ligger lagret på posisjon 0-4.

Tabell 5: Valg av fjernkontrollnummer etter retning på joystick

Retning	Fjernkontrollnr.
Forover	1
Bakover	2
Venstre	3
Høyre	4
Trykk	5

Når ID-nummeret er satt, setter fjernkontrollen seg i en sovemodus (SLEEP) for å spare strøm (se punkt 6). Her vil fjernkontrollen ligge til det blir gjort en bevegelse på joysticken som sender et interrupt til mikrokontrolleren. Denne avbrytelsen vekker mikrokontrolleren fra sovetilstanden (se punkt 7).

Når mikrokontrolleren har våknet og sjekket hva slags retning som ble trykt på joysticken sendes denne informasjonen til L01-kortet via SPI-bussen (se punkt 8). ID-nummeret sendes automatisk sammen med hver kommando fra fjernkontrollen til basestasjonen. Når dette er gjort legger mikrokontrolleren seg tilbake i SLEEP. Hvis L01-kortet ikke får en mottaksbekreftelse på den sendte kommandoen bruker den inntil fem forsøk med 4ms mellomrom før den gir opp. Dette skjer automatisk uten innspill fra mikokontrolleren.

Når kommandoen er bekreftet mottatt eller alle fem forsøk er brukt opp settes L01-kortet automatisk i standby-modus for å spare strøm.

For å resette den nåværende fjernkontrollen er man nødt til å åpne fjernkontrollboksen og koble til vår lille reset-jumper på programmeringspinnene (se Figur 26: Reset-jumper slik den ser ut innvendig). På denne modulen sitter det en 100ohm motstand som kobler $\overline{MCLR}$ til jord. Utformingen er slik at uansett hvordan man setter den på så kan man ikke gjøre feil.

Figur 26: Reset-jumper slik den ser ut innvendig

7.2 Forklaring av programkoden til basestasjonen

7.2.1 Grunnlag

Programmet som ble laget for basestasjonen er basert på Microchip sin HTTP Stack versjon 3.75. Denne versjonen av programvaren var i utgangspunktet beregnet for bruk på PICDEM.net 2. I starten av prosjektet forsøkte vi å bruke versjon 2.11 som fulgte med PICDEM.net, men hadde problemer med at den ikke lot seg kompilere med intakt TCP/IP-funksjonalitet. Etter å ha kontaktet Microchip fikk vi tilsendt en utgave av versjon 3.75 som var tilpasset for å fungere på vårt kort. Denne utgaven hadde redusert funksjonalitet grunnet den begrensede mengden arbeidsminne i den originale mikroprosessoren PIC18F452 (1536B) i forhold til PIC18F97J60 (3808B).

Siden vi uansett hadde bruk for en mikroprosessor med mer Flash-minne og større EEPROM fulgte vi anbefalingene fra Microchip og oppgraderte til en PIC18F4620. Denne prosessoren er fullstendig pinne- og kodekompatibel med PIC18F452 noe som gjorde selve oppgraderingsprosessen problemfri.

Vi har beholdt mye av den originale koden og lagt til eller tilpasset kode etter prosjektets behov. En stor del av prosjekttiden har gått med til å få oversikt over hvordan den originale koden fungerer og hva som kunne endres for å dekke oppgavens behov.

7.2.2 Virkemåte for hovedprogrammet

Microchip sin HTTP Stack er basert på samkjøring av flere separate prosesser som deler på prosesseringstiden og utfører sine oppgaver etter tur. Mange av funksjonene er programmert med et sett definerte tilstander som prosessene går gjennom etterhvert som de utfører sine oppgaver. Når en prosess har utført oppgavene som tilhører den aktuelle tilstanden oppdateres tilstanden før prosessen returnerer til hovedprogrammet. Neste gang hovedprogrammet overlater styringen til den aktuelle prosessen fortsetter den der den slapp, siden tilstanden ble oppdatert og lagret i slutten av forrige runde. På denne måten kan flere prosesser samarbeide tilsynelatende samtidig med relativt lite forsinkelse. Denne måten å håndtere flere samtidige oppgaver på kalles kooperativ multitasking.

Kooperativ multitasking har blitt brukt i flere operativsystemer til vanlige datamaskiner, som for eksempel System 7 til og med Mac OS 9 for Macintosh og Windows-utgaver før 95 og NT. Den mest aktuelle formen for multitasking i dagens operativsystemer kalles pre-emptive multitasking. Pre-emptive multitasking er mer effektiv med tanke på å passe på at hver enkelt prosess får sin tildelte andel av prosesseringsressursene. Hver prosess får utført sine prosesser så lenge det ikke er noen andre høyere prioriterte oppgaver som skal utføres. Pre-emptive multitasking er ikke det samme som interruptstyring, selv om de to tingene har mye til felles og kan eksistere i de samme systemene.

Siden prosessene og funksjonene som skal utføres av basestasjonen er relativt enkle og ingen av dem krever utstrakt bruk av prosesseringstid fungerer den kooperative multitaskingen hensiktsmessig. Innkommende data fra fjernkontroller eller nettverket blir mellomlagret i henholdsvis rf-brikken og nettverkskontrolleren slik at mikrokontrolleren slipper å overvåke og detaljstyre disse kontinuerlig for å hindre overflyt.

7.2.3 Initialisering av hovedprogrammet

Se vedlegg F, flytskjema for initialisering av basestasjonprogrammet.
Når basestasjonen tilkobles strømnettet starter initialiseringen umiddelbart. Første trinn utføres av InitializeBoard. Denne funksjonen setter opp inn- og utgangspinner i mikroprosessoren slik at den kan kommunisere med de tilkoblede enhetene på kortet. Vi har valgt å beholde våre egne initialiseringer utenfor denne funksjonen.

Displayet oppdateres med kortnavnet på basestasjonen, versjonsnummeret til programvaren og melding om at IP-adresse skaffes. Selve DHCP-prosessen blir ikke startet før programmet er i hovedløkken.

Tredje trinn initialiserer telleren som holder oversikt over forløpt tid. Til dette brukes Timer 0 i mikroprosessoren. Denne er nødvendig fordi flere av funksjonene i programmet er tidsavhengige.

Filsystemet (Microchip PIC File System) initialiseres i fjerde trinn. Siden vi bruker FLASH-minnet til å lagre nettsidene innebærer denne initialiseringen kun å resette antall åpne filforespørsler og statusflaggene.

Initappconfig gjør ikke mer enn å sette opp statusvariabelen som angir om DHCP er slått på eller ikke. Hvis vi hadde brukt den eksterne EEPROM-brikken ville denne funksjonen også lest de lagrede konfigurasjonsvariablene fra denne.

