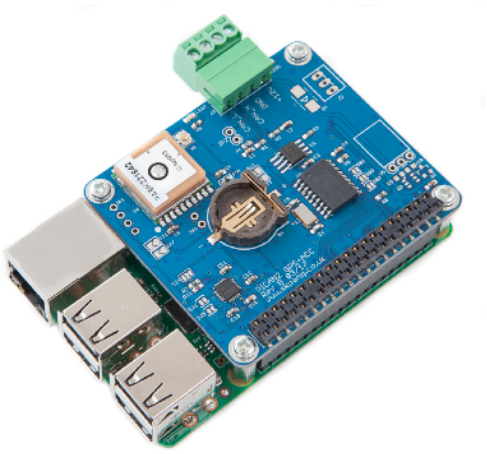

Prosjekt del 3

Raspberry Pi 3 / Buildroot og CAN-bus.

Om prosjekt del 3

Nødvendig SW for siste del av prosjektet er Ubuntu Linux (f.eks. installert på en virtuell maskin) og QT Creator (kjørende på Ubuntu Linux). Nødvendig HW finnes i lab-koffert.

Det er beregnet to lab-dager for del 3. Husk dokumentasjon/ rapportskriving underveis!

Dag 1 og 2

Bruk gjerne et "mirror" i Norge ved nedlasting av ISO-image:

http://ftp.uio.no/pub/linux/ubuntu-iso/

Oppgave 0 - Installere Ubuntu Linux på Virtualbox

- Ubuntu-installasjonen fra første øving kan gjenbrukes, fortrinnsvis velg siste LTS-versjon av Ubuntu (20.04.1 LTS). Pass på å gi den virtuelle maskinen minimum to prosessorer, gjerne fire (men la to kjerner være igjen til OS på host-maskin). Det bør også velges 4 GB RAM eller mer, gjerne 8 GB. Velg gjerne minimal installasjon av Ubuntu for å spare plass.
- Maskinen som kjører Ubuntu-linux kalles "host" under lab-en. Merk at host-maskinen bør ha tilgang til en ethenet-port (hvis Virtualbox kjøres på en maskin med ethernet-tilgang, så kan vi benytte denne porten).
- Raspberry pi kalles "target".

Oppgave 1 - Laste ned og konfigurere buildroot

Buildroot et et verktøy som benyttes for å konfigurere og bygge Linux-distribusjoner for innebygde datasystemer.

På lab-en skal vi benytte buildroot til å konfigurere en egen Linux-distribusjon for MAS234. Linux-distribusjonen skal fungere med CAN-bus på kombinasjonen Raspberry Pi 3 og Skpang CAN-kort.

Vi skal også legge inn en ekstra programvarepakke i buildroot-menyen, slik at vi får med vår egen programvare når distribusjonen genereres.

Buildroot har en stor samling programvarepakker og mengder av konfigurasjonsmuligheter.

Vi behøver heldigvis ikke benytte alt; og det finnes også ferdige oppsett for en god del maskinvare (slik som Raspberry Pi 3).

1. Last ned buildroot 2020.02.7 LTS på Ubuntu Linux. Pakk ut og legg buildroot-katalogen inni en mappe som heter "buildroot-projects". Altså filbane totalt f.eks `/home/brukernavn/buildroot-project/buildroot-2020.02.7`

Buildroot-prosjektets hjemmeside:

https://buildroot.org

Buildroot kan lastes ned fra prosjektets hjemmeside her: https://buildroot.org/

2. Åpne et terminalvindu og bruk cd (change directory) til å navigere frem til katalogen som ble pakket ut i forrige punkt. Kjør:

   ```
   make menuconfig
   ```

   Merk forøvrig at det **ikke** skal være nødvendig å kjøre noen av de buildroot-relaterte kommandoene som superbruker (unngå bruk av sudo).

   Tips: Sannsynligvis mangler biblioteket ncurses på maskinen. Dersom buildroot klager på dette, så må ncurses-devel installeres på Ubuntu før dere fortsetter. Pakken heter "libncurses-dev" i Ubuntu-pakkebrønnen.

3. Når `make menuconfig` i forrige steg fungerer; lukk konfiguratoren og kjør `make list-defconfigs`. Denne kommandoen lister opp eksisterende konfigurasjoner ("templates") for diverse maskinvare.

   Se over listen over hvilke ulike plattformer som støttes, spesielt Raspberry Pi-variantene.

