NTNU
Det skapende universitet

Energieffektiv dataoverføring

Rune Bjønnum

Master i kommunikasjonsteknologi
Innlevert: juli 2014
Hovedveileder: Lars Magne Lundheim, IET
Medveileder: Ivar Holand, ATMEL Norway

Norges teknisk-naturvitenskapelige universitet
Institutt for elektronikk og telekommunikasjon

Tittel:	Energieffektiv dataoverføring
Student:	Rune Bjønnum

Oppgavetekst:

Det er et økende marked for trådløs dataoverføring fra eller til utstyr med begrenset eneregitilgang. I slike applikasjoner er det viktig at utstyret bruker så lite energi som mulig. For å optimalisere energibruken må flere systemaspekter ses i sammenheng. Eksempel på dette er bruk av lokal prosessering i sensornoden for å oppnå redusert datarate i radiodelen. Andre momenter er valg av modulasjonsmetode, koding og høyere lags protokoller.

I et tidligere prosjekt ble disse forholdene undersøkt for en applikasjon med i utgangspunktet lav datarate. I det foreliggende prosjektet skal problemstillingen videreføres til applikasjoner med høyere datarate, så som audio, bilder og video.

I samråd med oppdragsgiver og veileder skal noen aktuelle applikasjoner velges. Potensielle systemløsninger skal sammenlignes med hensyn til energieffektivitet.

Arbeidet kan utføres ved analytiske beregninger, simulering og måling på tilgjengelig utstyr. Prosjektet utføres i samarbeid med ATMEL Norway.

Hovedveileder:	Lars Lundheim, IET
Ekstern veileder:	Ivar Holand, ATMEL Norway

Sammendrag

Små, batteridrevne enheter som kobles til et datanettverk har medført et økende behov for energieffektiv trådløs datakommunikasjon. Denne oppgaven undersøker hvordan datakompresjon kan benyttes for å redusere behovet for kommunikasjon, og hvordan dette påvirker energiforbruket. Resultatene som presenteres tyder på at datakomprimering bør utnyttes, dersom det tillates av applikasjonen.

Abstract

Small, battery-powered devices connected to a data exchange network have led to an increasing need for energy efficient wireless communications. This document investigates how data compression can be exploited to reduce the need for communication, and how this affects energy consumption. The results presentet indicate that data compression should be exploited, if permitted by the application.

Innhold

1 Introduksjon

1.1 Motivasjon og mål

Teknologipressen anno 2014 nærmest oversvømmes av nyheter som omhandler "the Internet of Things" (IoT) [1, 2, 3] og "Wearable Technology" (wearables) [4, 5]. Google annonserte nylig at deres operativsystem for mobiltelefoner, Android, gjør sitt inntog på markedet for wearables, i første omgang gjennom smarte klokker [6]. Produsenter som Samsung [7], LG [8] og Motorola [9] har sine respektive modeller, basert på Android Wear-plattformen, like rundt hjørnet. Det er også forventet at Apple skal lansere sitt bidrag til smartklokkemarkedet innen oktober 2014 [10, 11, 12]. Mer tradisjonelle hjemmeprodukter, som kjøleskap [13], røykvarslere [14] og blomsterpinner (!) [15] kan nå få utvidet funksjonalitet ved å kobles til et (trådløst) datanettverk.

Fremveksten av produkter som nevnt over, har medført et økende behov for energieffektiv trådløs kommunikasjon. I trådløse sensornoder[1] og andre batteridrevne enheter som tilhører et trådløst nettverk, er det ønskelig å maksimere batterilevetiden, for å redusere hyppigheten av batteribytte eller lading. Litteraturen [16, 17, 18] beskriver generelle teknikker for optimalisering av energiforbruk i slike enheter. En av teknikkene som nevnes, går ut på å skru av systemkomponenter, som for eksempel radioen, når de ikke er i bruk. [16] påstår at kommunikasjonsdelen av en sensornode har et høyere energiforbruk enn prosesseringsdelen, og at sensornoden derfor bør prosessere sensordata dersom det reduserer behovet for kommunikasjon. Påstanden underbygges med tall av [17], der det tydelig framgår at radioen står for en signifikant del av systemets energiforbruk.

Et tidligere arbeid [19] viser at påstanden stemmer for en gitt applikasjon med lav datarate. Denne oppgaven tar for seg energiforbruk i en sensornode som opererer med høyere datarate. Oppgaven ønsker å besvare følgende spørsmål: hvordan påvirkes energiforbruket til en sensornode, dersom den benytter en datakompresjonsalgoritme for å redusere behovet for kommunikasjon? Oppgaven har en praktisk innfallsvinkel, og overføring av lyd benyttes som eksempelapplikasjon.

[1] Sensornode: enhet som kan samle inn og prosessere sensordata, samt kommunisere med andre noder i et nettverk.

1.2 Struktur

Etter denne introduksjonen, presenteres nødvendig bakgrunnsmateriale for oppgaven. Deretter følger en beskrivelse av tester og målinger som er gjort. Resultater presenteres og diskuteres, før oppgavens konklusjon og forslag til videre arbeid.

1.3 Metode

Oppgaven har en praktisk innfallsvinkel. Et forenklet sensornettverk implementeres, og energiforbruk i en sensornode ved ulike situasjoner analyseres, basert på fysiske målinger.

1.4 Verktøy

En rekke verktøy vil bli benyttet på veien mot målet. Mikrokontrollere, radiomoduler og måleutstyr som oscilloskop og multimeter er håndfaste eksempler. På programvaresiden benyttes Microsoft Windows 8.1 operativsystem, MathWorks MATLAB R2013a og diverse andre applikasjoner.

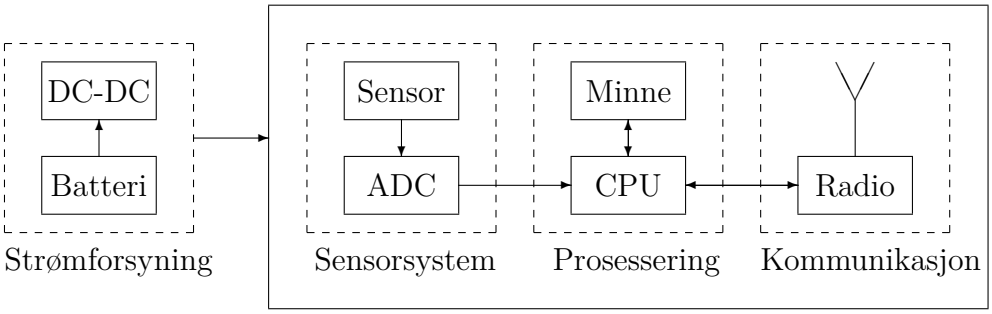
1.5 Måleenheter

Det er mye forvirring rundt bruken av enheter for data, som kB og MB. I denne oppgaven benyttes SI-prefiksenes opprinnelige betydning. Dette innebærer at 1kB = 1000B. Binære prefikser, som Ki og Mi, benyttes for å beskrive multiplum av 1024. 1KiB = 1024B. 1MiB = 1024KiB = 1024^2B.

2 Bakgrunn

2.1 Sensornode

En sensornode er en enhet som kan samle inn og prosessere sensordata, samt kommunisere med andre enheter i et nettverk. En typisk sensornode, illustrert i figur 2.1, kan deles opp i fire delsystemer: strømforsyning, sensorsystem, prosessering og kommunikasjon [17].

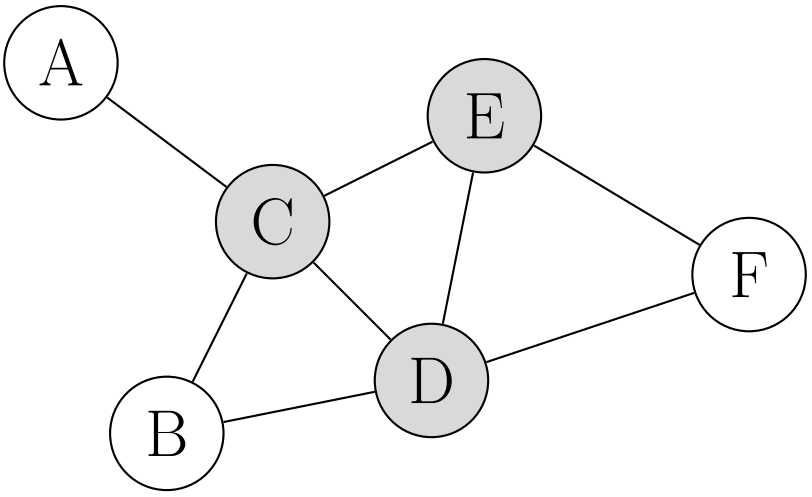


Figur 2.1: Arkitektur for trådløs sensornode.

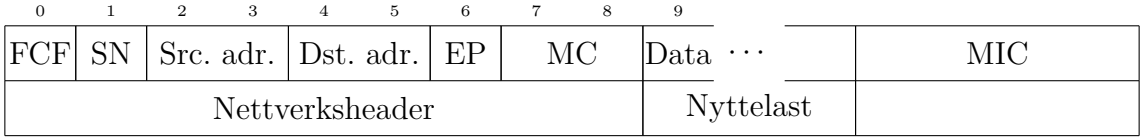
Strømforsyningen består av en energikilde, for eksempel et batteri, og nødvendige spenningsomformere (DC-DC). Sensorsystemet håndterer innsamling av sensordata, ved hjelp av sensorer og tilhørende analog- til digital-omformere (ADC). Prosesseringssystemet består av en mikroprosessor eller mikrokontroller med tilhørende minne, og håndterer innkommende data fra sensorsystemet. Kommunikasjonssystemet i en trådløs sensornode består typisk av en radio, beregnet på kortdistanse trådløs kommunikasjon.

2.2 Atmel Lightweight Mesh

Sensornoden i denne oppgaven er basert på Atmel Lightweight Mesh [20], som i følge [21] er Atmels enkleste kommunikasjonsrammeverk beregnet på IEEE 802.15.4-baserte radioer. Lightweight Mesh (LWM) er laget med tanke på brukervennlighet,



Figur 2.2: Ruternoder (grå) og kantnoder (hvite).



Figur 2.3: Rammeformat Lightweight Mesh.

med beskjedne maskinvarekrav. Dokumentasjonen [20, side 5] oppgir 8kB flashminne og 4kB RAM[1] for en typisk applikasjon.

Lightweight Mesh støtter to ulike typer noder, illustrert i figur 2.2: ruternoder (grå) og kantnoder (hvite). En linje mellom to noder angir at disse er innen rekkevidde til å kunne kommunisere direkte.

Ruternoder bygger opp kjernen av et LWM-nettverk, og kan videresende meldinger for å muliggjøre kommunikasjon mellom to noder som er utenfor hverandres rekkevidde, for eksempel node A og node F i figur 2.2. Det forventes at enhver ruternode alltid er tilgjengelig.

En typisk kantnode er gjerne batteridrevet, og ønsker derfor å gå i strømsparemodus når den ikke sender eller mottar data. LWM tar ikke ansvar for levering av data til en sovende node. Dersom nødvendig, må slik funksjonalitet implementeres av applikasjonen som benytter LWM.

Lightweight Mesh implementerer tredje lag i OSI-modellen [22]; nettverkslaget. LWM tar seg av konfigurasjon av det underliggende IEEE 802.15.4 MAC-laget, beskrevet i 2.3.2. Eksakte innstillinger er gjengitt i LWM-dokumentasjonen [20, kapittel 4].

En datapakke[2] omtales i Lightweight Mesh som en ramme. En LWM-ramme, illustrert i figur 2.3, utgjør MAC-lagets nyttelast (MSDU). LWM benytter en egen Frame Control Field (FCF) på 1B, som beskriver hvordan rammen skal håndteres, etterfulgt av et 1B sekvensnummer (SN) for rammeidentifikasjon. Nettverket benytter 2B adresser for rammens kilde (Src. adr.) og destinasjon (Dst. adr.).

Etter adressene kommer 1B (EP), som er en sammenslåing av to 4bit identifikato-

[1] RAM: Random Access Memory.

[2] Datapakke: en datablokk av gitt lengde, som overføres samlet.

Figur 2.4: Rammeformat IEEE 802.15.4 PHY.

rer: Source Endpoint og Destination Endpoint. Lightweight Mesh benytter såkalte endpoints til å avgjøre hvordan mottatte data skal behandles. En applikasjon kan ha inntil 15 åpne endpoints samtidig.

Dersom LWM-rammens FCF indikerer at rammen skal mottas av en gruppe noder (multicast), benyttes en 2B Multicast Header (MC). Dersom rammen er adressert til en enkelt node, droppes MC fra rammen. Deretter følger LWM-rammens nyttelast (Data).

Nettverket tilbyr støtte for kryptering av nyttelasten. To krypteringsalgoritmer støttes: maskinvareakselerert AES-128[3] (dersom støttet av aktuell maskinvare) og XTEA[4]. Dersom nyttelasten er kryptert, avsluttes en LWM-ramme med en 4B Message Integrity Code (MIC).

2.3 IEEE 802.15.4

IEEE 802.15.4 [23, 24] beskriver nødvendige protokoller for de to nederste lagene i OSI-modellen [22]: fysisk lag, kalt PHY, og datalink-lag, kalt MAC.

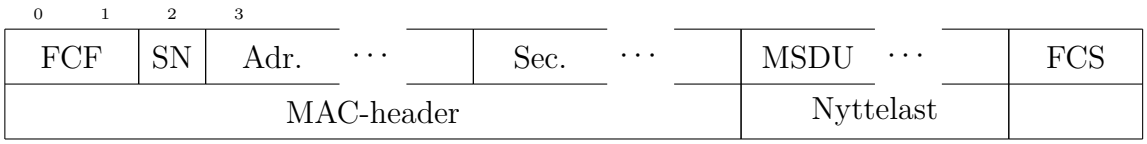
2.3.1 PHY

Oppbyggingen til en PHY-ramme, kalt PPDU[5] er illustrert i figur 2.4. En gyldig PPDU starter med en 5B synkroniseringsheader (SHR). Først 4B preamble, der alle byte er lik 0, etterfulgt av en Start of Frame Delimiter (SFD) med verdi 0xA7. SFD indikerer for mottakeren at neste innkommende byte er en PHY header (PHR). Mest signifikante bit i PHR er reservert; de resterende 7bit angir lengden på rammens nyttelast (PSDU[6]). Øvre grense for nyttelastens lengde er derfor gitt ved $2^7 - 1 =$ 127B.

2.3.2 MAC

PSDU inneholder en MAC-ramme, kalt MPDU[7], og illustrert i figur 2.5. MPDU starter med en 2B Frame Control Field (FCF), som beskriver innholdet i resten av rammen, og hvordan dette skal tolkes. Deretter følger et 1B sekvensnummer (SN) for å muliggjøre identifikasjon av rammen, samt et adressefelt (Adr.). IEEE 802.15.4

[3] AES-128: Advanced Encryption Standard med 128bit krypteringsnøkkel.
[4] XTEA: eXtended Tiny Encryption Algorithm.
[5] PPDU: PHY Protocol layer Data Unit.
[6] PSDU: PHY Service Data Unit.
[7] MPDU: MAC Protocol layer Data Unit.



Figur 2.5: Rammeformat IEEE 802.15.4 MAC.

byr på noen valgmuligheter for adressering av noder. Adressefeltet kan være mellom 0B og 20B langt, og lengden bestemmes av rammens FCF.

IEEE 802.15.4-2006 [23] støtter ulike sikkerhetsmekanismer, for å beskytte rammens nyttelast. Dersom rammen skal sikres, benyttes en sikkerhetsheader (Sec.) på inntil 14B til å beskrive hvordan rammen er sikret. Etter sikkerhetsheaderen kommer rammens nyttelast (MSDU[8]), og til slutt en 2B sjekksum, Frame Check Sequence (FCS), for å detektere korrupte rammer.

2.4 Nyttelast

Innholdet i en ramme fra et gitt lag i OSI-modellen kan generelt deles i to: nyttelast og metadata. I Lightweight Mesh utgjør for eksempel nettverksheaderen sammen med eventuell MIC nettverkslagets metadata. Maksimal størrelse på nyttelasten, D_{Nmax}, er derved gitt ved ligning 2.1, der D_{Rmax} er maksimal rammestørrelse, og D_{M} er størrelsen på rammens metadata.

$$D_{\mathrm{Nmax}} = D_{\mathrm{Rmax}} - D_{\mathrm{M}} \tag{2.1}$$

Ligning 2.1 kan benyttes for å finne ut hvor mye data en applikasjon kan sende i hver LWM-ramme. Som vist i 2.3.1, er maksimal nyttelast på PHY-laget 127B. Nyttelasten til PHY-laget er en MAC-ramme. For MAC-laget er D_{Rmax} derfor lik 127B. LWM konfigurerer MAC-laget slik at MAC-headerens lengde blir 9B: 2B FCF, 1B SN og 6B til adressering. MAC-lagets metadata utgjør derved 9B MAC-header og 2B FCS; totalt $D_{\mathrm{M}} = 11\mathrm{B}$. For MAC-laget er derfor $D_{\mathrm{Nmax}} = 116\mathrm{B}$.