Hvis knappen som er tilkoblet pinne RB5 holdes inne ved oppstart av basestasjonen oppdateres displayet med en statusbeskjed. Deretter startes SetConfig som er et tekstterminalbasert grensesnitt for å endre diverse innstillinger for kortet. For å bruke dette må man koble seg til basestasjonen via RS232-porten og bruke et terminalemuleringsprogram. Dette grensesnittet kan brukes til å endre serienummeret (og dermed MAC-adressen) til kortet, slå av/på DHCP som standard, endre standard IP-adresser og laste opp nettsidene i binært format (forutsatt at den eksterne EEPROM-brikken brukes til lagring, noe som ikke lenger er tilfelle).

StackInit setter utgangspunktet for stacken til IDLE-tilstanden. Hvis DHCP/IP-gleaning er aktivert settes flagget som angir at hovedprogrammet starter i (IP)konfigureringsmodus. Deretter kjøres fire separate initialiseringsrutiner:

Tabell 6: Initialiseringer for Ethernet

Initialiseringer	Oppgaver
MACInit	Tømmer alle sendebufrene og setter opp konfigurasjonsregistrene i nettverkskontrolleren.
ARPInit	Setter utgangspunktet for ARP-prosessen til IDLE og tømmer ARP- og IP-adressebufrene.
UDPInit	Stenger alle UDP-porter slik at de blir markert som ledige.
TCPInit	Stenger alle TCP-porter og setter dem opp slik at de er klare til bruk.

SPIInit setter opp de nødvendige konfigurasjonsregistrene for å bruke den synkront-serielle kommunkasjonskontrolleren i mikroprosessoren til SPI-kommunikasjon. Når dette er gjort brukes SPI-bussen til å sette opp konfigurasjonsregistrene i nRF24L01.

HTTPInit går gjennom alle de tilgjengelige tilkoblingsportene og setter dem til å lytte etter TCP-pakker sendt til HTTP-porten (80). Hver tilkobling sin tilstandsmaskin blir satt til å starte i IDLE-tilstanden.

Når alle disse trinnene er gjennomført er den grunnleggende initialiseringen av kortet ferdig. For å sørge for våre nye funksjoner blir det gjort et par ekstra initialiseringer. Den første er å lese antallet lagrede oppsett fra intern EEPROM-posisjon 80. Hvis dette antallet er større enn 10 er minnet sannsynligvis tomt (siden minnet som standard blir fylt med 0xFF ved tømming) og posisjonen blir overskrevet med verdien 0. De resterende initialiseringstrinnene foretar nullstilling av viktige variable og setting av pinne RC0-RC2 som innganger for å kunne ta i mot knappetrykk fra styrepanelet.

7.2.4 Hovedprogrammets forløp

Se vedlegg G, flytskjema for hovedprogrammets forløp.
Hovedløkka starter med å la StackTask utføre sine oppgaver. Dette innebærer å sjekke innkommende pakker fra nettverket for å avgjøre hva slags type de er for deretter å sette i gang de nødvendige prosessene for å behandle dem. Hvis pakkene er noe annet enn MAC-, ARP-, IP-, TCP- eller UDP-forespørsler forkastes pakkene siden programvaren ikke har noen forutsetninger for å håndtere andre typer enn de den har eksplisitte funksjoner for.

Etter at StackTask har gjort sitt tar HTTPServer-funksjonen over. Denne går gjennom hver av de åpne tilkoblingene og utfører de nødvendige handlingene. Dette kan innebære å levere filer direkte fra lageret eller å la hovedprogrammet gjøre enkelte beregninger før dataene sendes ut. Variable som trenger prosessering er indikert i .cgi filene med et prosenttegn fulgt av et 8bit tall i heksadesimalt format. Når HTTPProcess-funksjonen oppdager en slik variabel i de ønskede utdataene lar den funksjonen HTTPGetVar i hovedprogrammet ta over prosesseringen før den sender ut de ferdigbehandlede dataene. Alle dynamisk oppdaterte data på nettsiden håndteres på denne måten.

DiscoveryTask tar seg av sending av NetBiosnavn og MAC-adresse i tekstformat til klienter som etterspør dette gjennom en broadcastpakke adressert til UDP-port 30303. Forespørselen kan sendes og svaret vises ved hjelp av programmet MCHPDetect.exe og er nyttig for å finne adressene til eventuelle ukjente Microchip HTTP Stack-baserte enheter på nettverket. Dette programmet er den del av Microchip HTTP Stack v.3.75.

NaviSjekk foretar en enkel sjekk av bryterpanelets status og håndterer knappetrykket i henhold til flytskjemaet for NaviSjekk-funksjonen (se vedlegg H). For å muliggjøre hurtig endring av IP-adresser er denne funksjonen laget slik at man etter å ha holdt inne en knapp i et halvt sekund får en repetering på 10 trykk i sekundet. Hvis NaviSjekk konkluderer med at det har blitt foretatt et nytt knappetrykk eller at en av knappene repeteres kalles VisMeny for å oppdatere menysystemet i displayet.

Hvis menysystemet har blitt brukt til å velge et oppsett har det også satt flagget Tilstand.LastOppsett. Hvis dette er tilfelle leses det valgte oppsettet fra EEPROM til arbeidsminnet og holdes i arrayen Kommando. Siden vi har mer enn doblet minnekapasiteten etter byttet av mikroprosessor er ikke dette minneforbruket problematisk.

Hvis rf-brikken har mottatt nye data signaliseres dette ved at nivået på inngang RA0 legges lavt. Om man ikke har valgt et oppsett leses dataene av funksjonen Les_nRF uten noen videre behandling. I motsatt tilfelle kalles SendVX som tolker det mottatte fjernkontrollnummeret og retningen i henhold til det valgte oppsettet og fyller variablene VXKommando og VXIPadr med data som kan brukes til å kommandere VikinX-routeren(e). Når dette er klart settes flagget Tilstand.Send for å indikere at en ny kommando er klar for sending til en VikinX-router.

Hvis Tilstand.Send flagget er satt kalles VikinX-klienten ved hver gjennomkjøring av hovedløkka frem til flagget blir fjernet som følge av gjennomført kommandosending. Se ellers kildekoden i vedlegg L (VikinX.c).

Siste oppgave i programmets hovedløkke er å sjekke om IP-adressen har blitt fornyet som følge av utløpt låneperiode eller ved første gangs tildeling. Hvis dette er tilfelle kalles AnnounceIP som sender ut en melding på UDP-port 30303 med enhetens NetBios-navn, MAC-adresse og meldingen "DHCP/Power event occured". Programmet MCHPDetect.exe lytter etter og viser eventuelle slike pakker. Den nye IP-adressen vises også i displayet og hos eventuelle tilkoblede terminaler på RS232-porten.

7.3 Beskrivelse av funksjonene

7.3.1 NaviSjekk

void NaviSjekk(void)
Se vedlegg K (utdrag fra main.c) og vedlegg H, flytskjema for NaviSjekk-funksjonen.