4. Kjør "make raspberrypi3_defconfig" og bla litt rundt i menyene. Hvilken prosessor-arkitektur er valgt?

5. Nå KUNNE vi bygd en standard-konfigurasjon for Raspberry Pi 3. Imidlertid krever denne skjerm eller TTL-uart-kabel for terminal. Dette har vi foreløpig ikke, og det er helle ikke alltid skjerm tilgjengelig på et realistisk innebygget datasystem.

Eksempler på absolutte filstier:

```
/home/student/proj/
~/folder/fil
/etc/dhcp/dhcpd.conf
```

Eksempler på relative filstier:

```
mappe
fil
mappe/fil
brukernavn/mappe
.
../mappe/fil
```

Oppgave 2 - MAS234-konfigurasjon av buildroot

1. Dere får en kopi av et ferdig oppsett. (Dvs. i det minste halvferdig...). Se Canvas.

   Dette buildroot-prosjektet er konfigurert slik at CAN-bus-relevante pakker er forhåndsvalgt, i tillegg til at det er satt opp en SSH-server. Dermed kan det logges inn på maskinen over nettverk med SSH -- uten behov for skjerm.

   Prosjektet inneholder en egen mas234-pakke med et qtmake-basert C++-program. Dette kan åpnes i Qt Creator for videreutvikling. Oppgaven i siste trinn blir å lage et program som kommuniserer på CAN-bus basert på dette halvferdige oppsettet.

2. Hent buildroot-project.tar.gz på Canvas og pakk ut. Les README-filen. En kopi av buildroot må pakkes ut eller klones inn i buildroot-project-mappen.

3. Resulterende prosjekt-struktur skal være noe tilsvarende det som er vist i Vedlegg 1.

4. Stå i buildroot-project-mappen (hakket utenfor buildroot-mappen). Kjør:

   ```
   make O=${PWD}/buildroot-output/
   BR2_EXTERNAL=${PWD}/external -C ${PWD}/
   buildroot menuconfig
   ```

   Og sjekk oppsettet. Finnes det en can_pingpong-pakke her? Kan denne velges? Forsøk å legge inn nødvendige pakker for å oppfylle avhengighetene til can_pingpong.

   Opsjonen `-C` forteller hvor buildroot befinner seg (bruk absolutt filsti).

5. `O=/filsti/til/ouput-folder` velger hvor output skal plasseres når vi genererer konfigurasjon og bygger. Hva skjer hvis vi ikke legger til dette? Og hva betyr `O=${PWD}` ?

6. Hva er `BR2_EXTERNAL`, og hva gjør denne?

7. mas234-konfigurasjonsfilen kopieres inn i buildroot-output-folderen som .config. Merk at filer med navn som begynner på "." er skjulte filer. For å vise disse, bruk f.eks. `ls -la` fra terminalen. Dette overskriver eventuelle valg som ble gjort i forrige steg. Kjør kommandoen i punkt 7 på nytt.
8. Sjekk at can_pingpong er valgt. Lukk menuconfig. Start kompilering:

   ```
   make -C /filsti/til/buildroot-prosjekt/buildroot-output/
   ```

   eller:

   ```
   cd buildroot-output
   make
   ```

9. Vent, men følg med i tilfelle byggeprosessen stanser... Løs ut eventuelle feilmeldinger.
10. Når buildroot har bygd ferdig (tar gjerne 10-90 minutter avhengig av maskinvare), så skal det ligge et ferdig generert sd-kort-image i folderen buildroot-output/images/.

    sdcard.img skal nå skrives til et microSD-kort ved å benytte en microSD-kort-leser og linux-programmet dd. Merk at dd lett skriver over harddisken på maskinen hvis det brukes feil.

    Bruk fdisk -l for å liste diskene som er montert på maskinen FØR sd-kortet er satt i kortleseren. Merk deg / dev/sda, /dev/sdb etc..

    Plugg i minnekortleseren og minnekortet. Kjør fdisk -l på nytt. Disken som dukker opp nå er målet for kopieringen. Sjekk gjerne også at størrelsen fdisk indikerer er som forventet med tanke på hva slags kort som er satt inn.

    Bruk dd til å kopiere, f.eks. når måldisken heter /dev/sde:

    ```
    dd if=buildroot-output/images/sdcard.img of=/dev/sde bs=4M
    ```

    Dette bør gå relativt raskt. Det anbefales å kjøre umount på måldisken før kopiering med dd.

(F.eks. umount /media/<brukernavn>/* for å unmounte alle automonterte disker).