MAC-lagets nyttelast er en LWM-ramme. For nettverkslaget er D_{Rmax} derfor lik 116B. I denne oppgaven benyttes verken multicast eller sikkerhet. Nettverksheaderens lengde blir da 7B. Siden MIC kan droppes, består nettverkslagets metadata kun av nettverksheaderen, og $D_{\mathrm{M}} = 7\mathrm{B}$. For nettverkslaget er derfor $D_{\mathrm{Nmax}} = 109\mathrm{B}$, og denne nyttelasten kan benyttes av overliggende applikasjon.

2.5 Datakompresjon

Formålet med datakompresjon er å representere en datablokk med lengde D_{raw} ved hjelp av en annen datablokk med lengde D_{comp}, der $D_{\mathrm{comp}} < D_{\mathrm{raw}}$. Det finnes mange ulike teknikker for å oppnå dette, både applikasjonsspesifikke og generelle. Alle datakompresjonsalgoritmer, uansett formål, tilhører en av to klasser: tapsfrie algoritmer (lossless), som muliggjør eksakt rekonstruksjon av kildematerialet, og algoritmer

[8]MSDU: MAC Service Data Unit

som innfører et selektivt tap av informasjon (lossy). Eksempler på tapsfrie kompresjonsalgoritmer er LZ77 [25] og bzip2 [26]. Eksempler på algoritmer som innfører tap er JPEG [27], som er beregnet på komprimering av digitale fotografier, og AAC [28], som er beregnet på lyd.

2.6 Digital representasjon av lyd

Linear Pulse Code Modulation (LPCM) er en grunnleggende metode for digital representasjon av et analogt signal, som for eksempel lyd. LPCM-representasjonen finnes ved å måle det analoge signalets amplitude ved regelmessige tidsintervaller, og kvantisere målingene ved hjelp av en lineær kvantiserer med 2^B trinn. B angir LPCM-signalets bitdybde, som beskriver hvor mange bit som benyttes for å representere hver måling (sample). Signalets punktprøvingsfrekvens, f_s, beskriver hvor mange målinger som gjøres per sekund.

2.7 ADPCM

I stedet for å representere hver sample i et lydsignal hver for seg, er det mulig å utnytte at nabosamples ofte er ganske like hverandre. Dette gjøres av blant annet ADPCM[9], der en prediktor benyttes til å estimere neste sample basert på et antall foregående samples. ADPCM regner ut differansen mellom hver lydsample og dens predikerte verdi. Differansen føres inn i en kvantiserer, som returnerer et antall kvantiseringstrinn. For å rekonstruere differansen, multipliseres antall kvantiseringstrinn med størrelsen på kvantiseringstrinnet. Både prediktoren og størrelsen på kvantiseringstrinnet kan være adaptiv.

En konkret variant av ADPCM er IMA ADPCM [29], utviklet av IMA[10] tidlig på 1990-tallet. Denne benytter seg av en svært enkel prediktor; predikert verdi for en sample finnes ved å dekode forrige sample. Prediktoren er ikke adaptiv. Kvantisereren gir ut en 4bit verdi for hver sample. Adapsjon av størrelsen på kvantiseringstrinnet gjøres ved tabelloppslag. Detaljene for hvordan dette fungerer, er utenfor denne oppgavens rammer, men den spesielt interesserte leser anbefales å sjekke ut [30] og [29]. For denne oppgaven er det nyttig å vite at IMA ADPCM ble utviklet for å gi (på den tiden) god lydkvalitet og god datakompresjon, og samtidig ha beskjedne krav til prosessorkraft. Disse faktorene, kombinert med tilgjengelighet av kildekode, har ført til at IMA ADPCM benyttes som prosesseringsalgoritme i denne oppgaven. IMA ADPCM er en lossy algoritme, der signaltapet skjer ved kvantisering av differansene.

2.8 Energiforbruk

Denne oppgaven ønsker å sammenligne energiforbruk ved prosessering og trådløs overføring av data. Energiforbruket til en krets, E, henger sammen med et effekt-

[9] ADPCM: Adaptive Differential Pulse Code Modulation.
[10] IMA: The Interactive Multimedia Association.

forbruk, P, over et tidsintervall, t, som beskrevet i ligning 2.2. For å bestemme energiforbruket til en applikasjon som kjører på en mikrokontroller, er t lik kjøretiden til applikasjonen, målt i sekunder.

$$E = Pt \tag{2.2}$$

Effektforbruket til en krets, P, beskrives generelt av ligning 2.3, der V er elektrisk spenning over kretsen, og I er elektrisk strøm gjennom kretsen.

$$P = VI \tag{2.3}$$

Innsetting av ligning 2.3 i ligning 2.2, resulterer i ligning 2.4. Denne gir et uttrykk for kretsens energiforbruk, der alle variabler kan måles.

$$E = VIt \tag{2.4}$$

3 Testoppsett

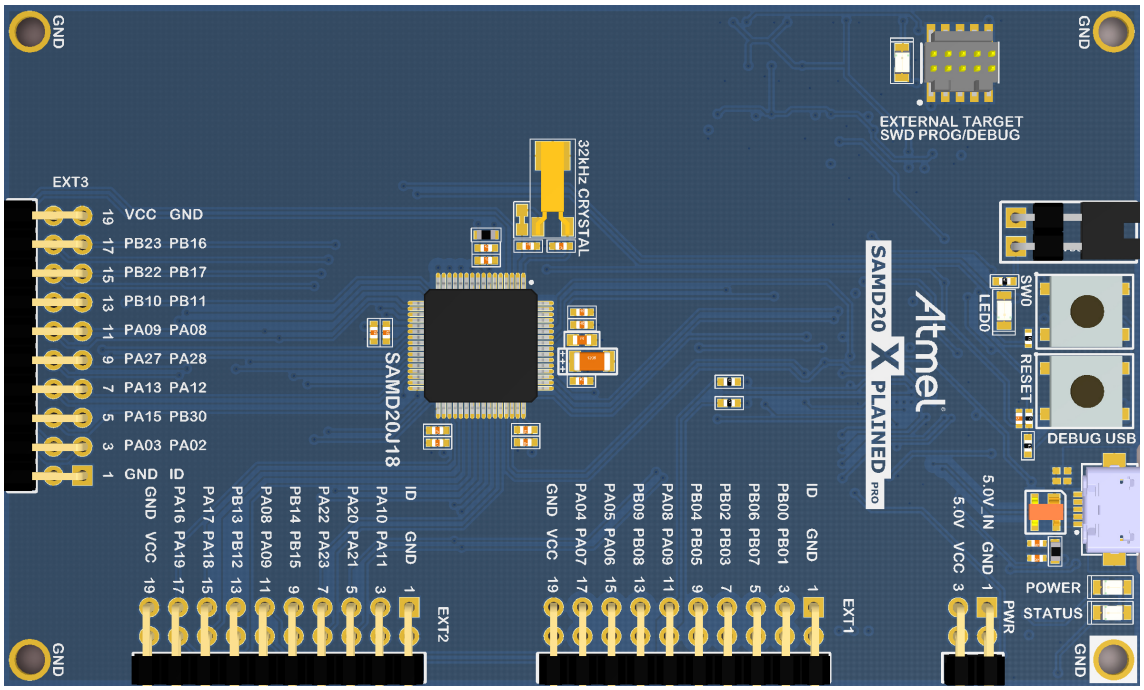
Denne oppgaven tar for seg et enkelt sensornettverk, bestående av to noder: en sensornode (data source) og en mottakernode (data sink), som illustrert i figur 3.1. Sensornoden ønsker å overføre et LPCM-lydsignal med punktprøvingsfrekvens f_s = 8kHz og bitdybde B_{raw} = 16bit/sample trådløst til mottakernoden.

Det antas at sensornoden er batteridrevet, og det er ønskelig at batteriet skal vare lengst mulig mellom hver opplading. Det antas også at mottakernoden er stasjonær, og at den derfor i praksis har ubegrenset energitilgang. I et slikt tilfelle kan det fokuseres på å redusere energiforbruket i sensornoden alene, uten å ta hensyn til energiforbruket til nettverket som helhet.

Denne oppgaven ønsker å undersøke hvordan sensornodens energiforbruk påvirkes, dersom lydsignalet komprimeres ved hjelp av IMA ADPCM før overføring. Det er også interessant å undersøke om sensornoden greier å gjennomføre datakompresjon i sanntid. For å finne ut dette, implementeres en prototype av begge nodene, og følgende tester gjennomføres på sensornoden:

1. Kompleksitetstest. Estimat av kompleksiteten til IMA ADPCM-algoritmen.
2. Prosesseringstest. Prosesseringstid for 8000 samples.
3. Referansetest. Energiforbruk ved overføring av 8000 ukomprimerte samples.
4. Systemtest. Energiforbruk ved prosessering og overføring av 8000 samples.
5. Overføringstest. Energiforbruk ved overføring av 8000 prosesserte samples.

Figur 3.1: Sensornettverk bestående av to noder.



Figur 3.2: SAM D20 Xplained Pro. (Illustrasjon: Atmel)

3.1 Maskinvare

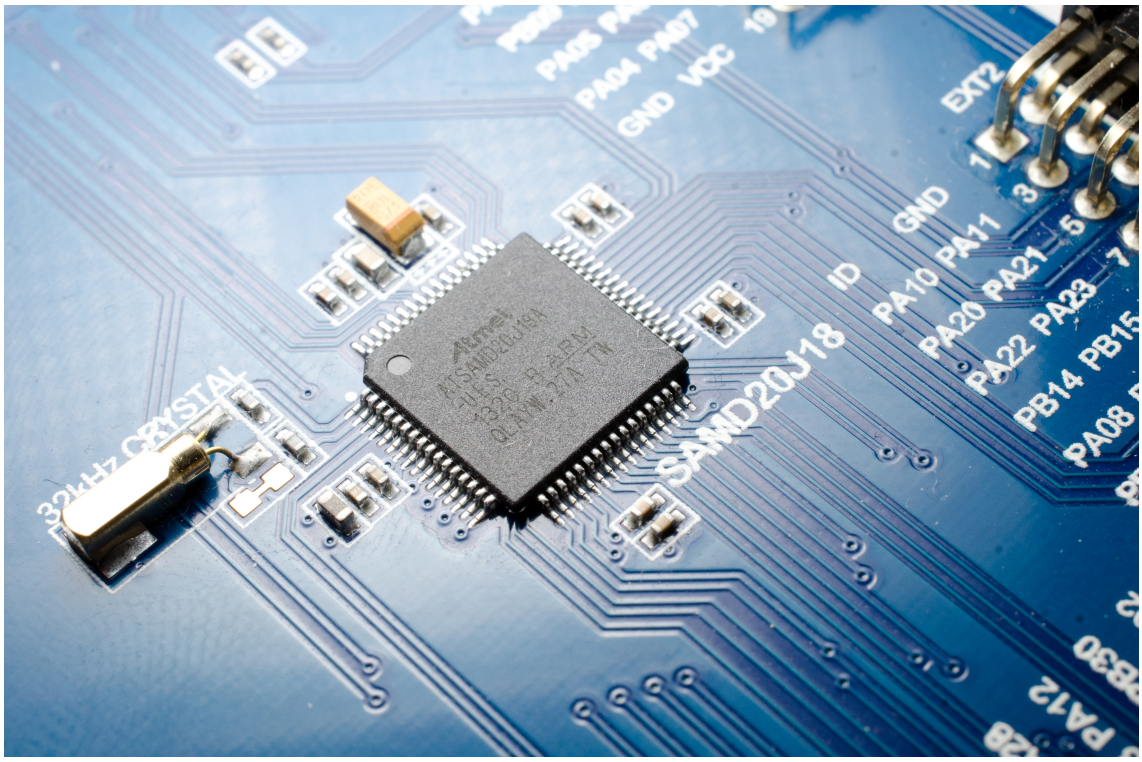
3.1.1 Mikrokontroller

Nodene implementeres på Atmel SAM D20 Xplained Pro [31]; et utviklingskort for Atmels SAM D20 mikrokontrollere, illustrert i figur 3.2 og 3.3. SAM D20 er en serie energieffektive 32-bits mikrokontrollere, basert på en ARM Cortex-M0+-kjerne. I følge ARM selv, er Cortex-M0+ den mest energieffektive ARM-prosessoren på markedet [32]. SAM D20 har ikke maskinvarestøtte for prosessering av flyttall, men den har en "single cycle multiplier", som gjør den i stand til å gjennomføre multiplikasjon av 32-bits heltall i løpet av bare én klokkesykel. Dette er meget nyttig innen digital signalbehandling.

Utviklingskortet inneholder en ATSAMD20J18A mikrokontroller [33]. Denne har 256kB flash-minne, 32kB SRAM, og støtter klokkefrekvenser på inntil 48MHz. Under alle forsøk har mikrokontrolleren benyttet en klokkefrekvens på $f_{\text{CPU}} = 8\text{MHz}$, generert av en intern RC-oscillator (kalt OSC8M i databladet [34]).

Programmering av mikrokontrolleren foregår ved hjelp av utviklingskortets feilsøkingsverktøy[1]. Kommunikasjon mellom datamaskin og feilsøkingsverktøy går via USB (Universal Serial Bus). Programmering og feilsøking på mikrokontrolleren benytter et Serial Wire Debug-grensesnitt (SWD) mellom feilsøkingsverktøy og mikrokontroller. Kommunikasjon mellom datamaskin og mikrokontroller er mulig ved hjelp av Atmels proprietære Data Gateway Interface (DGI), eller via UART (Universal Asynchronous Receiver/Transmitter). Sistnevnte er tilgjengelig fra tilkoblet datamaskin gjennom en virtuell serieport (COM-port).

[1] EDBG - Embedded Debugger.



Figur 3.3: ATSAMD20J18A mikrokontroller. (Foto: R. Bjønnum)

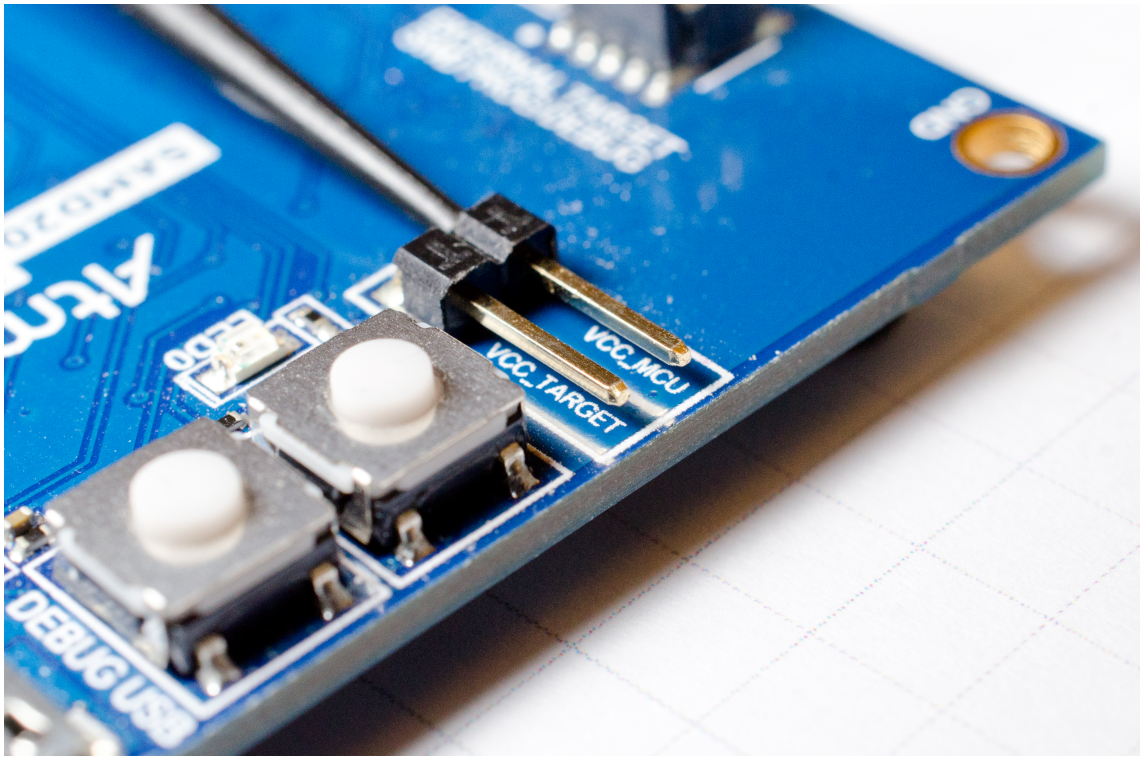
Kortet kan forsynes med strøm fra feilsøkingsverktøyets USB-kontakt, eller fra en ekstern strømforsyning. Valgt strømforsyning reguleres av en Exar SPX3819R2-L/TR [35] variabel spenningsregulator, som i følge [36] er konfigurert til å gi ut en spenning på 3.293V, med maksimalt avvik på 1%.