Tabell 7: Variable brukt i NaviSjekk

Navn	Type	Innhold	Verdi
Tilstand.KnappSjekk	byte	Status for navisjekk funksjonen	0-2
Tilstand.NaviKnapp	byte	Knappetrykk	0, 2 - 6
Tilstand.GjentaKnapp	byte	Kopi av naviknapp	0, 2 - 6
KnappHold	timer	Tid mellom trykk og repetering	
KnappGjenta	timer	Tid mellom hver repetering	

Første trinn er å lese av port C0-C2. Enkoderen på datterkortet gir ut tallverdien 7 hvis det ikke er noen knapper som har blitt trykt inn. Når tallverdien er lest og tilpasset lagres denne i variabelen Tilstand.NaviKnapp. Verdien null indikerer nå at ingen knapp har blitt trykket.

Funksjonens tilstand lagres i variabelen Tilstand.KnappSjekk. De forskjellige tilstandene er listet opp i tabell 8.

Tabell 8: Tilstandene til Tilstand.KnappSjekk

Tilstand	Betydning
0	Funksjonen har ikke lest av bryterpanelet.
1	Funksjonen har lest av bryterpanelet.
2	Bryterpanelets knapp har blitt holdt inne i over 0,5s.

Når en knapp på bryterpanelet trykkes inn får Tilstand.NaviKnapp en verdi forskjellig fra null. Det blir også lagret unna en kopi av retningen i Tilstand.GjentaKnapp. Tilstand.KnappSjekk oppdateres slik at neste gang hovedprogrammet kaller NaviSjekk-funksjonen blir ikke verdien i Tilstand.NaviKnapp oppdatert. Hvis dette ikke hadde blitt gjort hadde manøvreringen av menysystemet forutsatt at brukeren hadde overmenneskelige reaksjonsevner.

Hvis knappen slippes settes Tilstand.KnappSjekk tilbake til null slik at funksjonen er klar til å lese inn en ny retning når denne kommer.

Hvis retningsverdien er uendret (knappen har blitt holdt inne) i over et halvt sekund gjentas knappetrykket og en ny repeteringstimer startes. Denne sørger for å gjenta knappetrykket for hvert tiendedels sekund frem til knappen slippes.

7.3.2 VisMeny

void VisMeny(void)
Se vedlegg K for kildekoden til denne funksjonen. Vedlegg J er en oversikt over hvordan menyene henger sammen.

Variable brukt i denne funksjonen:

Tabell 9: Variable brukt i VidMeny

Navn	Type	Innhold	Verdi
LCDPos	byte	Startposisjon i displayet	0 - 31
Tilstand.NaviKnapp	byte	Knappetrykk på bryterpanelet	0, 2 - 6
HMeny	byte	Hvilken hovedmeny som er valgt	0 - 4
UMeny	byte	Hvilken undermeny som er valgt	0 – 7
FjkKnapp	byte	Retning fra fjernkontrollen	1 - 5
FjkNr	byte	Nummer på fjernkontroll som skal vises	1 – 5
VelgOppsett	byte	Brukes til å bla gjennom oppsettene	1 - 10
Tilstand.ValgtOppsett	byte	Nummeret til valgt oppsett	1 - 10
Tilstand.LastOppsett	byte	Indikerer at et nytt oppsett skal lastes	0 - 1
Tmp	byte	Mellomlagring av data	
HjelpeSiffer	byte	Hjelpevariabel ved endring av IP-adresse	0 - 3
Posisjon	int	Leseposisjon i EEPROM	0 - 1023
LCDText	array	Tekst som skal vises i displayet	
TempTekst	array	Mellomlagring av tekst før visning	
RetnPeker	peker	Retningen fra fjernkontrollen i klartekst	
TekstPeker	peker	Tekst som skal vises i displayet	
IPPeker	peker	IP-adresse som skal vises eller endres	

Denne funksjonen kalles når det har skjedd noe som forutsetter en oppdatering av displayet. Det kan være et trykk på basestasjonens bryterpanel, at det har blitt mottatt en ny kommando fra fjernkontrollen (forutsetter at menysystemet viser statusmenyen) eller at det valgte oppsettet har blitt slettet ved hjelp av nettsiden.

Når man har slått av DHCP kan IP-adressene i Nettverksmenyen endres ved å velge den ønskede IP-adressen med piltastene og bekrefte med OK. Menyen er nå i EndreIP-modus (indikert ved at variabelen EndresIP er satt til 1) og trykk på piltastene vil ikke bytte menybilde før man trykker OK for å avslutte endringen. Piltastene opp og ned brukes til å endre tallverdi mellom 0 og 255, mens piltast venstre og høyre brukes til å hoppe mellom de fire tallene i IP-adressen. Endringen starter alltid på det bakerste tallet i IP-adressen.

Den aktive IP-adressen er til enhver tid lik den viste IP-adressen og det er derfor anbefalt å ikke koble basestasjonen til nettverket før man har satt IP-adressen for å unngå mulige adressekonflikter med andre enheter. Ved oppstart av basestasjonen er IP-adressen satt til 0.0.0.0 før en gyldig adresse mottas via DHCP eller settes manuelt.

7.3.3 HTTPExecCmd

void HTTPExecCmd(BYTE** argv, BYTE argc)
Se vedlegg K (utdrag fra main.c) for kildekoden til denne funksjonen og eventuelt http.h i vedlegg A for tilhørende headerfil. Se også index.html i vedlegg A for å se koden nettsiden bruker til å kalle denne funksjonen. Denne funksjonen er en del av den originale koden i Microchip HTTP Stack 3.75 men har blitt tilpasset for å utføre oppgavene som er nødvendige i vårt system.

Tabell 10: Variable brukt i HTTPExecCmd

Navn	Type	Innhold	Verdi
command	byte	Kommandokode fra nettsiden	0 – 4
var	byte	Styreknapp på nettsiden (ubrukt)	0 – 9
i	byte	Hjelpevariabel til løkker	
VisFjktnr	byte	Fjernkontroll som ønskes vist	0 – 4
VisOppsettnr	byte	Oppsettet som ønskes vist	1 – 10
Feil	BOOL	Angir suksess eller fiasko	
NyKommando	array	Ny kommando som skal lagres	
TempTekst	array	Navn på oppsett som skal lagres eller vises	

Denne funksjonen utfører forskjellige handlinger avhengig av argumentene man kaller den med. Det første argumentet angir hva slags type handling som skal utføres. Dette argumentet står på plassen som vanligvis brukes til å indikere hva slags fil nettleseren ønsker fra tjeneren (basestasjonen). De påfølgende kommandoene angir nærmere hva som ønskes utført eller lagret.

Tabell 11: Kommandokoder fra nettside

Kode	Betydning
0	Styreknapp (virtuelle frontpanel- eller fjernkontrollknapper)
1	Lagre nytt oppsett eller endre et lagret oppsett
2	Klargjør for visning av et lagret oppsett
3	Lagre informasjon om en VikinX-router
4	Slett et lagret oppsett

0. Virtuelle knapper
Denne funksjonen har blitt brukt under utviklingen av basestasjonen til å teste menysystemet før løsningen for det fysiske bryterpanelet og fjernkontrollen var på plass. Nettsiden inkluderer ikke disse knappene i den gjeldende utgaven.