11. *Løs ut disken* og sett den inn i Raspberry Pi. Forsøk å boote.
12. <koble til med SSH, se oppgave 4>
13. For å bygge KUN can_pingpong-pakken og ikke hele buildroot-oppsettet, kjør:

```
make -C /filsti/til/buildroot-prosjekt/
buildroot-output/ can_pingpong-rebuild
```

14. For å kopiere krysskompilert program til raspberry pi, kjør:

```
scp /filsti/til/buildroot-prosjekt/
buildroot-output/build/can_pingpong-3.14/
can_pingpong root@192.168.234.234:/usr/
bin/can_pingpong
```

Raspberry Pi konfigureres med en *statisk* IPv4-adresse 192.168.234.234.

For å nå denne IP-adressen fra din datamaskin, på en enkel måte, så settes IP på denne også statisk til f.eks. `192.168.234.1`. Det viktige her er at maskinene er på samme subnett, eller at IP-ene kan rutes mellom nettverkene. Og at det passes på at to maskiner ikke tildeles samme IP på samme nettverk.

Bruk kommandoen `ifconfig` fra terminalen for å sjekke aktiv nettverkskonfigurasjon!

Se filen `/etc/network/interfaces` og gjør deg kjent med nettverks-konfigurasjonen. Både på host-maskinen (der vi kryss-kompilerer) og på target-maskinen (raspberry pi-maskinen). Se manual for `/etc/network/interfaces` her.

Vær skeptisk til statiske IP-adresser i reelle nettverk. Det er tungt å administrere sammenlignet med automatisk tildeling av IP-er med DHCP. Det er mulig å tildele fast IP til en bestemt maskin med DHCP!

Litt mer om IP-adresser ++

Adressen 192.168.234.234 begynner med 192.168, og er derfor en *privat adresse*. Private IP-adresser er ikke rutbare over internett, og benyttes ofte på lokale nettverk som f.eks. er koblet til internett via NAT (Network Address Translation). IANA har reservert følgende IP-adresseområder for denne typen bruk: https://en.wikipedia.org/wiki/Private_network .

Dersom vi ønsker å nå maskinen fra internett, kan den tildeles en offentlig IP-adresse. Eller en router kan settes opp til å videresende trafikk fra en offentlig IP til vår private IP (f.eks. ved hjelp av NAT).

Straks vi har satt opp maskinen med offentlig IP, eller konfigurert NAT-forwarding, kan den nås fra HELE internett. **God sikkerhet med tanke på sikkerhetsoppdateringer, passord etc. er dermed essensielt.**

Selv når vi IKKE har satt opp offentlig IP eller NAT-forwarding kan maskinen nås fra internett med visse teknikker. Anbefalingen er derfor å sikre den godt i de tilfellene *nettverket* er tilkoblet internett -- i praksis må den dermed sikres godt i de aller fleste tilfeller. Passord av denne typen bør ikke anvendes: https://en.wikipedia.org/wiki/List_of_the_most_common_passwords

Oppgave 3 - Koble opp hardware

Hver gruppe finner en Raspberry Pi 3 og et "pi hat"-kort med IMU, GPS og CAN-bus-funksjonalitet i kofferten. I tillegg har vi et microSD-kort pr. gruppe, som fungerer som "harddisk".

Dokumentasjon Raspberry Pi 3: https://www.raspberrypi.org/documentation/hardware/raspberrypi/README.md

Dokumentasjon Skpang Can-kort: http://skpang.co.uk/catalog/pican-with-gps-gyro-accelerometer-canbus-for-raspberry-pi-3-p-1521.html

- Merk: Pi-hat-kortet SKAL kobles opp m/standoffs slik at GPIO-headeren ikke brekker under bruk. Gi beskjed dersom det ikke er standoffs på deres kort.
- Husk å sjekke impedans på CAN-bus-en. Finnes det termineringsmotstand på SKpang-kortet?
- Det er behov for strømforsyning (micro-usb) og nettverkskabel for neste del.

Oppgave 4 - Sende en CAN-melding fra terminal

1. Sett opp et nettverk med statisk IP på porten hvor Raspberry Pi er koblet til. Velg ip 192.168.234.1 på host-maskinen og nettmaske 255.255.255.0.

2. Raspberry Pi med MAS234-konfigurasjon har IP 192.168.234.234. Koble til denne med SSH fra terminalen i Ubuntu:
   `ssh root@192.168.234.234`
   MERK: Det er IKKE anbefalt å logge inn som root over SSH. Spesielt ikke med et kort og dårlig passord som vi har her. Men så lenge vi kun kommuniserer med RPI på et eget, lokalt nettverk -- uten internett-tilkobling, så er det OK.