Kortet er utstyrt med en egen kontakt for å måle strømforbruket til mikrokontrolleren, vist i figur 3.4. Mikrokontrolleren forsynes med strøm fra VCC_MCU, mens VCC_TARGET kommer fra utgangen på spenningsregulatoren. Ved normal bruk kobles VCC_TARGET og VCC_MCU sammen ved hjelp av en jumper, som er fjernet i figur 3.4. Ved å fjerne jumperen, og koble inn et amperemeter mellom VCC_TARGET og VCC_MCU, kan man enkelt måle hvor mye strøm som går til selve mikrokontrolleren.

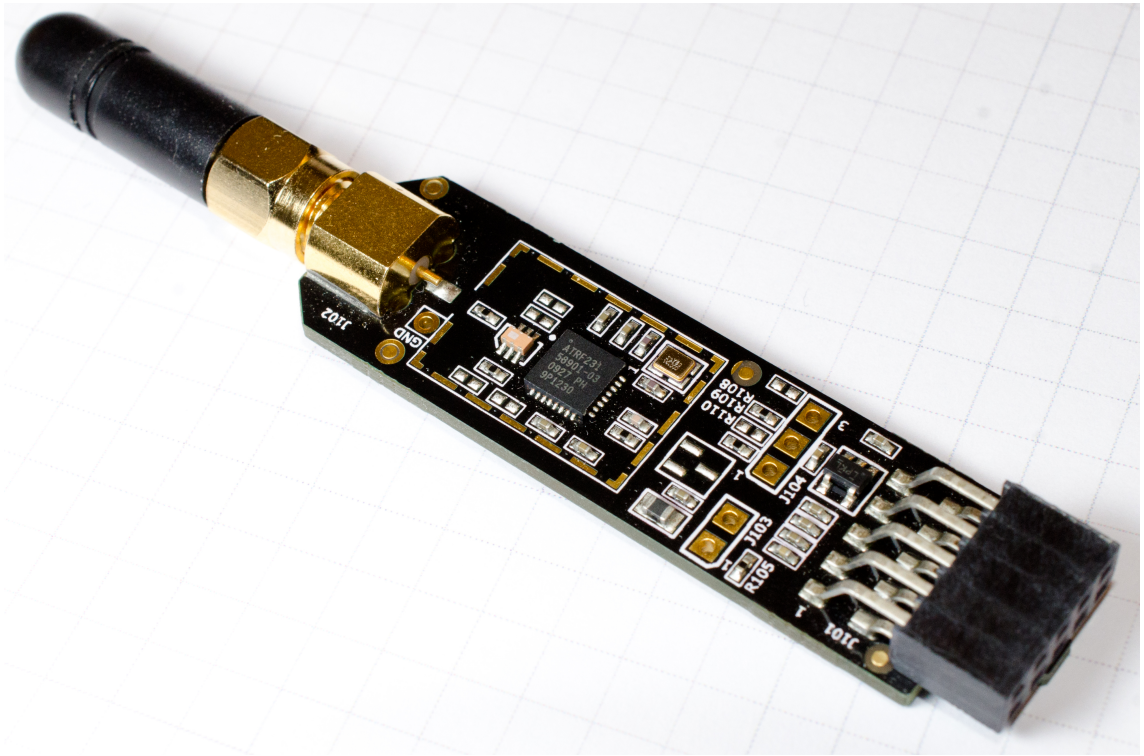
For tilkobling av utvidelseskort, som for eksempel I/O1 Xplained Pro [37] eller OLED1 Xplained Pro [38], er SAM D20 Xplained Pro utstyrt med tre 20-pins tilkoblingspunkter. Kortet benyttet i denne oppgaven har serienummer 0200008984.

3.1.2 Radio

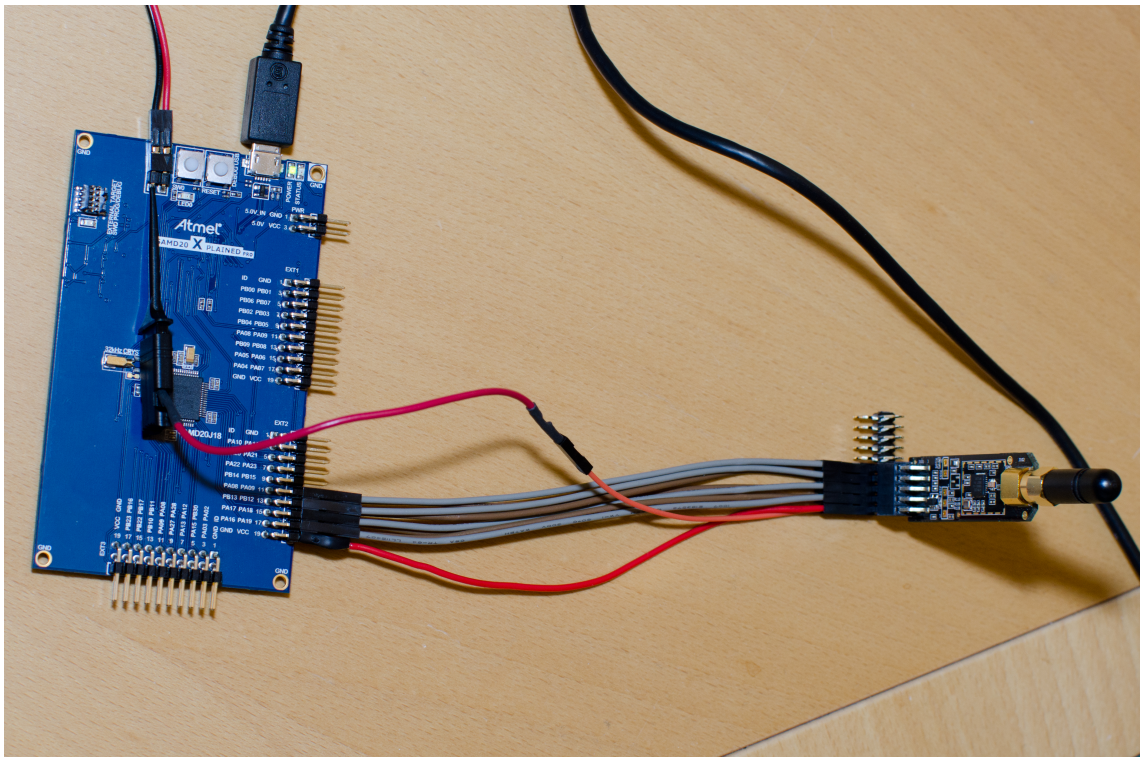
Trådløs kommunikasjon håndteres av Atmel AT86RF231; en 2.4GHz radiomodul beregnet på IEEE 802.15.4, ZigBee, 6LoWPAN, RF4CE, SP100, WirelessHART og ISM-applikasjoner [39]. Denne selges blant annet ferdig montert på et kretskort, sammen med alle nødvendige eksterne komponenter, som en del av utviklingssettet Atmel RZ600 [40]. AT86RF231-kortet fra RZ600 er avbildet i figur 3.5, og et slikt kort har blitt benyttet i denne oppgaven.



Figur 3.4: Kontakt for strømmåling. (Foto: R. Bjønnum)



Figur 3.5: AT86RF231 radiomodul. (Foto: R. Bjønnum)



Figur 3.6: Mikrokontroller og radio sammenkoblet. (Foto: R. Bjønnum)

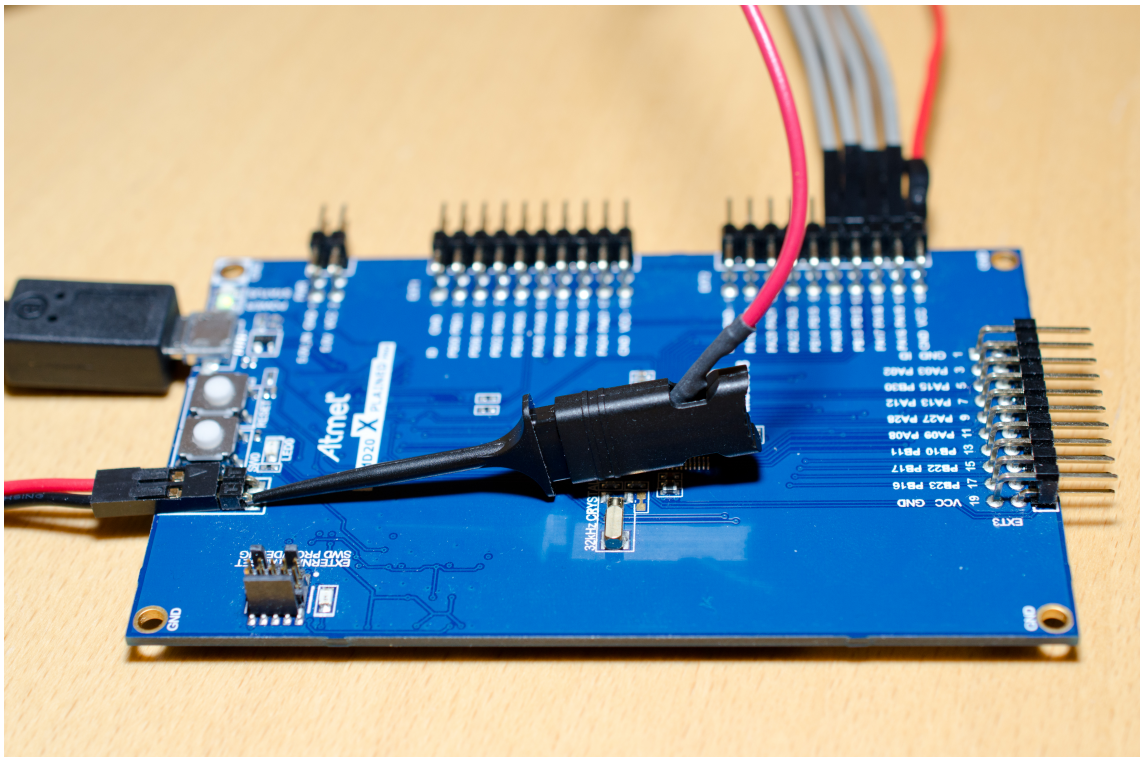
Utsendt effekt kan styres i 16 trinn, mellom −17dBm og +3dBm. Modulasjonsmetode er OQPSK[2], i henhold til IEEE-standard 802.15.4 [24, 23].

Et IEEE 802.15.4-kompatibelt nettverk på 2.4GHz-båndet opererer med en datarate på 250kbit/s. Selve radioen støtter en såkalt "High Data Rate"-modus, for applikasjoner som ikke trenger å oppfylle IEEE 802.15.4-standarden. High Data Rate tillater overføringshastigheter på inntil 2000kbit/s, men høyere datarater går på bekostning av mottakerens sensitivitet, som beskrevet i radioens datablad [39, side 137]. I denne oppgaven benyttes 250kbit/s.

Radio og mikrokontroller kommuniserer med hverandre via SPI[3]. SAM D20 Xplained Pro støtter SPI på alle sine utvidelsesporter, på pin 15 til pin 18. AT86RF231-kortet fra RZ600 har en 10-pins tilkoblingsport. Denne samsvarer med nedre halvdel (pin 11 til pin 20) av SAM D20 Xplained Pros utvidelsesporter, og kortene kan derfor kobles sammen direkte.

I denne oppgaven er det ønskelig å måle energiforbruket til mikrokontroller og radio samlet. Utvidelsesportenes pin 20, som samsvarer med radiomodulens VCC, går ikke via strømmålingskontakten beskrevet i 3.1.1. Radioen kobles derfor til mikrokontrolleren ved hjelp av kabler, som vist i figur 3.6. En klype, vist i figur 3.7, kobler radiomodulens VCC sammen med strømmålingskontaktens VCC_MCU. På denne måten vil et amperemeter tilkoblet strømmålingskontakten måle total strøm for mikrokontroller og radio.

[2]OQPSK: Offset Quadrature Phase Shift Keying.
[3]SPI: Serial Peripheral Interface



Figur 3.7: Klype for å hente ut VCC_MCU. (Foto: R. Bjønnum)

3.2 Programvare

3.2.1 Lightweight Mesh

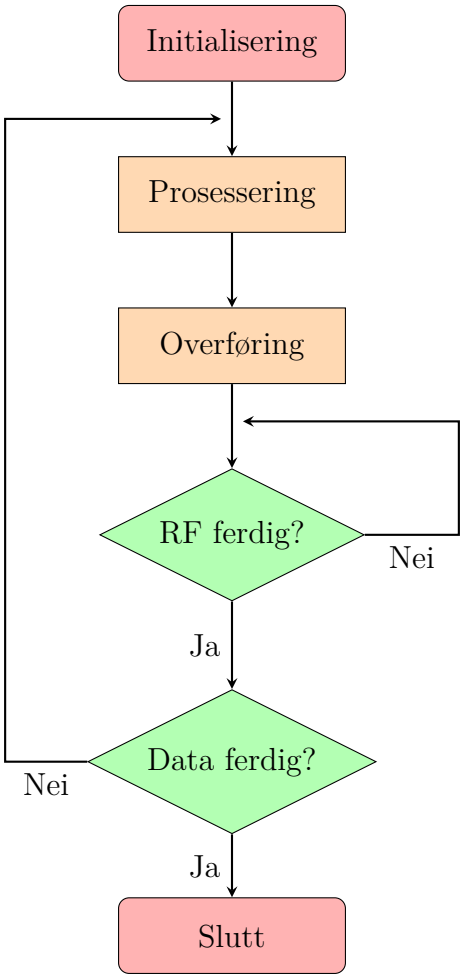
Nodene baseres på Atmels Lightweight Mesh-rammeverk, versjon 1.2.1. Rammeverket lastes ned fra Atmels nettsider [41] som en ZIP-fil. ZIP-filen inneholder, foruten selve rammeverket, all nødvendig dokumentasjon og noen eksempelapplikasjoner, som demonstrerer de ulike funksjonene rammeverket tilbyr.

3.2.2 ATMEL DSP Library

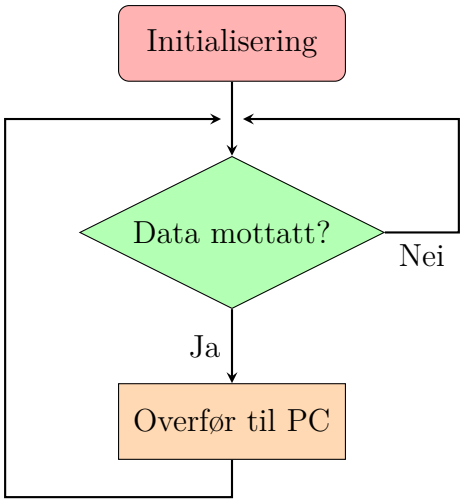
Atmel tilbyr et DSP-bibliotek [42] beregnet på deres UC3 32-bit AVR mikrokontrollere. DSP-biblioteket inneholder en implementasjon av IMA ADPCM-algoritmen, som med mindre tilpasninger også fungerer på andre prosessorarkitekturer. Denne implementasjonen av IMA ADPCM benyttes i sensornoden, og kan også kompileres for Windows, som beskrevet i 3.2.5.

3.2.3 Sensornode

Siden denne oppgaven kun omhandler energiforbruk ved prosessering og overføring av data, er sensorsystemet i sensornoden forenklet. I arbeidet med å implementere sensornoden ble det eksperimentert med ulike metoder for å mate noden med forhåndsgenererte data, som lesing av filer fra et microSD-kort og dataoverføring fra



Figur 3.8: Applikasjonsflyt sensornode.