1. Lagring av oppsett
Et eksempel på en anmodning fra nettsiden når et oppsett ønskes lagret kan være ”1?TestOppsett&4=2&101=0&2=8&0”.

Det første argumentet indikerer at det ønskes å lagre eller endre et lagret oppsett. Etter spørsmålstegnet følger navnet på det aktuelle oppsettet. Deretter følger koden for fjernkontrollnummeret og retningen på joysticken, i dette tilfellet fjernkontroll nummer 5 og retning venstre.

Nivånummeret kommer som neste tall, her nivå 101, og deretter hvilken av de to VikinX-routerene skal styres, i dette tilfellet den første. Tallet to indikerer at det skal kobles tre signaler med denne kommandoen og de to siste tallene indikerer at kommandoen gjelder fra og med inngang 9 og utgang 1.

Sagt i klartekst betyr dette: "Når TestOppsett er valgt skal et trykk bakover på fjernkontroll nummer fem kommandere alle routere i nivå 101 (som kan nås fra den angitte VikinX-routeren) til å koble inngangene 9-11 mot utgangene 1-3."

Lagring eller endring av et oppsett utføres kun hvis de inntastede verdiene er innenfor gyldige intervaller. Hvis oppsettet ikke finnes fra før lagres det kun hvis det er plass til flere oppsett.

Tabell 12: Gyldige intervaller for oppsettdata

Variabel	Gyldig intervall
Navnlengde	1 – 16
Inngang/utgang	1 – 64
Nivå	1 – 200

Hvis man skriver inn et navn med mer enn 16 bokstaver blir kun de første 16 tegnene lagret. Mellomrom bør indikeres med en understrek, da det nåværende programmet i basestasjonen mangler kodesnutten som skal til for å tolke om ”%20” til riktig ASCII-verdi.

2. Vise et lagret oppsett
Et eksempel på en anmodning fra nettsiden når de lagrede kommandoene for en fjernkontroll i et gitt oppsett ønskes vist kan være ”2?TestOppsett&2”. Her ønskes kommandoene for fjernkontroll tre i TestOppsett vist.

Hvis det er noen lagrede oppsett i basestasjonen gjøres fjernkontrollnummeret om fra ASCII-kode til en ren tallverdi og funksjonen FinnOppsett kalles for å lete frem nummeret til det ønskede oppsettet. Selve visningen blir utført når nettsiden et halvt sekund senere ber om en oppdatering av VisData.cgi.

3. Lagre informasjon om en VikinX-router
Et eksempel på en anmodning fra nettsiden når ip-adresse og navn på en VikinX-router ønskes endret kan være ”3?1&192=168&0=5&Nyttnavn”. Her ønskes ip-adressen for VikinX-router nummer to endret til 192.168.0.5 og navnet endret til "Nyttnavn". Hvis man ikke skriver inn et navn blir ikke navnet endret.

4. Slette et lagret oppsett
Et eksempel på en anmodning fra nettsiden når et oppsett ønskes slettet kan være ”4?PoliceSquad”. Programmet kaller funksjonen FinnOppsett for å lete etter et oppsett

med det angitte navnet. Hvis dette finnes blant de lagrede oppsettene blir funksjonen SlettOppsett kalt.

7.3.4 HTTPGetVar

word HTTPGetVar(BYTE var, WORD, ref, BYTE* val)
Se vedlegg K (utdrag fra main.c) for kildekode og eventuelt http.h i vedlegg A for tilhørende headerfil. Se også index.html og cgi-filer i vedlegg A for koden til tekstområdene som inneholder de etterspurte variabelkodene. En oversikt over de aktuelle variabelkodene er å finne i Tabell 14: Variabelnavn og koder.

Når nettsiden ber om å få tilsendt en fil som inneholder disse variabelkodene blir hver av variabelkodene prosessert av HTTPGetVar før den tilhørende verdien returneres til nettleseren.

Tabell 13: Variable brukt i HTTPGetVar

Navn	Type	Innhold	Verdi
var	byte	Variabelkoden	0 – 255
i	byte	Tellervariabel til løkker	
tmp	Byte	Mellomlagring av data fra EEPROM	
ref	word	Hvor langt man er i prosesssen	
val	peker	Peker til returVariable	

Tabell 14: Variabelnavn og koder i cgi-filene

Variabelnavn	Kode	Variabelnavn	Kode	Variabelnavn	Kode
VAR_LED0	00	**VAR_NIV4**	0A	**VAR_OS1**	34
VAR_STACK	01	**VAR_VX4**	0F	**VAR_OS2**	35
VAR_STACK_DATE	02	**VAR_ANT4**	14	**VAR_OS3**	36
VAR_ANTOS	03	**VAR_INN4**	19	**VAR_OS4**	37
VAR_NIV1	07	**VAR_UT4**	1E	**VAR_OS5**	38
VAR_VX1	0C	**VAR_UTANT4**	2B	**VAR_OS6**	39
VAR_ANT1	11	**VAR_NIV5**	0B	**VAR_OS7**	3A
VAR_INN1	16	**VAR_VX5**	10	**VAR_OS8**	3B
VAR_UT1	1B	**VAR_ANT5**	15	**VAR_OS9**	3C
VAR_UTANT1	28	**VAR_INN5**	1A	**VAR_OS10**	3D
VAR_NIV2	08	**VAR_UT5**	1F	**CMD_DOWN**	0
VAR_VX2	0D	**VAR_UTANT5**	2C	**CMD_UP**	1
VAR_ANT2	12	**VAR_VX1IP1**	20	**CMD_RIGHT**	2
VAR_INN2	17	**VAR_VX1IP2**	21	**CMD_LEFT**	3
VAR_UT2	1C	**VAR_VX1IP3**	22	**CMD_OK**	4
VAR_UTANT2	29	**VAR_VX1IP4**	23	**CMD_FJKOPP**	5
VAR_NIV3	09	**VAR_VX2IP1**	24	**CMD_FJNED**	6
VAR_VX3	0E	**VAR_VX2IP2**	25	**CMD_FJVEN**	7
VAR_ANT3	13	**VAR_VX2IP3**	26	**CMD_FJHOY**	8
VAR_INN3	18	**VAR_VX2IP4**	27	**CMD_FJINN**	9
VAR_UT3	1D	**VAR_VXNAVN1**	30	**LEDIG**	FF
VAR_UTANT3	2A	**VAR_VXNAVN2**	31		

Variabelen ref er nødvendig fordi HTTP-tjeneren kun returnerer én byte om gangen og derfor må kalles flere ganger for å få levert større variable. Ved første kall inneholder ref verdien HTTP_START_OF_VAR (definert som 0x0000) og når programmet er ferdig med å returnere det den skal setter den ref = HTTP_END_OF_VAR (definert som 0xFFFF). Den største strenglengden som kan returneres per variabel er derfor 65535 tegn som i vårt tilfelle er tilstrekkelig med god margin.