Figur 3.9: Applikasjonsflyt mottakernode.

PC via UART. Ingen av disse metodene resulterte i ønsket ytelse, og det ble derfor valgt å hardkode et datasett på 2000 16bit samples inn i applikasjonen, slik at måleresultatene ikke skulle bli farget av et suboptimalt sensorsystem. Forhåndsgenererte data er ønskelig for å kunne sjekke om prosessering og overføring gjøres feilfritt; ellers kan mikrokontrollerens ADC benyttes.

Lightweight Mesh baserer seg på at applikasjonen deles inn i oppgaver, som gjennomføres sekvensielt i en uendelig løkke. LWM implementerer en oppgave som tar seg av nettverket, kalt `SYS_TaskHandler()`. Brukeren implementerer resten. Denne applikasjonen består av én oppgave kalt `APP_TaskHandler()`, som implementerer en enkel tilstandsmaskin. Applikasjonsflyten er illustrert i figur 3.8.

Initialisering innebærer oppsett av nettverk og andre enheter, som brytere og lysdioder. Radioen settes til full sendestyrke, og skrus deretter av. Applikasjonen venter så på et trykk på utviklingskortets SW0-knapp, som indikerer at prosessering kan starte.

Når prosesseringen starter, skrus radioen på, for å være klar til å overføre prosesserte data. Så snart en LWM-ramme er fylt opp, begynner overføringen. Applikasjonen venter så på bekreftelse fra mottakernoden på at pakken er mottatt (RF ferdig?). Dersom det ikke er mer data igjen å prosessere (Data ferdig?), skrus radioen av, og mikrokontrolleren går i strømsparemodus (Slutt). Ellers prosesseres neste pakke.

Dersom det skal prosesseres flere enn 2000 samples, gjøres dette ved å kjøre flere iterasjoner av prosessering av 2000 samples. I referansetesten og systemtesten prosesseres 8000 samples, som tilsvarer fire iterasjoner på 2000 samples hver.

3.2.4 Mottakernode

Mottakernoden har en noe enklere applikasjonsflyt, illustrert i figur 3.9. Initialisering innebærer oppsett av nettverk og UART for kommunikasjon med tilkoblet PC. UART opererer ved en hastighet på 115200Bd. 1Bd = 1symbol/s. UART overfører 1bit/symbol, og hastigheten blir derfor lik 115.2kbit/s, som er lavere enn radioens

oppgitte overføringshastighet. For å unngå at UART blir en flaskehals, benyttes et 16KiB mellomlager mellom radio og UART. Under testing overføres aldri mer enn $8000 \times 2\text{B} = 16\text{kB}$ per test, og mellomlageret skal derfor være stort nok.

Etter initialiseringen, venter noden på innkommende datapakker (Data mottatt?). Dersom en ny pakke er mottatt, overføres denne via UART til tilkoblet PC. Overføring til PC gjøres for å kunne verifisere at overføringen har gått feilfritt for seg, da det ikke er implementert støtte for retransmisjon av tapte datapakker i Lightweight Mesh. Som en bonus, kan man i tillegg sjekke om den valgte prosesseringsalgoritmen fungerer som den skal, ved å sammenligne mottatte data med en kjent referanse.

3.2.5 PC

MathWorks MATLAB R2013a benyttes til å generere et testsignal på 1000 samples, bestående av fem sinusfunksjoner med frekvenser $F_1 = f_\text{s}/4$, $F_2 = f_\text{s}/8$, $F_3 = f_\text{s}/16$, $F_4 = f_\text{s}/32$ og $F_5 = f_\text{s}/64$, der f_s er signalets punktprøvingsfrekvens. I denne oppgaven benyttes $f_\text{s} = 8\text{kHz}$. Sinusfunksjonene summeres, og det resulterende signalet normaliseres, slik at alle samples ligger i intervallet $[-32768, 32767]$. Testsignalet skrives til en tekstfil i form av kommaseparerte verdier, som etter mindre tilpasninger kan kopieres rett inn i et array i C-kode. Signalet blir duplisert for å få en lengde på 2000 samples, og lagt inn i et array i filen `data.h`. `data.h` blir så inkludert i kildekoden til sensornoden. For kildekode til testsignalgeneratoren, se `Tonegenerator.m` i vedlegg A.

Alle testene i denne oppgaven forutsetter feilfri overføring av data. To verktøy benyttes for å sjekke om en overføring er feilfri: Microsoft-verktøyet `fc` og det egenutviklede verktøyet `SerialDump`. Begge programmene kjøres i kommandolinjen (`cmd`) på en datamaskin med Microsoft Windows 8.1.

`SerialDump` lytter til en gitt serieport, og lagrer innkommende data i en fil. For kildekode til `SerialDump`, se `SerialDump` i vedlegg A. Følgende kommando benyttes for å kjøre verktøyet:

```
#   SerialDump <filnavn> <port>
```

`<filnavn>` angir hvor innkommende data skal lagres. `<port>` angir hvilken serieport det skal lyttes til, for eksempel `COM3`.

Mottakernoden kobles til datamaskinen via USB. Alle innkommende data fra radioen videresendes til datamaskinen ved hjelp av feilsøkingsverktøyets virtuelle serieport. `SerialDump` kjøres, og den trådløse dataoverføringen startes.

Den resulterende filens størrelse sjekkes manuelt, for å enkelt se om datapakker har gått tapt. Dersom størrelsen er riktig, benyttes `fc` til å sammenligne filen med en kjent referanse. Dersom filene er like, har overføringen gått feilfritt. `fc` brukes på følgende måte:

```
#   fc /b <referanse> <mottatt>
```

`<referanse>` angir referansefilens plassering, og `<mottatt>` angir filen som ble generert ved hjelp av `SerialDump`. Parameteren `/b` tvinger `fc` til å tolke filene som binære filer; i utgangspunktet tolkes alle filer, med noen unntak, som tekst [43].

Referansefilen genereres ved hjelp av verktøyet `WriteRaw`, se vedlegg A for kildekode. Følgende kommando benyttes for å kjøre verktøyet:

```
#  WriteRaw <filnavn> <lengde>
```

`<filnavn>` angir filen det skal skrives til. `<lengde>` angir filens lengde i B. Maksimal verdi for `<lengde>` er 16000, som resulterer i en fil på 8000 samples. I referansetesten benyttes en fil bestående av 8000 samples som referansefil, generert ved hjelp av følgende kommando:

```
#  WriteRaw ref_8k_smp_raw.bin 16000
```

Filer generert av `WriteRaw` kan spilles av på PC-ens høyttalere ved hjelp av MATLAB-funksjonen `playSound.m`, se vedlegg A for kildekode. Denne benyttes ved hjelp av følgende kommando i MATLABs kommandolinje:

```
#  playSound('<filnavn>');
```

`<filnavn>` angir filen som skal spilles av.

Referansefilen til systemtesten konverteres fra `ref_8k_smp_raw.bin` ved hjelp av verktøyet `adpcm`, som består av den samme IMA ADPCM-algoritmen som benyttes i sensornoden, pakket inn i en applikasjon og kompilert for Windows. Se vedlegg A for kildekode. Følgende kommando benyttes for å kjøre verktøyet:

```
#  adpcm <e | d> <kilde> <destinasjon>
```

`<e | d>` angir om kildefilen skal enkodes (e) eller dekodes (d). `<kilde>` angir kildefilens plassering. `<destinasjon>` angir destinasjonsfilens plassering. Referansefilen til systemtesten genereres ved følgende kommando:

```
#  adpcm e ref_8k_smp_raw.bin ref_8k_smp_adpcm.bin
```

3.3 Måleutstyr

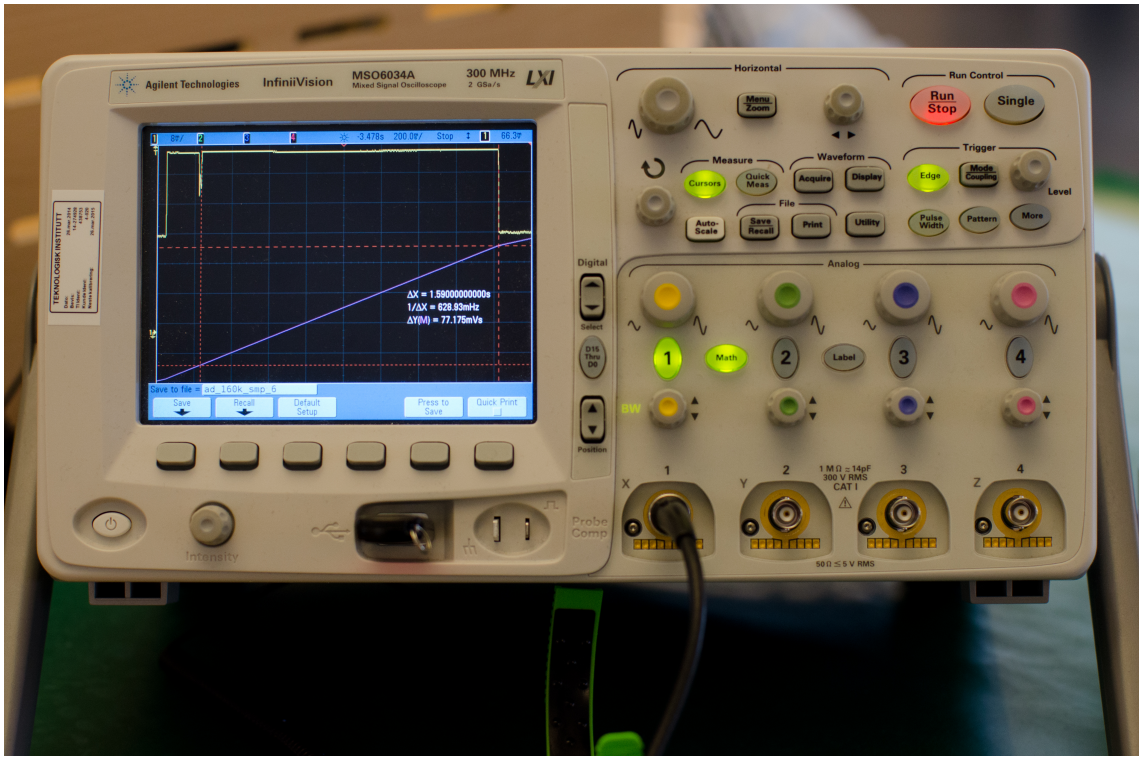
3.3.1 Multimeter

Enkle kontrollmålinger av spenning og strøm gjøres ved hjelp av et Tenma 72-7735 multimeter, avbildet i figur 3.10. Databladet [44] oppgir en nøyaktighet på $\pm(0.8\% + 1)$ ved måling av likespenning i størrelsesorden 4V. Ved måling av likestrøm i størrelsesorden 400µA og 4000µA, skal nøyaktigheten være $\pm(1\% + 2)$. I størrelsesorden 40mA og 400mA reduseres nøyaktigheten til $\pm(1.2\% + 3)$.

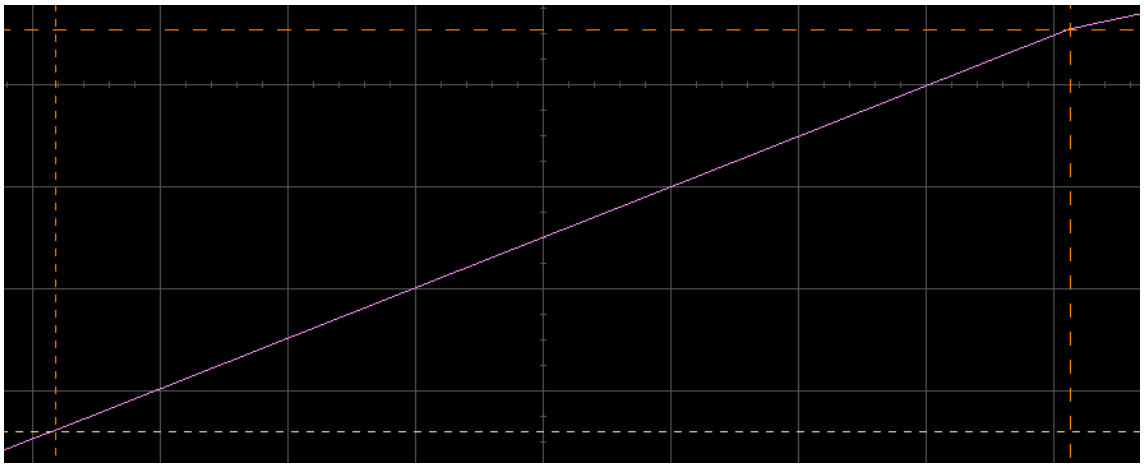
3.3.2 Oscilloskop

For å måle spenning og strøm over tid, benyttes et Agilent Technologies MSO6034A oscilloskop, avbildet i figur 3.11. Dette er et 4-kanals oscilloskop med 300MHz båndbredde, og punktprøvingshastighet på 2GSa/s. Det benyttede eksemplaret har serienummer MY49400110.

Figur 3.10: Tenma 72-7735 multimeter. (Foto: R. Bjønnum)



Figur 3.11: Agilent Technologies MSO6034A oscilloskop. (Foto: R. Bjønnum)



Figur 3.12: Pekere og integralfunksjon på oscilloskop.

Oscilloskopet tillater å regne ut et gjennomsnitt av flere målinger, for å redusere påvirkning fra støy. I denne oppgaven benyttes et gjennomsnitt av to målinger, som beskrevet i oscilloskopets brukermanual [45, side 267].

Under testing har det vist seg at radioen har et svært ujevnt strømforbruk når den overfører data, som fører til hurtige variasjoner i målinger foretatt med oscilloskop. Dersom vi måler en spenning $V_\mathrm{m}(t)$, finnes et gjennomsnitt av denne spenningen, $\overline{V_\mathrm{m}}$, over tidsintervallet $\Delta t = t_2 - t_1$, ved ligning 3.1.

$$\overline{V_\mathrm{m}} = \frac{1}{\Delta t} \int_{t_1}^{t_2} V_\mathrm{m}(t)\,\mathrm{d}t \tag{3.1}$$

Ligning 3.1 lar seg enkelt implementere på oscilloskopet ved hjelp av to funksjoner: pekere (cursors) [45, side 214] og integral [45, side 232]. Signalet $V_\mathrm{m}(t)$ velges som kilde for integralfunksjonen. Skopet settes opp til å vise tid på X-aksen og resultatet av integralfunksjonen på Y-aksen. Pekerne settes opp til å vise verdier fra integralfunksjonen, ved å sette pekernes kilde til "Math".

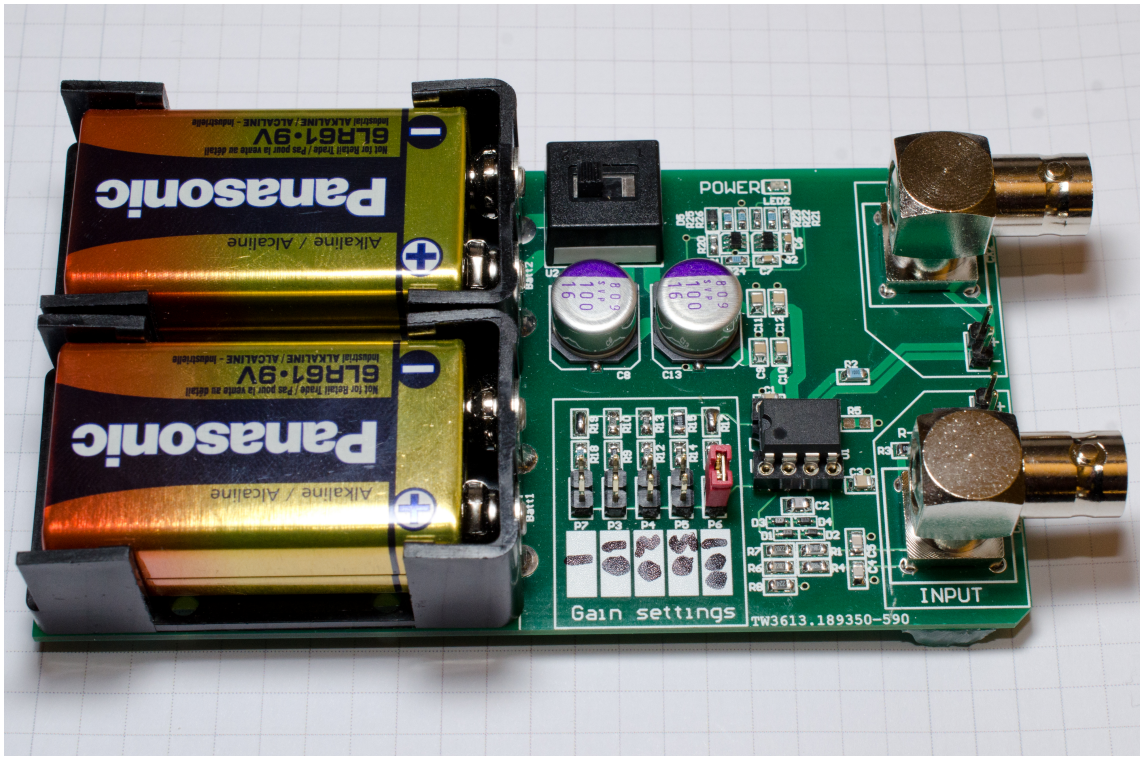
To pekere kan plasseres på hver akse. X-pekerne (vertikale linjer) plasseres i tidspunktene t_1 og t_2, og Y-pekerne (horisontale linjer) plasseres der resultatet av integralfunksjonen krysser X-pekerne, illustrert i figur 3.12. Pekerne vises som stiplede linjer, integralfunksjonen som heltrukken linje diagonalt i figuren.