Når programmet blir bedt om innholdet i variabelkodene for de forskjellige delene av oppsettene må disse leses ut fra EEPROM og sorteres ut slik at de kan presenteres på nettsiden. Beskrivelse av lagringsformatet finnes i avsnittet for funksjonen LagreOppsett.

Javascriptfunksjonene i koden til nettsiden sørger for automatisk oppdatering av listen over lagrede oppsett og VikinX-routere ved første gangs lasting av nettsiden og ved endring av de lagrede opplysningene. Ved visning av lagrede kommandoer vil uspesifiserte retninger og oppsett som ikke finnes vises som bindestreker. Se vedlegg P, bilde av nettsiden.

7.3.5 SendVX

void SendVX(BYTE fjretn, BYTE fjnr, BYTE* kommliste)
Se vedlegg L (VikinX.c) og VikinX.h i vedlegg A.

Denne kommandoen tar utgangspunkt i mottatt retningskommando fra fjernkontroll, nummer på fjernkontrollen som sendte kommandoen og det valgte oppsettet. Ut i fra disse opplysningene setter den sammen de to variablene VXKommando og VXIPadr slik at TCP-klienten kan settes i gang med å kommandere VikinX-routeren.

Tabell 15: Variable brukt i SendVX

Navn	Type	Innhold	Verdi
fjretn	byte	Retning på joystick	0 – 5
fjnr	byte	Fjernkontrollens nummer	1 – 5
kommliste	peker	Peker til lastet kommandoliste	
lesepos	int	Leseposisjon i EEPROM	
i	byte	Tellervariabel for IP-lesing	0 – 4
n	byte	Tellervariabel for kommandolesing	0 – 72
VXKommando	array	Kommandosett som skal sendes	
VXIPadr	array	IP-adressen til VikinX-routeren	

Kommandolista inneholder alle de lagrede kommandoene for det aktive oppsettet. Når SendVX kalles regner den seg frem til riktig leseposisjon i kommandolista ut i fra angitt fjernkontrollnummer og joystickretning. Deretter leses og behandles dataene fra kommandolista slik at nivånummer, inngangsnummer, utgangsnummer, antall signaler og ip-adressen til aktuell VikinX-router finnes. Lagringsformatet i kommandolista er en kopi av lagringsformatet i EEPROM og står beskrevet i avsnittet om funksjonen LagreOppsett.

Når SendVX har gjort sitt har VXKommando og VXIPadr følgende informasjon:

Tabell 16: Innhold i VXKommando og VXIPadr

Posisjon	VXKommando	VXIPadr
0	Nivånummer (1-200)	Første tall i IP-adressen
1	Antall signaler (0-7)	Andre tall i IP-adressen
2	Inngangsnummer (0-63)	Tredje tall i IP-adressen
3	Utgangsnummmer (0-63)	Fjerde tall i IP-adressen

7.3.6 VikinXTCPClient

BOOL VikinXTCPClient(void)
Se vedlegg L (VikinX.c) og VikinX.h i vedlegg A.

Denne funksjonen er en tilpasset utgave av GenericTCPClient fra Microchip HTTP Stack 3.75. Den sørger for å gjennomføre kommunikasjonen med VikinX-routeren som er siste del i styringsprosessen. Funksjonen kalles når SendVX er utført og et ferdig kommandosett er klart til sending. Den er bygd opp med tilstander for hvert trinn i prosessen slik at den kan kalles flere ganger frem til jobben er utført. Dette er nødvendig siden mottak av pakker fra nettverket (og dermed svarene fra VikinX-routeren) håndteres av funksjonen StackTask i starten av hovedløkka.

Dokumentasjonen fra Network Electronics (vedlegg Q på CD, Control Protocol for VikinX Modular Routers) beskriver hvilke kommandoer vi kan sende for å styre VikinX-routerene. Kommandoene sendes som ren tekst og er enkle i oppbygning.

Tabell 17: Variable brukt i VikinXTCPClient

Navn	Type	Innhold
i, j	byte	Tellervariable for løkker
TallTekst	array	Tall i ASCII-format

Se vedlegg I, flytskjema for VikinXTCPClient.
Første gang funksjonen kalles begynner den med å finne MAC-adressen til VikinX-routeren som skal motta kommandoen. Så snart denne er funnet settes det opp en TCP-port som vil bli brukt til å sende dataene. Når denne er klar begynner prosessen med å bygge opp tekststrengen kommandoen skal bestå av. Det vil alltid være minst ett kontaktpar (én inngang og én utgang) som skal kobles. Trinn 5 utføres kun hvis det skal kobles to eller flere signaler.

Hvis det er ønskelig å koble et signal med tre kontakter (f.eks komponent-video) fra inngangene 3-5 til utgangene 7-9 i nivå 42 vil kommandoen være ”x l42 3 7 4 8 5 9”

Det første tegnet, x, indikerer at kommandoen "crosspoint take" ønskes utført (kobling av inngang til utgang). Nivånummeret følger etter et mellomrom og bokstaven l for

level. Deretter følger parvis inngang og utgang som skal kobles sammen. Inngang 3 skal kobles til utgang 7, inngang 4 til utgang 8 og inngang 5 til utgang 9.

Når kommandooppbygningen er ferdig sendes kommandoen. Eventuelle svar fra VikinX-enheten blir ikke behandlet i den nåværende versjonen av programmet. Det er med andre ord ingen automatisk kontroll på om koblingen ble utført. Testingen som har blitt utført har ikke kunnet fremprovosere noen feil og eventuelle problemer er umiddelbart åpenbare for brukeren (siden bilde/lydkilden ikke skifter).

VikinX-routerene er laget slik at de kun kan ha én åpen nettverksforbindelse om gangen og derfor automatisk kobler fra ved lengre tids inaktivitet (rundt 10s) slik at en inaktiv klient ikke skal kunne blokkere andre klienter som ønsker å kommunisere med routeren. Vårt klientprogram lar tilkoblingen stå åpen etter at kommandoen er sendt slik at hvis det kommer en ny kommando opprettes det ikke noen ny tilkobling med mindre VikinX-routeren har stengt forbindelsen. Siden en stengt forbindelse ikke oppdages før det er gjort et forsøk på å sende kommandoen én gang reagerer systemet raskest på kommandoer som kommer tett på den foregående. Forsinkelsen som oppstår når forbindelsen må opprettes på nytt er likevel veldig liten.

7.3.7 SPIInit

void SPIInit (void)
Se vedlegg O (SPInRF24.c) og SPInRF24.h i vedlegg A.

Denne funksjonen tar for seg initialisering av SPI-grensesnittet i mikrokontrolleren og oppsett av registre i nRF24L01-brikken i henhold til systemets behov.

Første del i initialiseringen var å sette RC3 (SCK) som utgang, RC4 (SDI) som inngang og RC5 (SDO) som utgang. I tillegg ble RA0 (nRF_IRQ) definert som inngang og RA1 (nRF_CSN) som utgang. For å sette opp SPI-grensesnittet ble SSPSTAT satt til 0xC0 og SSPCON1 til 0x22. Dette fører til at SPI-grensesnittet blir skrudd på med datasampling i slutten av data-ut perioden, dataoverføring på fallende klokkeflanke, PIC i master modus, lav idle-tilstand for klokke og klokkefrekvens 307,2kHz (1/64 av mikrokontrollerens frekvens). Dette tilsvarer en overføringshastighet på drøyt 38kiB per sekund som er mer enn nok for basestasjonens behov.

Når SPI-grensesnittet er klart begynner oppsettet av registrene i nRF24L01. For å skrive til registrene sendes først en kommando som angir at man ønsker å skrive til et register og deretter verdien som skal skrives til registeret.

Tabell 18: Kommandoer brukt til nRF24L01

Kommando	Format	Beskrivelse
Les register	000AAAAA	AAAAA angir adressen til registeret
Skriv til register	001AAAAA	AAAAA angir adressen til registeret
Les nyttelast	01100001	Nyttelasten klokkes ut ved neste forespørsel
Skriv til nyttelast	10100000	Data som skal skrives leses fra neste sending

Hver gang en ny kommando skal sendes til nRF-brikken må CSN-pinnen gå fra høy til lav og holdes der til kommunikasjonen er fullført. Data tilhørende en kommando må leses eller sendes før CSN-pinnen går høy igjen.

Basestasjonens nRF24L01 ble satt opp som hovedmottager (primary rx) og det ble åpnet for at den skal kunne indikere nye pakker ved å bruk av RX_DR (data ready) interrupt. Siden systemet bruker Auto-Acknowledge måtte det tillates CRC-sjekking.

Auto-Acknowledge sørger for at eventuelle pakker som mottas hos mottager automatisk fører til at en bekreftelse blir sendt til avsenderen. Hvis senderen ikke mottar denne bekreftelsen kan den forsøke å sende pakka på nytt. Antallet forsøk og ventetiden mellom forsøkene er definert hos senderenheten. I vårt oppsett ble dette satt til maksimalt fem forsøk med 4ms mellomrom. Siden basestasjonen skulle kunne kommunisere med opptil fem fjernkontroller ble det åpnet for denne funksjonen i pipe 0-4.

Systemet bruker et identifikasjonsnummer på opptil 5byte til å identifisere senderenhetene, hvor de første fire i dette systemet var identiske og lik 0xABCDEF42 mens den siste byten hadde en verdi i området 00-04 for å indikere fjernkontrollnummer 1-5. I basestasjonens L01-brikke fikk hver av disse adressene en egen pipe som mottok eventuelle datapakker fra den aktuelle avsenderen.

Sendestyrken ble satt til 0dBm (1mW) og dataoverføringshastigheten til 1Mbps. Siden det kun var snakk om retning og statusbeskjeder fra fjernkontrollene ble de satt opp til å sende kun én byte som inneholdt den informasjonen. Tilsvarende ble hver pipe satt opp til å indikere at de var fulle når de har mottatt denne ene byten.

Tabell 19: Variable brukt i SPIInit

Navn	Type	Innhold
temp	byte	Mottatte data fra nRF24L01

7.3.8 SPI

void SPI(BYTE info, BYTE* svar)
Se vedlegg O (SPInRF24.c).

Tabell 20: Variable brukt i SPI

Navn	Type	Innhold
info	byte	Data som skal ut på SDO
svar	byte	Data som er mottatt på SDI

Denne funksjonen sender data ut på bussen ved å skrive til SSPBUF. Data overføres i begge retninger samtidig slik at for hver bit som klokkes ut på SDO leses det inn et bit på SDI. Når Buffer Full bitet i SSPSTAT-registeret er satt er dette en indikasjon på at mottatte data er klart til å leses fra SSPBUF.

7.3.9 Les_nRF

void Les_nRF(BYTE* retn, BYTE* num)
Se vedlegg O (SPInRF24.c).

Tabell 21: Variable brukt i Les_nRF

Navn	Type	Innhold	Verdi
Retn	byte	Data fra fjernkontrollen	0 – 5
Num	byte	Fjernkontrolnr. (pipe-nummer)	1 – 5
Status	byte	Dummyvariabel for å ta imot data	

Denne funksjonen kalles når nRF-brikken indikerer at den har mottatt en ny pakke ved å legge IRQ-linja (koblet til RA0 på mikrokontrolleren) lav. Første trinn er å be om å få lese nyttelasten. Samtidig som denne kommandoen klokkes ut mottas statusregisteret i variabelen num. Deretter sendes en ren dummy-kommando ut for å klokke inn dataene fra mottaksbufferet i nRF-brikken.

Statusregisteret som ble lest inn i variabelen num indikerte hvilken pipe som hadde mottatt data i bitposisjon 1-3. Ved å maskere ut de andre bitene, bitshifte variabelen ett hakk til høyre og legge på én får man ut fjernkontrollnummeret i verdiområdet 1-5.

For å være sikker på at mottaksbufferet ikke skal kunne fylles opp tømmes det straks kommandoen er lest. Deretter fjernes interruptflagget for mottatte data (RX_DR).

Som en ekstra indikasjon skifter den ene lysdioden på basestasjonkortet mellom av og på for hver mottatte pakke.

7.3.10 EESkrivByte

BOOL EESkrivByte(unsigned int adresse, unsigned char skriv)
Se vedlegg M (Datalagring.c).

Denne funksjonen skriver én byte data til en spesifisert adresse i mikrokontrollerens innebygde EEPROM. Koden er basert på standardkoden som står beskrevet i databladet til mikrokontrolleren (PIC18F4620). Siden vår mikrokontroller har 1024 byte lagerplass er det nødvendig med 9 bit for å angi adressen. Den mest signifikante biten legges i EEADRH mens de åtte øvrige bitene legges i EEADR.

Tabell 22: Variable brukt i EESkrivByte

Navn	Type	Innhold
skriv	byte	Data som skal lagres
adresse	int	Adressen i EEPROM

7.3.11 EELesByte

BYTE EELesByte(unsigned int adresse)
Se vedlegg M (Datalagring.c).

Denne funksjonen leser og returnerer én byte fra en adresse i mikrokontrollerens EEPROM. Funksjonen er basert på standardkoden som står beskrevet i databladet til mikrokontrolleren (PIC18F4620).

7.3.12 LagreOppsett

BYTE LagreOppsett(BYTE nr, BYTE komm[], BYTE tekst[])
Se vedlegg M (Datalagring.c) og eventuelt vedlegg K (utdrag fra main.c).