Skopet beregner avstanden mellom pekerne for hver akse, og presenterer disse som henholdsvis ΔX og ΔY. ΔX på skopet tilsvarer Δt i ligning 3.1. ΔY på skopet tilsvarer $\int_{t_1}^{t_2} V_\mathrm{m}(t)\,\mathrm{d}t$. $\overline{V_\mathrm{m}}$ finnes enkelt ved ligning 3.2, der ΔX og ΔY leses direkte ut av skopet.

$$\overline{V_\mathrm{m}} = \frac{\Delta Y}{\Delta X} \tag{3.2}$$

3.3.3 Strømprobe

Mellom utviklingskort og oscilloskop kobles en strømprobe, som vist i figur 3.13. Denne består av en motstand på $R = 1\Omega$, som kobles i serie med kretsen som skal



Figur 3.13: Strømprobe. (Foto: R. Bjønnum)

måles. Spenningsfallet over denne motstanden, V_{R}, forsterkes av en differensialforsterker, implementert ved hjelp av en THAT Corporation THAT1512P08-U [46] operasjonsforsterker. Differensialforsterkerens utgangsspenning, V_{D}, måles ved hjelp av oscilloskopet. Probens forsterkning, A, kan velges i fem trinn, der laveste ikke gir noen forsterkning ($A = 1$), og høyeste trinn gir 100 ganger forsterkning ($A = 100$). Forholdet mellom V_{R} og V_{D} er gitt ved ligning 3.3.

$$V_{\mathrm{D}} = AV_{\mathrm{R}} \tag{3.3}$$

Sammenhengen mellom kretsens strømforbruk, I_{R}, og spenningsfall over målemotstanden, V_{R}, er gitt ved ligning 3.4.

$$V_{\mathrm{R}} = I_{\mathrm{R}}R \tag{3.4}$$

Ved å sette ligning 3.4 inn i ligning 3.3, finnes sammenhengen mellom differensialforsterkerens utgangsspenning, V_{D}, og kretsens strømforbruk, som vist i ligning 3.5.

$$V_{\mathrm{D}} = AI_{\mathrm{R}}R \tag{3.5}$$

Ligning 3.5 løses med hensyn på I_{R} for å finne kretsens strømforbruk ut fra målt V_{D}. Dette resulterer i ligning 3.6.

$$I_{\mathrm{R}} = \frac{V_{\mathrm{D}}}{AR} \tag{3.6}$$

Basert på datablad for mikrokontroller og radio [34, 39] forventes det at systemets strømforbruk kan komme opp i størrelsesorden 20mA. Dette gir et spenningsfall

over målemotstanden på $V_R = 20\text{mV}$, i henhold til ligning 3.4. Differensialforsterkeren forsynes med ±9V, som medfører at maksimal utgangsspenning fra proben er omtrent ±8V (databladet til den benyttede operasjonsforsterkeren [46] oppgir maksimalt ±13.3V utsving ved ±15V forsyningsspenning). Ved $V_R = 20\text{mV}$ og $A = 100$ blir $V_D = 2000\text{mV} = 2\text{V}$, og $A = 100$ kan derfor trygt benyttes i denne oppgaven.

Koblingsskjema for strømproben er gjengitt i vedlegg B.

3.4 Målemetode

3.4.1 Kompleksitetstest

Antall klokkesykler en prosessor trenger for å gjennomføre en gitt algoritme, gir et godt mål på algoritmens kompleksitet. ARM Cortex M0+ kan, som beskrevet i [47, kap. 5.1.1], ha en 24bit systemtimer (SysTick). En slik er tilgjengelig på Atmel SAM D20 [34, kap. 10.1.1]. SysTick teller ned til null fra en programmerbar startverdi, synkront med prosessorens klokke. Denne telleren kan, under forutsetninger beskrevet nedenfor, benyttes til å gi et godt estimat av algoritmens kjøretid. Metoden er beskrevet i algoritme 3.1 (pseudokode).

Algoritme 3.1 Estimering av kjøretid med SysTick.

Parametre: $0 <$ startverdi $< 2^{24}$
  Sett SysTick startverdi(startverdi)
  Nullstill SysTick-teller
  Start teller
  Kjør algoritme
  Stopp teller
  kjøretid ← startverdi − teller
  return kjøretid

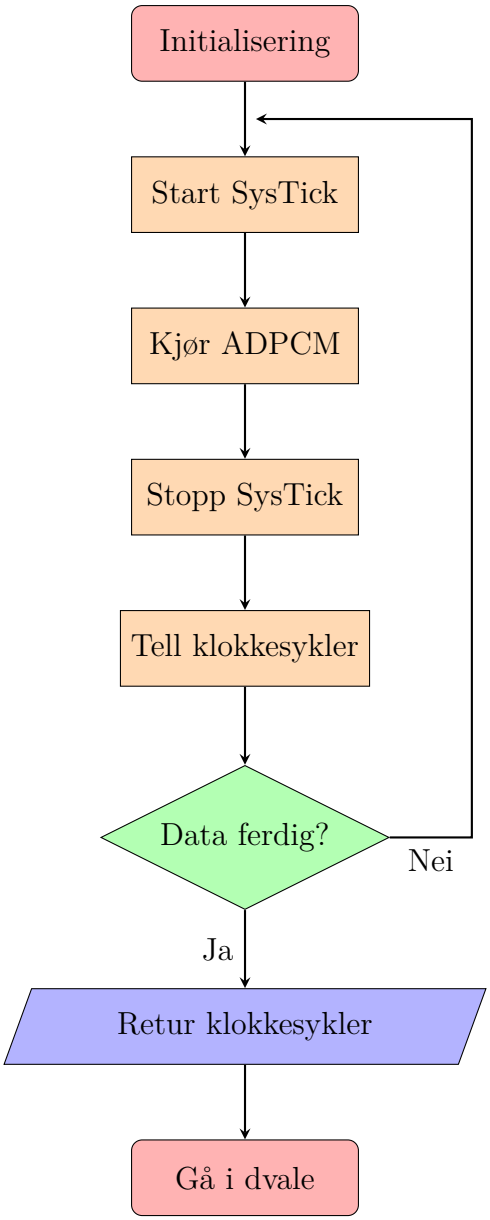
Algoritme 3.1 begynner med å sette SysTick-tellerens startverdi. Startverdien er et positivt 24bit heltall uten fortegn, og høyeste tillatte verdi er dermed $2^{24} - 1 = 16777215$. Deretter blir telleren først nullstilt, så startet. Algoritmen som skal måles blir kjørt, og SysTick-telleren blir stoppet. Valgt algoritmes kjøretid i klokkesykler, C_A, finnes så ved hjelp av ligning 3.7, der C_{start} er SysTick-tellerens startverdi, og C_{stopp} er SysTick-tellerens verdi etter å ha blitt stoppet.

$$C_A = C_{start} - C_{stopp} \tag{3.7}$$

Start og stopp av telleren krever noen klokkesykler, så metoden er ikke helt nøyaktig, men den gir likevel et godt bilde på valgt algoritmes kjøretid.

Med 8MHz klokkefrekvens, tar det telleren $\frac{16777215}{8000000} \approx 2.097\text{s}$ å telle ned fra høyeste tillatte verdi til null. Når telleren treffer nullpunktet, vil den enten utløse et interrupt[4], eller sette et flagg i et register, før den nullstiller seg selv, og starter på nytt.

[4]Interrupt: et signal til prosessoren, som indikerer en hendelse som krever umiddelbar oppmerksomhet.



Figur 3.14: Applikasjonsflyt `SysTickTest`.

Ved implementasjon av algoritme 3.1 uten modifikasjon, vil en omstart av telleren få fatale konsekvenser for måleresultatet. Denne metoden kan derfor kun benyttes sammen med algoritmer der forventet kjøretid (antall klokkesykler) er lavere enn valgt startverdi for SysTick-telleren.

Kompleksitetstesten er implementert som en egen applikasjon, kalt `SysTickTest`. Applikasjonen er basert på Atmel Software Framework [48], som er en samling drivere og biblioteker for hurtig programvareutvikling for Atmels mikrokontrollere.

Applikasjonsflyten for `SysTickTest` er vist i figur 3.14. Den starter med initialisering av systemet, setter opp UART for kommunikasjon med PC, og setter opp nødvendige variabler. Deretter startes SysTick-telleren, og dataprosesseringen begynner. 2000 16bit samples prosesseres av IMA ADPCM-algoritmen. SysTick-timeren stoppes, og antall brukte klokkesykler akkumuleres i variabelen `totalCycles`. Hele prosessen fra og med start av SysTick, kjøres i en løkke på 80 iterasjoner, slik at totalt 80 $\times$

2000 = 160000 samples blir prosessert. Dersom siste iterasjon av løkken er ferdig (Data ferdig?), returneres akkumulert antall klokkesykler til PC via UART, og leses enkelt av ved hjelp av en terminalemulator, som for eksempel `PuTTY`. Til slutt går mikrokontrolleren i dvale.

Resultatet av kompleksitetsanalysen kan benyttes til å avgjøre om den valgte algoritmen kan kjøre i sanntid på en gitt plattform med en gitt konfigurasjon. Dette gjøres ved å finne algoritmens gjennomsnittlige prosesseringstid per sample, C_{avg}, målt i klokkesykler, som vist i ligning 3.8. C_{A} er prosesseringsalgorigmens kjøretid, som beskrevet i ligning 3.7, ved prosessering av N samples.

$$C_{\text{avg}} = \frac{C_{\text{A}}}{N} \tag{3.8}$$

Ved klokkefrekvens f_{CPU}, tilsvarer C klokkesykler en tid t, som vist i ligning 3.9.

$$t = \frac{C}{f_{\text{CPU}}} \tag{3.9}$$

Ved å benytte ligning 3.9 og 3.8, finnes gjennomsnittlig prosesseringstid per sample, t_{avg}, målt i sekunder og vist i ligning 3.10.

$$t_{\text{avg}} = \frac{C_{\text{A}}}{N f_{\text{CPU}}} \tag{3.10}$$

Kildesignalets punktprøvingsfrekvens f_{s} definerer en nedre grense for hvor mange samples algoritmen må greie å ta unna per sekund, for å kunne operere i sanntid. Dette gir en øvre grense $\frac{1}{f_{\text{s}}}$ for prosesseringstid per sample. Kravet for at algoritmen kan operere i sanntid, forutsatt at prosessoren ikke har andre oppgaver, er formalisert ved ligning 3.11.

$$t_{\text{avg}} < \frac{1}{f_{\text{s}}} \tag{3.11}$$

3.4.2 Energiforbruk

Ligning 2.4 på side 8, gjengitt her for oversiktens skyld, beskriver energiforbruket til en krets.

$$E = VIt \tag{2.4}$$

Under alle tester forsynes utviklingskortet med strøm gjennom USB-kontakten. På det aktuelle kortet ble spenningen VCC_MCU målt til å være 3.288V, med prosessoren i våken tilstand, og strømproben tilkoblet strømmålingskontakten. Spenningen $V = 3.288\text{V}$ blir derfor benyttet ved alle videre beregninger.

Ligning 3.6 på side 21 viser hvordan kretsens strømforbruk henger sammen med en spenning på utgangen til strømproben, som måles ved hjelp av oscilloskopet. Proben benytter en forsterkning $A = 100$ og en målemotstand $R = 1\Omega$. I_{R} settes inn for I i ligning 2.4.

$$I_{\text{R}} = \frac{V_{\text{D}}}{AR} \tag{3.6}$$

Ligning 3.1 og 3.2 på side 20 viser hvordan oscilloskopets integralfunksjon og pekere kan benyttes til å finne gjennomsnittet av en målt spenning over et gitt tidsrom.

Skopet måler V_{D}, og integralfunksjonen settes opp til å integrere denne. $\overline{V_{\mathrm{m}}}$ gir da en gjennomsnittsverdi av V_{D} i tidsintervallet ΔX. $\overline{V_{\mathrm{m}}}$ settes derfor inn for V_{D} i ligning 3.6.

$$\overline{V_{\mathrm{m}}} = \frac{\Delta Y}{\Delta X} \tag{3.2}$$

ΔX defineres til å være lik kjøretiden til applikasjonen, t, siden det er ønskelig å vite gjennomsnittlig strømforbruk når applikasjonen kjører. I praksis løses dette ved å plassere pekerne på X-aksen til oscilloskopet i applikasjonens start- og sluttpunkt. Sensornoden aktiverer radioen med en gang den begynner å prosessere data, og deaktiverer den igjen når all overføring er ferdig. Tidspunkt for aktivering og deaktivering av radioen vises tydelig som henholdsvis stigende og synkende flanker i den målte spenningen V_{D}. Disse flankene angir applikasjonens start- og sluttpunkt. Ved å plassere pekerne på X-aksen på hver sin flanke, finnes enkelt kjøretiden til applikasjonen. ΔX settes inn for t i ligning 2.4.

Ved å sette inn $\overline{V_{\mathrm{m}}}$ for V_{D}, I_{R} for I og ΔX for t, oppnås ligning 3.12, som beskriver en applikasjons energiforbruk ved hjelp av målinger fra oscilloskop. Denne ligningen benyttes til alle energiberegninger i kapittel 4, med $V = 3.288\mathrm{V}$, $A = 100$ og $R = 1\Omega$.

$$E = \frac{V}{AR}\Delta Y \tag{3.12}$$

Rent praktisk gjennomføres energimålingene ved å starte mottakernoden og `SerialDump`, som beskrevet i 3.2.5. Oscilloskopet settes i gang med å måle, og sensornoden startes. Når sensornoden er ferdig med sine oppgaver, og går i dvale, stoppes datainnsamlingen på oscilloskopet. I etterpåklokskapens ånd burde start og stopp av datainnsamling blitt gjort ved å koble opp en GPIO-pinne på mikrokontrolleren som ekstern trigger på oscilloskopet. Mikrokontrolleren setter GPIO-pinnen til logisk høy før og etter kjøring av applikasjonen, og logisk lav i det applikasjonen starter.

Etter datainnsamling, avsluttes `SerialDump`, og mottatte data sjekkes for feil, som beskrevet i 3.2.5. Ved feilfrie data, plasseres oscilloskopets pekere, som beskrevet i 3.3.2, og ΔY leses av og settes inn i ligning 3.12. Dersom feil oppstår, startes testen på nytt.

Alle testene kjøres flere ganger, for å sikre at resultatene som presenteres i kapittel 4 er representative.

4 Resultater

4.1 Kompleksitet IMA ADPCM

Kjøring av `SysTickTest` gir kjøretiden $C_{160k} = 12754725$ klokkesykler ved prosessering av 160000 samples. Dette gir en gjennomsnittlig prosesseringstid på $C_{avg} = \frac{12754725\,\text{sykler}}{160000\,\text{sample}} \approx 79.72\,\text{sykler/sample}$.

Et tidligere arbeide estimerer at datakompresjon lønner seg foran kommunikasjon på den benyttede plattformen, dersom prosessering koster mindre enn $C_1 = 192$ klokkesykler for hver sparte bit [19]. IMA ADPCM komprimerer et signal med 16bit/sample ned til 4bit/sample, og sparer på denne måten $16 - 4 = 12$bit/sample. Kostnaden for IMA ADPCM er dermed $C_{IMA} = \frac{79.72\,\text{sykler/sample}}{12\text{bit/sample}} \approx 6.64\,\text{sykler/bit}$.

Selv om C_1 er basert på grove estimater [19], er C_{IMA} såpass mye lavere enn C_1 at det kan forventes å være energimessig lønnsomt å benytte IMA ADPCM i stedet for LPCM.

Ligning 3.11 på side 24 benyttes for å avgjøre om IMA ADPCM kan kjøres i sanntid på sensornoden. t_{avg} finnes ved å sette inn C_{160k} for C_A, $N = 160000$ og $f_{CPU} = 8$MHz i ligning 3.10, som gir at $t_{avg} \approx 9.96$µs. For $f_s = 8$kHz er $\frac{1}{f_s} = 125$µs, og kravet i ligning 3.11 er dermed tilfredsstilt. IMA ADPCM kan derfor kjøres i sanntid på sensornoden ved $f_s = 8$kHz, dersom den ikke har andre oppgaver.

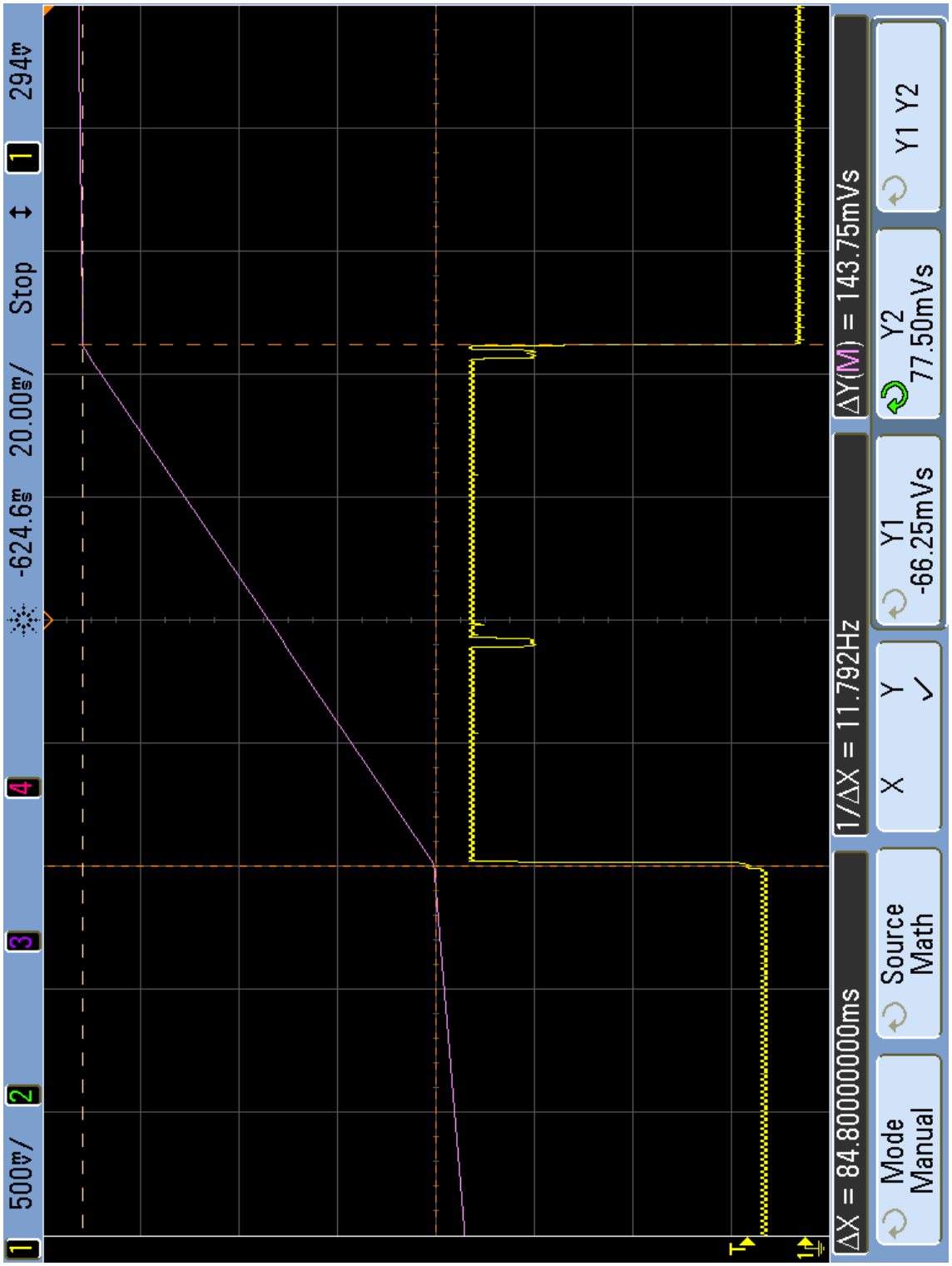
Ved å løse ligning 3.11 for f_s, og sette likhetstegn mellom de to sidene av ulikheten, finnes et uttrykk for maksimal punktprøvingsfrekvens ved sanntidsoperasjon, f_{smax}, som vist i ligning 4.1.

$$f_{smax} = \frac{1}{t_{avg}} \tag{4.1}$$

$t_{avg} \approx 9.96$µs gir at sensornoden skal klare sanntids IMA ADPCM-enkoding for $f_s < f_{smax} = 100$kHz ved $f_{CPU} = 8$MHz, dersom den ikke har andre oppgaver.

4.2 Prosesseringstest

Ved $t_{avg} \approx 9.96$µs, er forventet prosesseringstid for IMA ADPCM-enkoding av 8000 samples $t_{8k} = t_{avg} \times 8000 \approx 80$ms. For å verifisere dette, settes sensornoden opp



Figur 4.1: Prosessering av 8000 samples, 16bit LPCM til 4bit IMA ADPCM.

til å prosessere 8000 samples med IMA ADPCM, uten å overføre disse via radio. Radioen skrus på før prosesseringen begynner, og skrus av igjen når den er ferdig, for at det skal være enkelt å se på oscilloskopet når applikasjonen starter og slutter. Siden mikrokontrolleren håndterer start og stopp av radio i tillegg til prosessering, forventes det at målt tidsforbruk er noe høyere enn 80ms.

Skjermskudd fra oscilloskop i figur 4.1 viser at prosessering av 8000 samples i tillegg til start og stopp av radio tar omtrent 84.8ms. $\Delta Y = 143.75\text{mV}\,\text{s}$ settes inn i ligning 3.12, og resultatet blir $E_{\text{pro}} = 4.7\text{mJ}$. E_{pro} er energiforbruk til hele systemet under prosesseringstesten, og kan ikke tolkes som energiforbruk til IMA ADPCM-algoritmen.

4.3 Referansetest

En referansetest gjennomføres for å finne ut hvor mye energi sensornode-implementasjonen bruker på overføring av 8000 samples, 16bit = 2B LPCM. Sensornoden settes opp til å overføre 8000 samples via radio, uten å benytte IMA ADPCM. Som vist i 2.4, er maksimal nyttelast per Lightweight Mesh-ramme $D_{\text{Nmax}} = 109\text{B}$. Dette tilsvarer $\frac{109}{2} = 54.5$ samples per ramme. Applikasjonen runder dette ned til 54 samples per ramme, for å unngå å splitte samples mellom to LWM-rammer. 8000 samples fordeles på $4 \times \left\lceil \frac{2000}{54} \right\rceil = 152$ rammer.

Figur 4.2 viser skjermskudd fra oscilloskop etter gjennomføring av referansetesten, der X- og Y-pekere har blitt plassert for å måle energiforbruk, som beskrevet i 3.4.2. $\Delta Y = 3.2550\text{V}\,\text{s}$ settes inn i ligning 3.12, og resultatet blir $E_{\text{ref}} = 107\text{mJ}$.

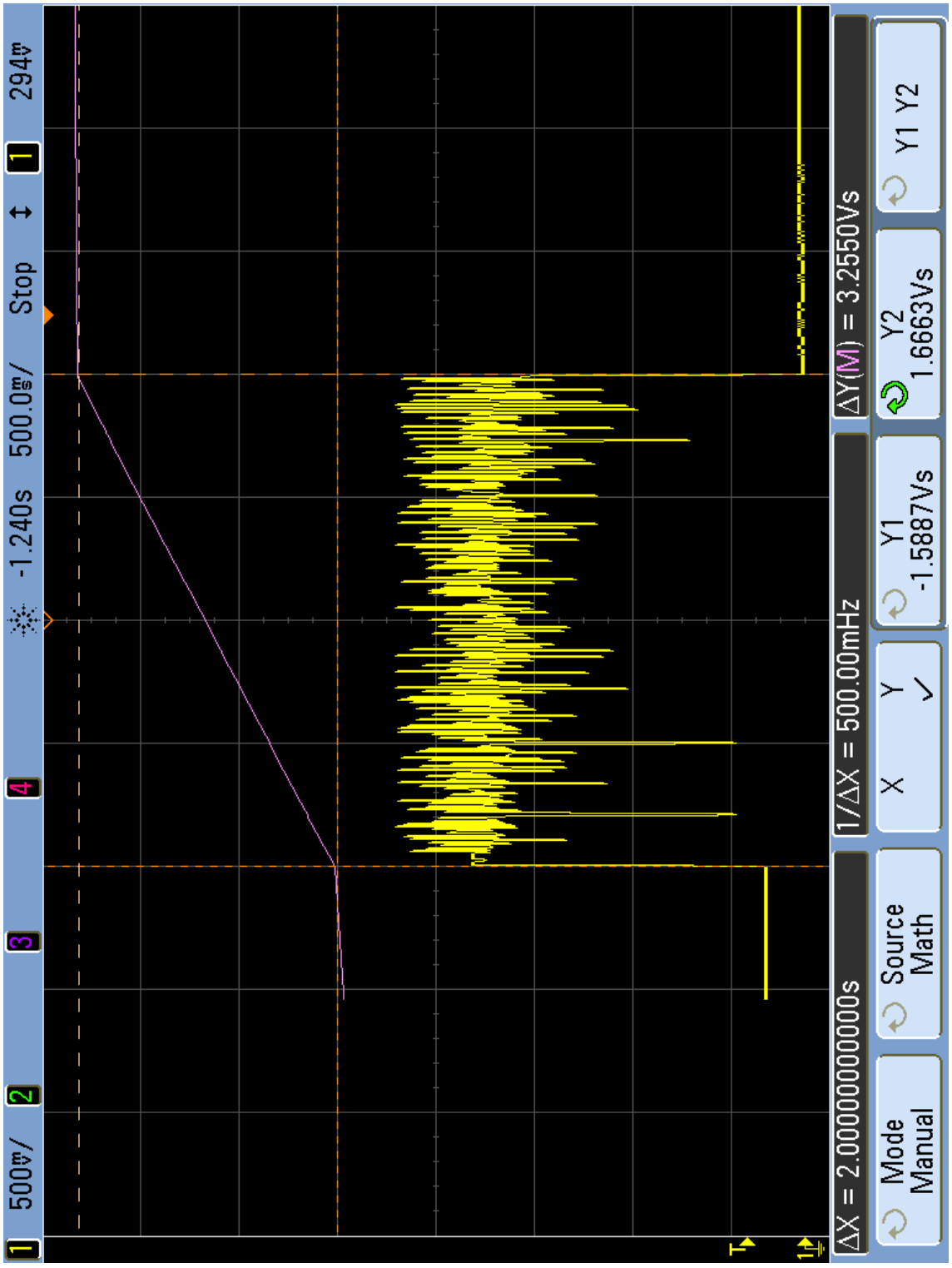
Figur 4.2 viser at kjøretiden for referansetesten er på hele 2s. Overføring av 8000 samples på 16bit/sample i løpet av 2s betyr en gjennomsnittlig rate for overføring av nyttelast på $R_{\text{netto}} = 64\text{kbit/s}$.

Som vist i 2.2, overføres en del metadata sammen med nyttelasten for hver LWM-ramme. LWM legger til 7B metadata per ramme. MAC-laget legger til 11B per ramme. PHY-laget legger til 6B per ramme. Totalt utgjør dette 24B metadata per ramme. Ved overføring av 152 rammer, overføres totalt $24\text{B} \times 152 = 3648\text{B}$ metadata.

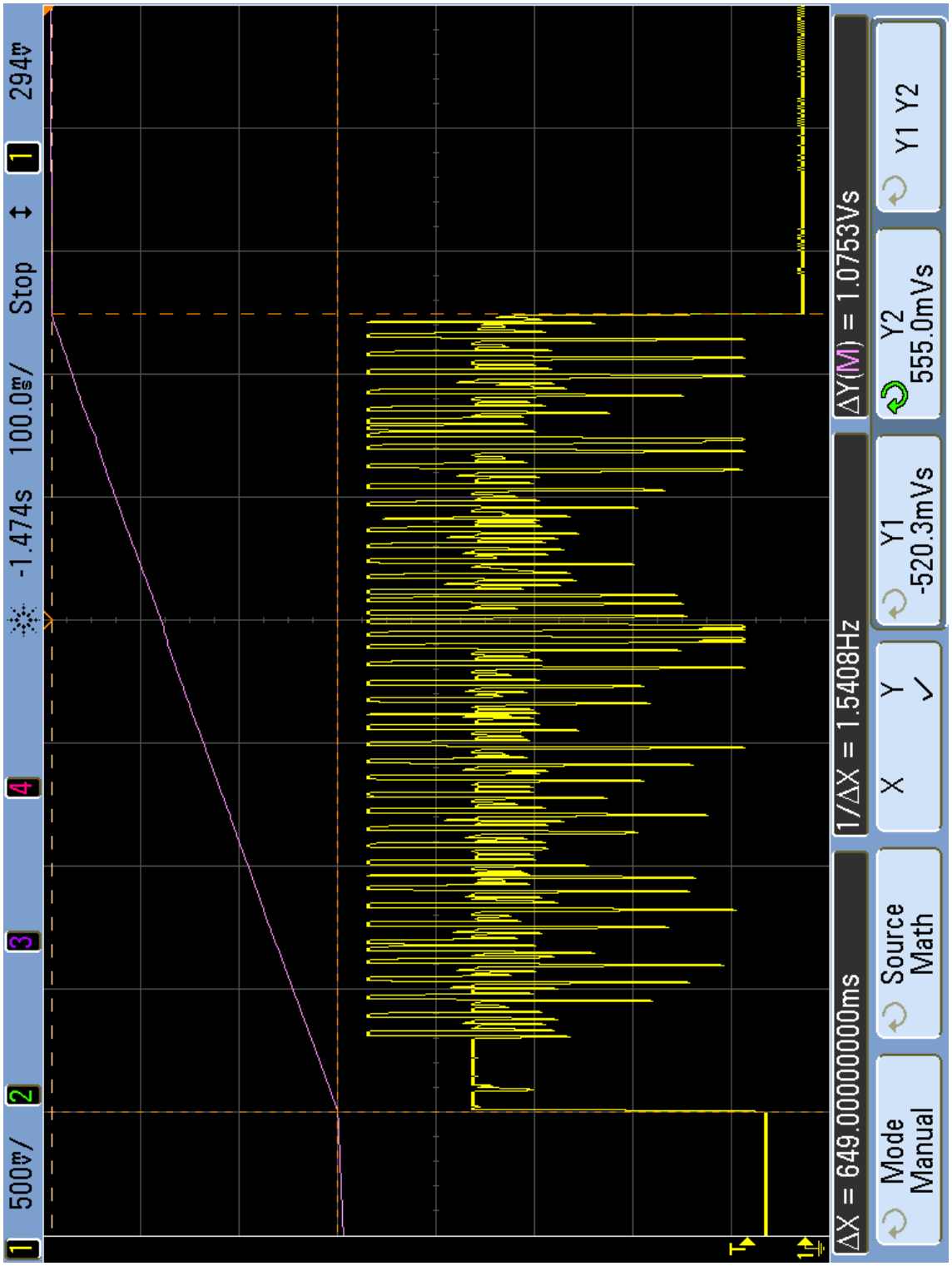
Summering av nyttelast og metadata gir at det overføres totalt $3648 + 2 \times 8000 = 19648\text{B} = 157184\text{bit}$ i løpet av to sekunder. Dette gir en gjennomsnittlig brutto datarate på $R_{\text{brutto}} = \frac{157184\text{bit}}{2\text{s}} = 78.592\text{kbit/s}$ i løpet av denne testen.