Denne funksjonen tar for seg lagring av en fjernkontrollkommando i et eksisterende eller nytt oppsett. Funksjonen blir kjørt etter at dataene fra nettsiden har blitt prosessert og godkjent (alle data innenfor godkjente intervaller).

Tabell 23: Variable brukt i LagreOppsett

Navn	Type	Innhold	Verdi
i	byte	Tellervariabel til løkker	
ant	byte	Antall oppsett i EEPROM	0 – 10
nr	byte	Oppsettnummer fra HTTPExecCmd	0 – 10
nummer	int	Oppsettnummer for å angi posisjon	
Posisjon	int	Lagringsadresse i EEPROM	0 – 990
Komm	array	Tre byte prosessert kommando	
Tekst	array	Navn på oppsettet	

Den medsendte variabelen nr angir om oppsettet finnes fra før. Hvis den har en verdi mellom 1 og 10 betyr det at det er et eksisterende oppsett som skal oppdateres. Hvis den inneholder 0 betyr det at det er et nytt oppsett som skal lagres. Lagringen utføres kun hvis det er færre enn 10 oppsett lagret i EEPROM.

Hvert oppsett opptar 91byte plass fordelt på navnet (16byte) og de fem kommandoene (3byte) til hver av de fem fjernkontrollene.

Tabell 24: Lagringsformat for fjernkontrollkommandoer

Byte 0	Byte 1	Byte 2
NNNNNNNN	VAAAIIII	IIUUUUUU

De første åtte bitene angir nivånummeret kommandoen gjelder for. Verdi 1-200. Bitposisjon V angir hvilken av de to VikinX-routerene som skal motta kommandoen.

AAA angir hvor mange signaler ut over ett som skal overføres. Verdi 0-7. Inngangsnummerets fire øverste bit lagres i midterste byte og de to siste i siste byte. Verdi 0-63. Utgangsnummeret ligger i de seks nederste bitene i siste byte. Verdi 0-63.

Hvis hele lagringsprosessen forløper uten problemer returneres tallet 1 for å indikere suksess, ellers returneres tallverdien 0.

7.3.13 LagreIP

void LagreIP(BYTE nr, BYTE ip[], BYTE nvn[])
Se vedlegg M (Datalagring.c).

Denne funksjonen tar i mot et nummer som angir hvilken VikinX-router som dataene gjelder for, ip-adressen som skal lagres og eventuelt et nytt navn.

Funksjonen lagrer først IP-adressen. Når disse fire bytene er lagret sjekkes det om det har blitt sendt med et navn. Hvis dette er tilfelle skrives dette inn på riktig posisjon. Navnene lagres med startposisjon 0 eller 16, mens de tilhørende ip-adressenes lagerplass starter på henholdsvis posisjon 64 eller 68.

Tabell 25: Variable brukt i LagreIP

Navn	Type	Innhold	Verdi
nr	byte	Angir VikinX-router 1 eller 2	
i	byte	Telleravariabel til løkkene	
Posisjon	int	Skriveposisjon i EEPROM	0 – 71
ip	array	IP-adresse, 4byte	0 - 255
nvn	array	Navnet som skal lagres	

7.3.14 LastOppsett

void LastOppsett(BYTE nr, BYTE* kommliste)
Se vedlegg M (Datalagring.c).

Denne funksjonen får angitt nummeret til oppsettet den skal laste og en peker til hovedprogrammets kommandoliste som skal motta de leste kommandoene fra det valgte oppsettet.

Lesinga begynner på første byte etter oppsettets navn og slutter etter at alle 75 byte er lest.

Tabell 26: Variable brukt i LastOppsett

Navn	Type	Innhold	Verdi
nr	byte	Nummeret til oppsettet som ønskes lest	1 – 10
i	byte	Tellervariabel til løkka	0 – 75
kommliste	peker	Peker til arrayen som skal motta kommandoene	
Posisjon	int	Leseposisjon i EEPROM	97 – 990

7.3.15 SlettOppsett

void SlettOppsett(BYTE nr)
Se vedlegg M (Datalagring.c).

Denne funksjonen tar i mot nummeret til et oppsett som ønskes slettet. Hvis oppsettet er det siste i minnet blir det overskrevet med verdien 0xFF som indikerer at minneadressen er ledig. Hvis det ikke er det siste oppsettet som skal slettes blir først dataene for det siste oppsettet i minnet skrevet over oppsettet som ønskes slettet, før det siste oppsettet blir slettet.

Tabell 27: Variable brukt i SlettOppsett

Navn	Type	Innhold	Verdi
nr	byte	Nummeret til oppsettet som ønskes slettet	1 – 10
ant	byte	Antall oppsett i EEPROM	0 – 10
i	byte	Tellervariabel til løkker	
Posisjon	int	Leseposisjon i EEPROM	

7.3.16 FinnOppsett

BYTE FinnOppsett(char* navn)
Se vedlegg M (Datalagring.c).

Denne funksjonen tar i mot en peker til en tekststreng og leter blant de lagrede oppsettene for å se om det er mulig å finne et oppsett med samme navn. Hvis den finner det returneres oppsettets nummer. Hvis ikke returneres null.

Tabell 28: Variable brukt i FinnOppsett

Navn	Type	Innhold	Verdi
i, j	byte	Tellervariable til løkkene	
ant	byte	Antall oppsett i EEPROM	0 – 10
TmpTxt	array	Mellomlager for navn lest fra EEPROM	
Posisjon	int	Leseposisjon i EEPROM	

7.4 Beskrivelse av nettsiden

7.4.1 Generell introduksjon

Se vedlegg P for et bilde av nettsiden.
Nettsiden som vises når man kobler seg til basestasjonens ip-adresse med en nettleser har fire primære bruksområder:

1. Lagring/endring av kommandooppsettene.
2. Visning av lagrede kommandoer.
3. Lagring/endring av aktuelle VikinX-enheter
4. Slette lagrede oppsett.

Nettsiden er delt i to hovedkolonner hvor den venstre inneholder alt av input-felter og den høyre inneholder en oppdatert liste over lagrede oppsett og VikinX-routere.

Funksjonaliteten til nettsiden ble realisert ved bruk av javascript og cgi-filer med innhold som oppdateres av basestasjonen hver gang de blir lastet. Denne måten å lage nettsider på kalles for AJAX, som står for asynkront Javascript og XML. Se index.html i vedlegg A. Basisfunksjonene er beholdt fra demonstrasjonsnettsiden som fulgte med Microchip HTTP Stack 3.75 og har blitt supplert med følgende fire funksjoner:

Tabell 29: Oversikt over nettsidens nye funksjoner

Funksjonsnavn	Hensikt
LagreKommando	Samler inntastede data og ber basestasjonen om å lagre oppsettet.
VisKommandoer	Viser lagrede kommandoer for en av fjernkontrollene i et oppsett.
LagreVXIP	Lagrer/endrer ip-adresse og navn til de aktuelle VikinX-routerene.
SlettOppsett	Ber om at et angitt oppsett slettes.

Hver gang det lagres, endres eller slettes et oppsett oppdateres listen over lagrede oppsett etter en forsinkelse på et halvt sekund. Denne forsinkelsen ble lagt inn for å sørge for at basestasjonen hadde rukket å lagre alle endringer før de skulle leses fra EEPROM. Tilsvarende oppdateres listen over tilgjengelige VikinX-routere hver gang en endring av ip-adresse eller navn på disse foretas.

Koden som er tilgjengelig åpner for at nettsiden kunne blitt brukt til å vise samme type statusbeskjeder som displayet på basestasjonkortet. Vi har valgt å ikke gjøre dette i nåværende utgave av nettsiden, men hvis det viser seg at dette kan være nyttig er det ikke vanskelig å legge til den nødvendige funksjonaliteten senere.

7.4.2 Nettsidens bestanddeler

Tabell 30: Nettsidens filer

Statiske filer	
index.html	HTML-koden for nettsiden (ikke W3C-godkjent i nåværende form)
vxrlogo.gif	Logoen på toppen av nettsiden

Dynamiske filer	
ipVXnavn.cgi	Valgliste med navn på VikinX-routere. Brukes i ip-endringsfeltet
OSnavn.cgi	Liste med navnene på de lagrede oppsettene i basestasjonen.
Visdata.cgi	Liste med lagrede kommandoer for en fjernkontroll i et angitt oppsett
VXIP.cgi	Liste med de lagrede navnene og ip-adressene til VikinX-routerene.
VXnavn.cgi	Valgliste med navn på VikinX-routere. Brukes ved lagring av oppsett.

8.0 Resultat

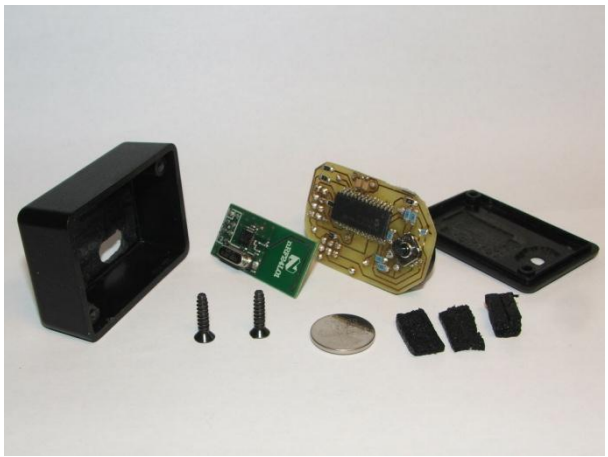
8.1 Faktiske resultater

Resultatet av denne oppgaven ble en fungerende enhet som fungerer tilfredsstillende i henhold til kravspesifikasjonen med unntak av batteristatusmåling. Vi har kommet frem til løsning på dette, men på grunn av tidsnød har det ikke latt seg gjennomføre. Vi er spesielt fornøyd med at fjernkontrollen ble såpass kompakt og ergonomisk. Når det er sagt er det selvfølgelig rom for å gjøre den enda mer kompakt.

Rekkevidden ser ut til å være ganske nøyaktig 10 meter med fri sikt. Dette tror vi vil kunne bedres hvis vi får på en skikkelig antenne på basestasjonen. Responstiden til systemet er upåklagelig så lenge fjernkontrollen er innenfor rekkevidde.

Nettsiden fungerer etter hensikten og brukergrensesnittet er forståelig for de fleste etter litt tilvenning.

Mulige forbedringspunkter:
- 20mm batteri i stedet for 16mm for å bedre batterikapasiteten.
- Lavere klokkefrekvens på mikrokontroller i fjernkontrollen for å spare strøm.
- Enda mer kompakt utforming av fjernkontrollen kan muligens oppnås ved å bruke nRF24E1 da denne har en egen innebygget mikrokontroller.
- Realisere spenningsmåling i fjernkontrollen.
- Lage en egen resetknapp så man slipper å åpne fjernkontrollen hver gang man skal skifte ID.
- Kommunikasjon med andre basestasjoner.
-



Figur 28: Fjernkontrollens bestanddeler

Figur 27: Fjernkontrollen i profil sammen med en vanlig lighter

9.0 Konklusjon

Prosjektet har vært en meget lærerik prosess selv om vi brukte lengre tid enn beregnet på å komme i gang med den reelle oppgaveløsningen. Dette føler vi var delvis forårsaket av vår manglende erfaring fra faget elektronikk-konstruksjon. Vi brukte mye tid på å sette oss inn i hvordan vi skulle utnytte og programmere mikrokontrollerne vi hadde valgt. Det ble ikke lettere av at den medfølgende kildekoden til utviklingskortet vårt var mindre samarbeidsvillig enn forventet. Heldigvis snudde dette seg rundt påsketider.

Å utvikle en fungerende løsning fra bunn av er en krevende og omfattende jobb, men også en morsom prosess når ting begynner å fungere. I vårt tilfelle slapp vi heldigvis å begynne på helt bar bakke med vårt valg av utviklingskort.

I løpet av prosjektet har vi blitt flinkere til å realisere løsninger samtidig som vi tar hensyn til flere konkurrerende faktorer. Rent praktisk har vi blitt vesentlig flinkere til å lage de nødvendige kretskortene og å håndtere loddebolten. Vår økte kunnskap om feilsøking ved hjelp av oscilloskop, debuggingsverktøy og det gode gamle multimeteret er ferdigheter vi regner med å ha god nytte nytte av senere.

I løpet av prosjekttiden har vi søkt assistanse hos flere av lærerne ved elektrosekjsonen som har vært behjelpelige med både utstyr, tips og triks.

Øivind Stuan

Anders Johansen

10.0 Litteratur- og kildeliste

- TCP/IP Lean: Web servers for embedded systems 2nd edition av Jeremy Bentham ISBN 1-57820-108-X

- Embedded Design with the PIC18F452 Microcontroller av John B. Peatman ISBN 0-13-046213-6

- ELFA for bestilling av komponenter www.elfa.se

- Farnell for bestilling av komponenter www.farnell.no

- Datablad for PIC16F873

- Datablad for PIC18F4620

- Datablad PIC18F452

- Datablad joystick

- Datablad nRF24L01

- Datablad nRF24L01 Evaluation kit

- Datablad LM385 spenningsreferanse

- Datablad 74LS148 Enkoder

- Datablad LM317 spenningsregulator

- PICDEM.net users guide

- B Knudsen CC5X kompilator http://www.bknd.com/cc5x/index.shtml

- Microchip C18 kompilator http://www.microchip.com/stellent/idcplg?IdcService=SS_GET_PAGE&nodeId=1406&dDocName=en010014&part=SW006011

Alle databladene og programmene vi brukte, med unntak av ISIS og ARES, ligger på CD'en bakerst i heftet.

11.0 Vedlegg