4.4 Systemtest

En systemtest gjennomføres for å finne energiforbruket til sensornoden ved overføring av 8000 samples, som først prosesseres ved hjelp av IMA ADPCM-algoritmen. Sensornoden settes opp til å prosessere og overføre 8000 samples. Hver prosesserte sample tar 4bit, og det er derfor mulig å overføre $\frac{109}{0.5} = 218$ samples per ramme. 8000 samples fordeles på $4 \times \left\lceil \frac{2000}{218} \right\rceil = 40$ rammer.



Figur 4.2: Overføring av 8000 samples, 16bit LPCM.



Figur 4.3: Prosessering og overføring av 8000 samples, 4bit IMA ADPCM.

Figur 4.3 viser skjermskudd fra oscilloskop etter gjennomføring av systemtesten. $\Delta Y = 1.0753\text{V}\,\text{s}$ settes inn i ligning 3.12, og resultatet blir $E_{\text{sys}} = 35.4\text{mJ}$.

Figur 4.3 viser at kjøretiden for systemtesten er på 649ms. Overføring av 8000 samples på 4bit/sample i løpet av 649ms betyr en gjennomsnittlig rate for overføring av nyttelast på $R_{\text{netto}} \approx 49.3\text{kbit/s}$. I løpet av denne tiden overføres $24\text{B} \times 40 = 960\text{B}$ metadata. Dette gir en gjennomsnittlig brutto datarate på $R_{\text{brutto}} = \frac{39680\text{bit}}{649\text{ms}} = 61.14\text{kbit/s}$ i løpet av denne testen.

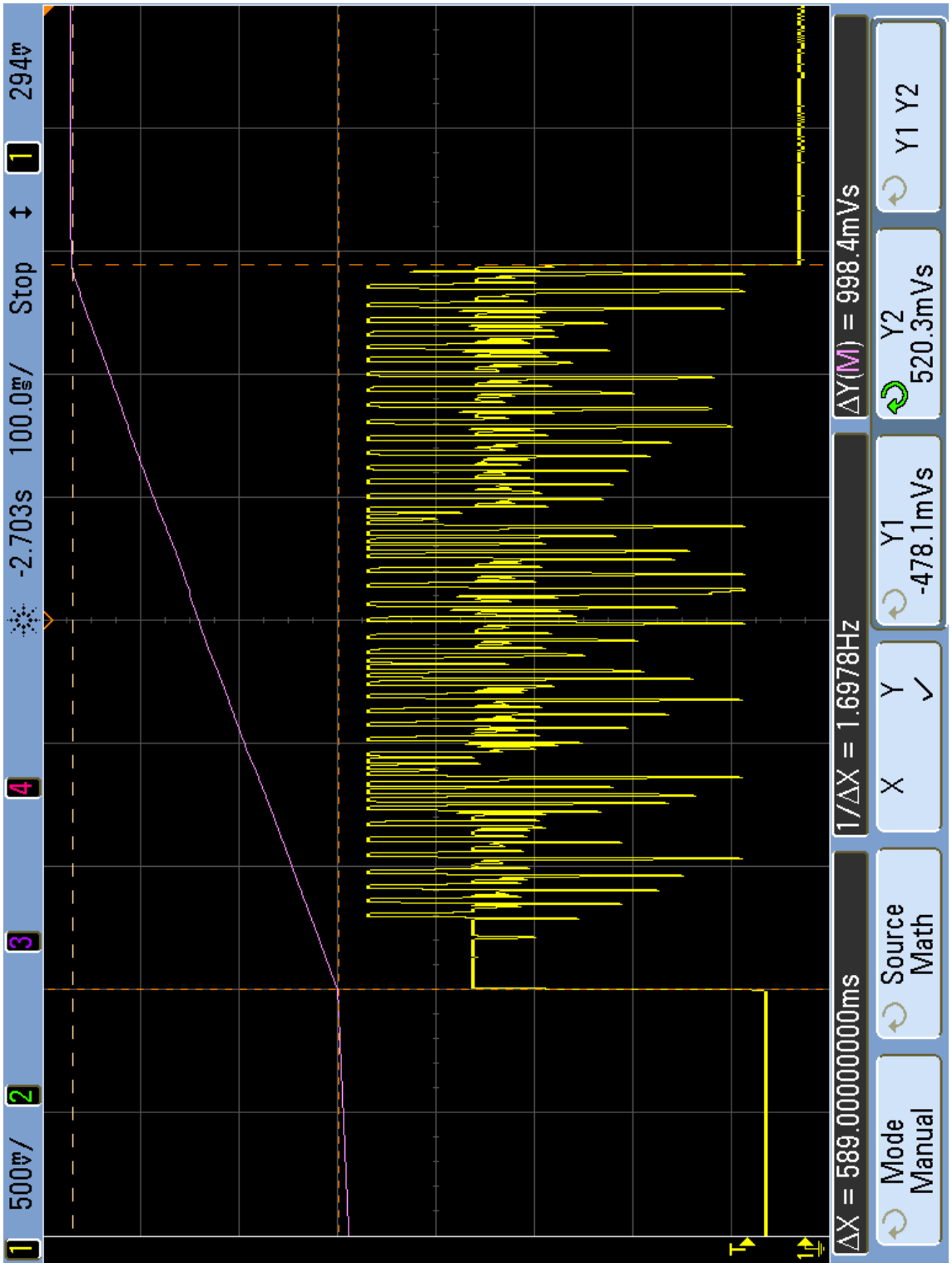
4.5 Overføringstest

En overføringstest gjennomføres for å finne energiforbruket til sensornoden ved overføring av 8000 samples, som på forhånd har blitt prosessert ved hjelp av IMA ADPCM-algoritmen. 8000 4bit samples tilsvarer 4kB data, som igjen tilsvarer 2000 16bit samples. Overføring av 8000 4bit samples er derfor ekvivalent med overføring av 2000 16bit samples.

Sensornoden settes opp til å overføre 2000 16bit samples. Det overføres 54 samples per ramme. 2000 samples fordeles på $\left\lceil \frac{2000}{54} \right\rceil = 38$ rammer.

Figur 4.4 viser skjermskudd fra oscilloskop etter gjennomføring av overføringstesten. $\Delta Y = 998.4\text{mV}\,\text{s}$ settes inn i ligning 3.12, og resultatet blir $E_{\text{ovr}} = 32.8\text{mJ}$.

I denne testen er det gjort en metodefeil, som gjør at resultatet ikke er like nøyaktig som for de andre testene. Dersom 8000 4bit samples hadde blitt overført, hadde alle 109B nyttelast per ramme blitt utnyttet. Ved overføring av 2000 16bit samples, benyttes $2 \times 54 = 108\text{B}$ av nyttelasten. Ved å benytte alle 109B, hadde 2000 16bit samples blitt fordelt på $\left\lceil \frac{2000}{54.5} \right\rceil = 37$ rammer i stedet for benyttede 38 rammer. For resultatets del betyr dette at metadata for alle protokoll-lag blir overført én gang mer enn nødvendig. Feilen ble oppdaget på et tidspunkt da en ny test ikke var praktisk gjennomførbart.



Figur 4.4: Overføring av 2000 samples, 16bit LPCM.

5 Diskusjon

Kapittel 4 viser at Atmel SAM D20 har mer enn nok prosesseringskraft til å kjøre IMA ADPCM-enkoding i sanntid på signaler med punktprøvingsfrekvens $f_s = 8\text{kHz}$, ved en klokkefrekvens på $f_{CPU} = 8\text{MHz}$. Ved denne klokkefrekvensen skal den i teorien klare sanntids IMA ADPCM-enkoding av signaler med $f_s < 100\text{kHz}$, dersom den ikke har andre oppgaver. Ved å øke f_{CPU}, støttes enda høyere verdier av f_s.

I referansetesten bruker sensornoden $E_{ref} = 107\text{mJ}$ på å overføre 8000 16bit LPCM-samples via et enkelt Lightweight Mesh-nettverk. Den samme noden bruker $E_{sys} = 35.4\text{mJ}$ på å prosessere de samme 8000 16bit LPCM-samples ved hjelp av IMA ADPCM, og deretter overføre de resulterende 8000 4bit ADPCM-samples via det samme nettverket. Siden $E_{sys} < E_{ref}$, sparer man i dette tilfellet energi på å velge IMA ADPCM-komprimering foran overføring av rå LPCM-data. Resultatene tyder på at påstanden i [16], at kommunikasjonsdelen av en sensornode har et høyere energiforbruk enn prosesseringsdelen, stemmer for systemet beskrevet i denne oppgaven.

Som beskrevet i kapittel 3, ønsker sensornoden å overføre et LPCM-lydsignal med punktprøvingsfrekvens $f_s = 8\text{kHz}$ og bitdybde $B_{raw} = 16\text{bit/sample}$, som resulterer i en datarate $R_{raw} = B_{raw} f_s = 128\text{kbit/s}$. Referansetesten viser at sensornoden benyttet i denne oppgaven ikke greier å overføre et slikt signal i sanntid, uten kompresjon.

Ved IMA ADPCM-enkoding av signalet, får det resulterende signalet en bitdybde $B_{ADPCM} = 4\text{bit/sample}$. Resulterende datarate blir $R_{ADPCM} = B_{ADPCM} f_s = 32\text{kbit/s}$. Systemtesten viser at sensornoden benyttet i denne oppgaven klarer å håndtere både komprimering av kildesignal og overføring av resultatsignal i sanntid, ved de benyttede innstillinger.

Resultatene av referansetesten og systemtesten tyder på at kommunikasjonssystemet i den benyttede sensornoden er en flaskehals. Det blir imidlertid feil å sammenligne beregnet brutto datarate i testene med radioens oppgitte brutto datarate på 250kbit/s [39, side 1]. Beregningene i testene regner ut en *gjennomsnittlig* brutto datarate over hele applikasjonens kjøretid. Beregningene tar ikke høyde for at radioen har en oppgitt oppstartstid på omtrent 375ms fra dvalemodus [39, side 168].

Siden radioens oppstartstid er såpass stor i forhold til hele testenes kjøretid, er de beregnede overføringsratene trolig misvisende. Ved en reimplementasjon av sensor-

noden, burde radioen være påskrudd fra start, slik at oppstartstid ikke påvirker måleresultatet. I stedet for å la flanker i strømmålingen definere applikasjonens kjøretid, burde en GPIO-pinne på mikrokontrolleren benyttes til å indikere om applikasjonen kjører.

Det hadde også vært interessant å gjennomføre en grundigere analyse av Lightweight Mesh, både teoretisk og ved målinger, for å avdekke nøyaktig hva som er flaskehalsen i denne oppgaven. For å få gode tall på mulig hastighet i LWM-nettverket, burde en lengre test gjennomføres.

Et annet moment som ikke er tatt høyde for i denne oppgaven, er lydkvaliteten på det overførte signalet. IMA ADPCM er valgt av praktiske hensyn. I følge [49], som gir en sammenligning av en rekke kompresjonsalgoritmer for lyd, gir IMA ADPCM et signal/støy-forhold (SNR) på +35.81dB. Tilsvarende tall for 16bit LPCM er oppgitt å være +96.83dB. Mer komplekse algoritmer, som for eksempel MP3, oppnår høyere SNR-verdier og lavere datarater (høyere kompresjon) enn IMA ADPCM. Det ville vært interessant å undersøke nærmere i videre arbeide om sensornoden i denne oppgaven er i stand til å kjøre for eksempel MP3-enkoding i sanntid.

6 Konklusjon og videre arbeid

Denne oppgaven ønsker å besvare følgende spørsmål: hvordan påvirkes energiforbruket til en sensornode, dersom den benytter en datakompresjonsalgoritme for å redusere behovet for kommunikasjon? Resultatene i denne oppgaven viser tydelig at energikostnad ved kommunikasjon via Atmel Lightweight Mesh er høyere enn energikostnad ved IMA ADPCM-enkoding. En enkel kompresjonsalgoritme, som IMA ADPCM, gjør det mulig å redusere applikasjonens energiforbruk ved å redusere behovet for kommunikasjon. Ulempen med IMA ADPCM er at den ikke er tapsfri, og at den derfor ikke egner seg til alle applikasjoner.

I et videre arbeid ville det vært svært interessant å studere en mer kompleks komprimeringsalgoritme, som for eksempel MP3, for å finne ut om denne kan kjøre i sanntid på såpass beskjeden maskinvare som Atmel SAM D20. Det ville også vært interessant å se en analyse av energiforbruk for en slik algoritme, tilsvarende arbeidet i denne oppgaven, og sett om det finnes en grense for hvor mye prosessering som lønner seg med tanke på energiforbruk i en sanntidsapplikasjon.

Bibliografi

[1] Natasha Lomas, TechCrunch. *10BN+ Wirelessly Connected Devices Today, 30BN+ In 2020's Internet Of Everything, Says ABI Research.* URL: http://techcrunch.com/2013/05/09/internet-of-everything/.

[2] Samantha Murphy Kelly, Mashable. *The Internet of Things: Everything You Need to Know In 2 Minutes.* URL: http://mashable.com/2014/06/25/what-is-the-internet-of-things/.

[3] Mike Kavis, Forbes. *The Internet Of Things Will Radically Change Your Big Data Strategy.* URL: http://www.forbes.com/sites/mikekavis/2014/06/26/the-internet-of-things-will-radically-change-your-big-data-strategy/.

[4] The Guardian. *Wearable technology.* 2014. URL: http://www.theguardian.com/technology/wearable-technology.

[5] TechCrunch. *TechCrunch - Wearable.* 2014. URL: http://techcrunch.com/search/wearable.

[6] Google. *Sharing whats up our sleeve: Android coming to wearables.* URL: http://googleblog.blogspot.co.uk/2014/03/sharing-whats-up-our-sleeve-android.html.

[7] Kurt Lekanger, Amobil.no. *Samsung klar med sin første Android Wear-klokke.* URL: http://www.amobil.no/artikler/samsung-klar-med-sin-forste-android-wear-klokke/161243.

[8] Kurt Lekanger, Amobil.no. *Nå er LGs første Android Wear-klokke lansert.* URL: http://www.amobil.no/artikler/na-er-lgs-forste-android-wear-klokke-lansert/161246.

[9] Lior Ron, Motorola. *Moto 360: It's Time.* URL: http://motorola-blog.blogspot.no/2014/03/moto-360-its-time.html.

[10] Neil Hughes, Apple Insider. *Apple's first wearable device affirmed to be on track for October debut.* URL: http://appleinsider.com/articles/14/06/06/apples-first-wearable-device-affirmed-to-be-on-track-for-october-debut.

[11] MacRumors. *iWatch*. 6–2014. URL: http://www.macrumors.com/roundup/iwatch/.

[12] Ashleigh Allsopp, Macworld UK. *Apple iWatch release date, rumours and images*. URL: http://www.macworld.co.uk/news/apple/apple-iwatch-release-date-rumours-pictures-3425479/.

[13] Samsung. *Apps on Your Fridge*. 2014. URL: http://www.samsung.com/us/topic/apps-on-your-fridge.

[14] Nest. *Life with Nest Protect*. 2014. URL: https://nest.com/smoke-co-alarm/life-with-nest-protect/.

[15] Parrot. *Flower Power - intelligent wireless sensor for your plants*. 2014. URL: http://www.parrot.com/flowerpower.

[16] Giuseppe Anastasi mfl. "Energy Conservation in Wireless Sensor Networks: a Survey". I: *Ad Hoc Networks* 7 (5–2009), s. 537–568.

[17] Vijay Raghunathan mfl. "Energy-Aware Wireless Microsensor Networks". I: *IEEE Signal Processing Magazine* (3–2002), s. 40–50.

[18] Luca Benini og Giovanni de Micheli. *Dynamic Power Management: Design Techniques and CAD Tools*. Norwell, MA, USA: Kluwer Academic Publishers, 1998. ISBN: 079238086X.

[19] Rune Bjønnum. *Trådløs laveffekts datatransmisjon*. Prosjektoppgave. Norges teknisk-naturvitenskapelige universitet.

[20] Atmel Corporation. *AVR2130: Lightweight Mesh Developer Guide*. 3–2014. URL: http://www.atmel.com/images/atmel-42028-lightweight-mesh-developer-guide_application-note_avr2130.pdf.

[21] Atmel Corportation. *Communication Stacks*. 2014. URL: http://www.atmel.com/products/Wireless/802154/software.aspx.

[22] ISO. "Open Systems Interconnection Basic Reference Model: The Basic Model". I: *ISO/IEC 7498-1:1994* (1994).

[23] IEEE Computer Society. "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)". I: *IEEE Std 802.15.4-2006* (2006).

[24] IEEE Computer Society. "Wireless Medium Access Control (MAC) and Physical Layer (PHY) Specifications for Low-Rate Wireless Personal Area Networks (LR-WPANs)". I: *IEEE Std 802.15.4-2003* (2003).

[25] Jacob Ziv og Abraham Lempel. "A universal algorithm for sequential data compression". I: *IEEE TRANSACTIONS ON INFORMATION THEORY* 23.3 (1977), s. 337–343.

[26] Julian Seward. *bzip2*. 2014. URL: http://www.bzip.org.

[27] Gregory K Wallace. "The JPEG still picture compression standard". I: *Consumer Electronics, IEEE Transactions on* 38.1 (1992), s. xviii–xxxiv.

[28] ISO. "Generic coding of moving pictures and associated audio information Part 7: Advanced Audio Coding (AAC)". I: *ISO/IEC 13818-7:2006* (2006).

[29] The Interactive Multimedia Association. *Recommended Practices for Enhancing Digital Audio Compatibility in Multimedia Systems*.

[30] Davis Yen Pan. “Digital audio compression”. I: *Digital Technical Journal* 5.2 (1993), s. 28–40.

[31] Atmel Corporation. *SAM D20 Xplained Pro Evaluation Kit*. 2014. URL: http://www.atmel.com/tools/atsamd20-xpro.aspx.

[32] ARM Ltd. *Cortex-M0+ Processor*. 2014. URL: http://arm.com/products/processors/cortex-m/cortex-m0plus.php.

[33] Atmel Corporation. *ATSAMD20J18*. 2014. URL: http://www.atmel.com/devices/ATSAMD20J18.aspx.

[34] Atmel Corporation. *SAM D20 Datasheet*. 5–2014. URL: http://www.atmel.com/Images/Atmel-42129-SAM-D20_Datasheet.pdf.

[35] Exar Corporation. *SPX3819 500mA Low-Noise LDO Voltage Regulator*. 5–2014. URL: http://www.exar.com/common/content/document.ashx?id=615&languageid=1033.

[36] Atmel Corporation. *SAM D20 Xplained Pro Design Documentation*. 2014. URL: http://www.atmel.com/images/Atmel-42102-SAMD20-Xplained-Pro_User-Guide.zip.

[37] Atmel Corporation. *I/O1 Xplained Pro*. 2014. URL: http://www.atmel.com/tools/ATIO1-XPRO.aspx.

[38] Atmel Corporation. *OLED1 Xplained Pro*. 2014. URL: http://www.atmel.com/tools/ATOLED1-XPRO.aspx.

[39] Atmel Corporation. *AT86RF231 Datasheet*. 9–2009. URL: http://www.atmel.com/Images/doc8111.pdf.

[40] Atmel Corporation. *RZ600*. 2014. URL: http://www.atmel.com/tools/rz600.aspx.

[41] Atmel Corporation. *Atmel Lightweight Mesh*. 2014. URL: http://www.atmel.com/tools/lightweight_mesh.aspx.

[42] Atmel Corporation. *ASF-DSP*. 2014. URL: http://asf.atmel.com/docs/latest/asf_dsp.html.

[43] Microsoft Corporation. *Microsoft Windows XP - Fc*. 2014. URL: https://www.microsoft.com/resources/documentation/windows/xp/all/proddocs/en-us/fc.mspx?mfr=true.

[44] Tenma. *72-7735 Digital Multimeter Specifications*. URL: http://www.farnell.com/datasheets/1375656.pdf.

[45] Agilent Technologies. *InfiniiVision Oscilloscopes User's Guide*. 3–2011. URL: http://cp.literature.agilent.com/litweb/pdf/54695-97026.pdf.

[46] THAT Corporation. *THAT Corporation 1510/1512 Datasheet*. 2013. URL: http://www.thatcorp.com/datashts/THAT_1510-1512_Datasheet.pdf.

[47] ARM Ltd. *Cortex-M0+ Technical Reference Manual*. 2012. URL: http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ddi0484c/ch05s01s01.html.

[48] Atmel Corporation. *Atmel Software Framework*. 2014. URL: http://asf.atmel.com.

[49] E. Olson. *Audio Compression Codecs*. URL: http://www.baudline.com/solutions/codec/index.html.

A

Kildekode

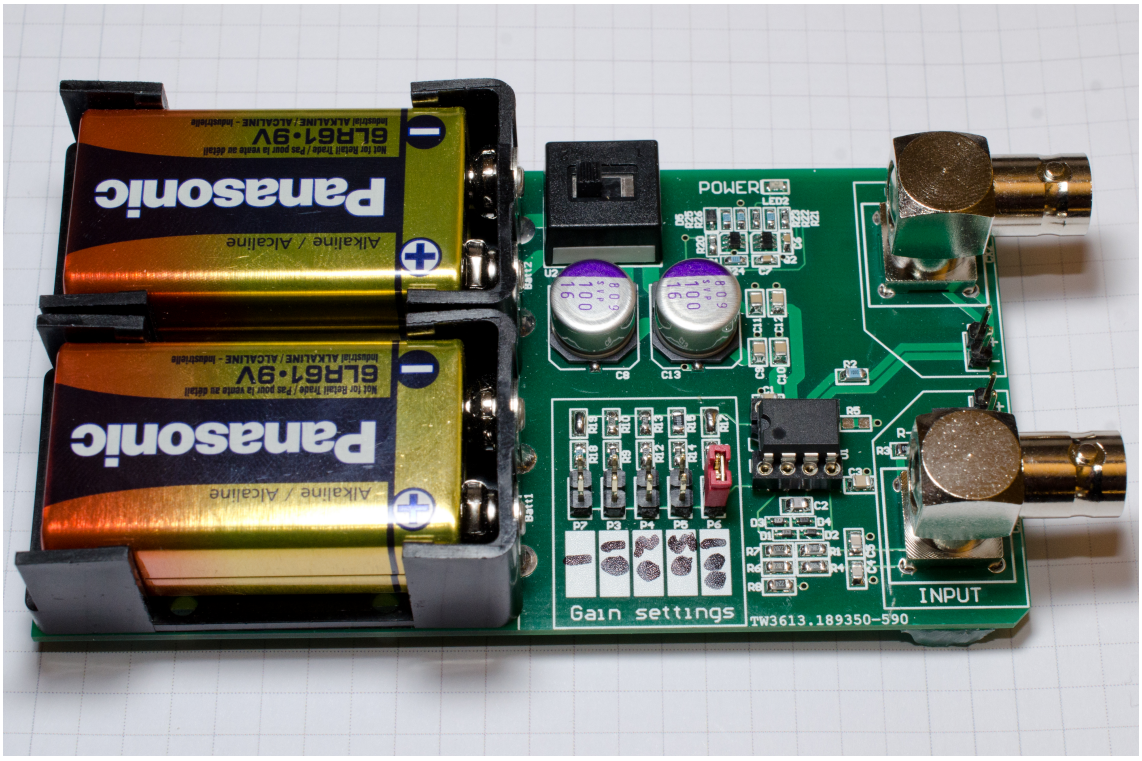
Denne rapporten følges av en ZIP-fil, som inneholder kildekoden til alle egenutviklede applikasjoner benyttet i forbindelse med oppgaven. En forenklet mappestruktur med beskrivelse for innholdet i ZIP-filen er vist under.

```
/
├── Lightweight Mesh - inneholder Atmel Lightweight Mesh-rammeverket
│   └── apps
│       ├── ADPCM-Transmitter - kildekode for sensornoden
│       └── SerialReceiver - kildekode for mottakernoden
├── SysTickTest - kildekode for kjøretidsestimering
├── Utils
│   ├── ADPCM-codec - kildekode for Windows-applikasjon for IMA ADPCM
│   ├── bin
│   │   ├── adpcm.exe - kjørbar Windows-applikasjon for IMA ADPCM
│   │   ├── SerialDump.exe - kjørbar Windows-applikasjon for lytting på
│   │   │   serieport
│   │   └── WriteRaw.exe - kjørbar Windows-applikasjon for generering av
│   │       referansedata
│   ├── Matlab
│   │   ├── playSound.m - MATLAB-funksjon for å spille av referansedata
│   │   └── Tonegenerator.m - MATLAB-script for å generere testsignal
│   ├── SerialDump - kildekode for applikasjon for lytting på serieport
│   └── WriteRaw - kildekode for generering av referansedata
└── 2014-07-07 - Masteroppgave.pdf - denne rapporten i PDF-format
```

B

Koblingsskjema strømprobe

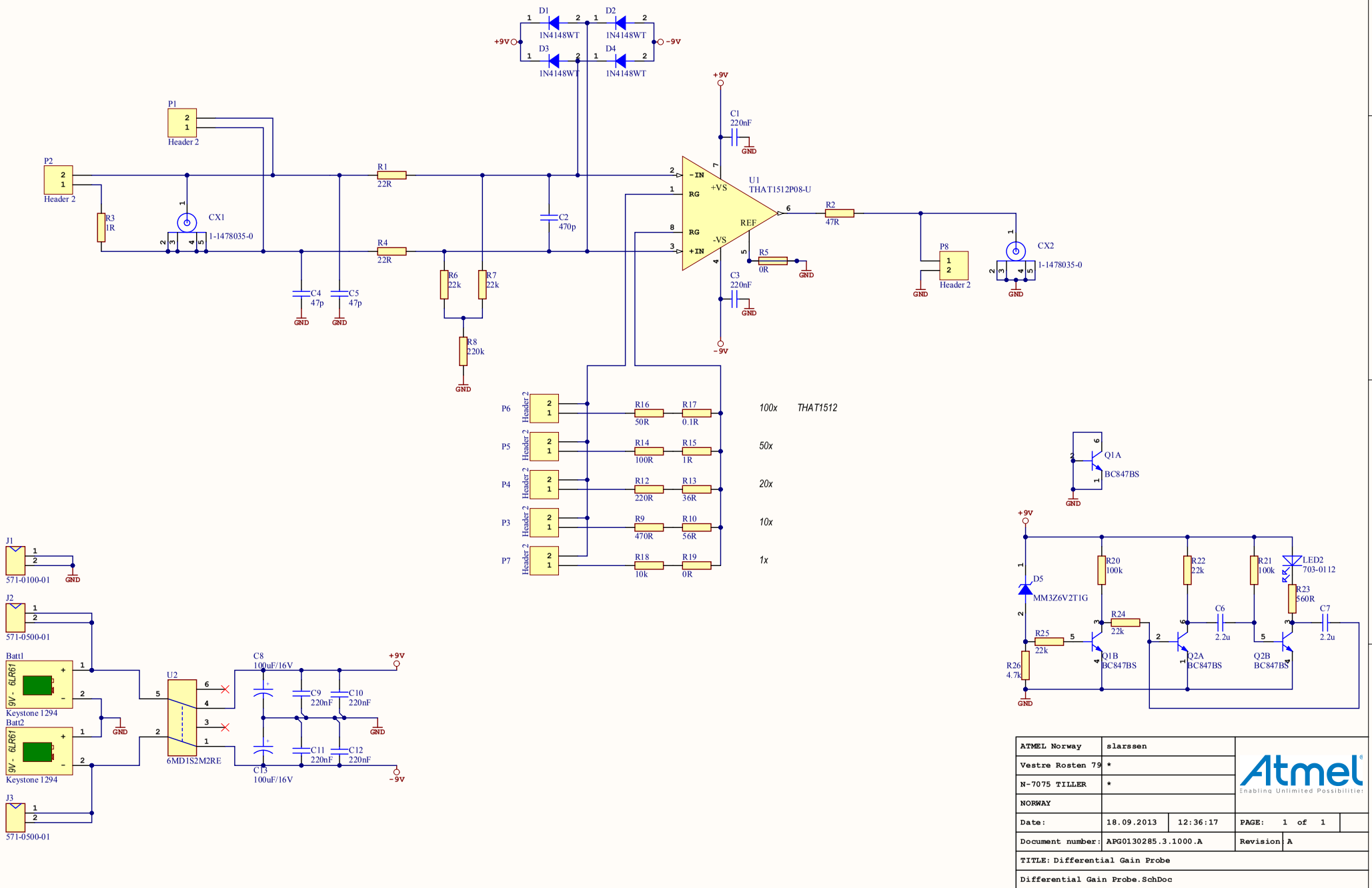
De neste to sidene viser koblingsskjema og deleliste for strømprobe, designet av Stig Larssen, ATMEL Norway. Gjengitt med tillatelse fra designeren.



Gain settings: 1x, 10x, 20x, 50x, 100x.

Gain = 0.5+(5000/RG) for THAT1512

Gain = 1+(10000/RG) for INA217



ATMEL Norway	slarssen		
Vestre Rosten 79	*		
N-7075 TILLER	*		
NORWAY			
Date:	18.09.2013	12:36:17	PAGE: 1 of 1
Document number:	APG0130285.3.1000.A		Revision A
TITLE: Differential Gain Probe			
Differential Gain Probe.SchDoc			

Component list

Bill of Materials For Project [Differential Gain Probe.PrjPCB] (No PCB Document Selected)

Source Data From: Differential Gain Probe.PrjPCB
Project: Differential Gain Probe.PrjPCB
Variant: None

Report Date: 18.09.2013 12:36:50
Print Date: 18.09.2013 12:36:26

#	Designator	Quantity	Column Name Error:Valu	Manufacturer	MPN	Description
1	Batt1, Batt2	2		Keystone Corp.	Keystone 1294	Battery holder, 9V, PCB, THM, Keystone-1294
2	C1, C2, C3, C4, C5, C9, C10, C11, C12	9		Murata	GRM21BR71E104KA01L, [NoParam], GRM21BR71E104KA01L, [NoParam], [NoParam], GRM21BR61C106KE15, GRM21BR71E104KA01L, GRM21BR61C106KE15, GRM21BR71E104KA01L	Ceramic capacitor, SMD 0805, X7R, 50V, ±10 %, Ceramic capacitor, SMD 0805, NP0, 50V, ±5 %, Ceramic capacitor, SMD 0805, X7R, 50V, ±10 %, Ceramic capacitor, SMD 0805, NP0, 50V, ±5 %, Ceramic capacitor, SMD 0805, NP0, 50V, ±5 %, Ceramic capacitor, SMD 0805, X5R, 16V, 10 %,, Ceramic capacitor, SMD 0805, X7R, 50V, ±10 %, Ceramic capacitor, SMD 0805, X5R, 16V, 10 %,, Ceramic capacitor, SMD 0805, X7R, 50V, ±10 %
3	C6, C7	2		Kemet	C0603C225K8PAC7867	Ceramic capacitor, SMD 0603, X5R, 10V, ±10%
4	C8, C13	2		Sanyo	50CV220AX	SMD electrolytic capasitor Ø10mm, H10.2mm, 105degC, 50V, Max rippel=0.450(@105degC, 100kHz), Tan=0.14. Sanyo's CV-AX series
5	CX1, CX2	2		TYCO	1-1478035-0	BNC Coaxial jack right-angle, 50ohm
6	D1, D2, D3, D4	4		Diodes Incorporated	1N4148WT-7-F	Diode, Switching, 125mA, 100V, SOD-523
7	D5	1		ON Semiconductor	MM3Z6V2T1G	Zener Diode, SMD SOD-323, 6.2V, 0.2W, 5%
8	J1	1		Deltron Emcon	571-0100-01	4mm PCB power connector, 10A, BLACK
9	J2, J3	2		Deltron Emcon	571-0500-01	4mm PCB power connector, 10A, RED
10	LED2	1		Multicomp	703-0112	LED, 0603, Red, 1.8V, 20mA, 15mcd, 140°
11	P1, P2, P3, P4, P5, P6, P7, P8	8				Header, 2-Pin
12	Q1, Q2	2		Philips	BC847BS	General purpose SMD small signal BJT Dual transistor, 2xNPN. Body Sot363
13	R1, R2, R3, R4, R5, R6, R7, R8, R9, R10, R12, R13, R14, R15, R16, R17, R18, R19	18				Thick film resistor, SMD 0805, 0.125W, 1%
14	R20, R21, R22, R23, R24, R25, R26	7		Panasonic, Panasonic, Panasonic, [NoParam], [NoParam], KOA, KOA	ERJ-3EKF1003V, ERJ-3EKF1003V, ERJ-3EKF1003V, [NoParam], [NoParam], RK73H1JTTD1002F, RK73H1JTTD1001F	Thick film resistor, SMD 0603, 1/10W, 1%
15	U1	1		THAT Corporation	THAT1512P08-U	Low -Noise, High bandwith Audio Preamplifier Differential Opamp, DIP8
16	U2	1		Multicomp	6MD1S2M2RE	Slide switch, DPDT, On-On, THM, 20V, 100mA
		63				

Approved

Notes