   Passord: mas234

3. Kjør `ifconfig` for å sjekke hvilke nettverks-grensesnitt som er aktive. Hvis ikke can0 er synlig; kjør:

   ```
   modprobe spi_bcm2835
   modprobe mcp251x
   ```

   Dette laster kjernemodulene (drivere) for SPI og CAN-tranceiveren på SKPANG-kortet (MCP2515).

   Deretter kjøres følgende for å konfigurere can-grensesnittet:

   ```
   /sbin/ip link set can0 up type can bitrate 250000
   ```

4. Nå skal det være mulig å sende og motta CAN-meldinger på can0-grensesnittet. Sjekk gjerne at can0 eksisterer vha. `ifconfig`.

5. Koble Peak-adapteret på CAN-bus-en. Start candump med:

   ```
   candump can0
   ```

   Candump printer alle can-meldinger som mottas til terminalen. Send noen meldinger fra PCAN-programmet, og verifiser at disse mottas.

   Avslutt candump ved å trykke `ctrl+c`

CAN-konfigurasjonen for RPI-maskinen er ikke komplett i oppsettet fra Canvas, og ip link-kommandoen for å sette opp can0 vil feile.

Dere må enten sørge for å legge til følgende linjer nederst i filen /boot/config.txt , eller inkludere tilsvarende ved hjelp av f.eks. overlay i buildroot menuconfig:

```
# CAN controller
dtoverlay=mcp2515-
can0,oscillator=16000000,interrupt=25

dtoverlay=spi-bcm2835
```

6. Send et par CAN-meldinger fra Raspberry Pi:

   ```
   cansend can0 234#23.40.00.00.FF.FF.FF.00
   ```

   Sjekk at disse kommer frem.

Oppgave 5 - Fritt spill

1. Koble opp minst to enheter på CAN-nettverket, f.eks. Teensy-en og Raspberry Pi-en.
2. To krav: Minst én node skal lese sensordata og sende disse til en annen node. Noden som mottar sensordata skal gjøre noe med disse, og sende tilbake prosesserte data til sensornoden eller en annen node.
3. Det er mulig å låne f.eks. en RC-servo og styre denne fra Teensy. Benytt i så fall eget (lab)-powersupply på servoen.
4. To eller flere grupper kan koble seg sammen på ett og samme CAN-nettverk.

Oppgave 6 - Innlevering av utstyr

Gruppene kan benytte lånt utstyr frem til og med siste lab-dag. Alt utstyr må være innlevert før rapportens innleveringsfrist.

Vedlegg A

Prosjektstruktur for buildroot-prosjektet:

```
.
├── buildroot
├── buildroot-output
├── external
│   ├── Config.in
│   ├── configs
│   │   └── mas234_rpi3_defconfig
│   ├── external.desc
│   ├── external.mk
│   └── package
│       └── mas234
│           ├── can_pingpong
│           │   ├── can_pingpong.mk
│           │   ├── Config.in
│           │   └── README.md
│           ├── Config.in
│           └── mas234.mk
├── mas234
│   └── can_pingpong
│       ├── can_pingpong.pro
│       └── main.cpp
├── mas234-rootfs-overlay
│   └── etc
│       └── network
│           └── interfaces
```

- Nedlastet buildroot-versjon 2020.02.7 (eller annen versjon dersom dere benytter dette) skal befinne seg i mappen buildroot.
- Katalogen buildroot-output skal være tom før defconfig lastes inn.
  - Build-konfigurasjonen lagres i buildroot-output, og påvirker dermed ikke buildroot. Alle endringer som lagres fra menuconfig plasseres i .config-filen (skjult fil i output-katalogen).
  - Makefile genereres og plasseres i output-katalogen.
  - Ved bygging (etter at Makefile er generert), så plasseres genererte filer også i output-katalogen. Det betyr at vi ikke blander resultater (genererte filer) sammen med kildefiler.
- external-katalogen inneholder våre tilpasninger av buildroot. Configs inneholder en tilpasset konfigurasjon av raspberrypi3_defconfig, package inkluderer et C++-prosjekt, og peker til mas234-katalogen.
- mas234-katalogen inneholder et Qt Creator-prosjekt, og vi kan dermed programmere C++ i et kjent miljø direkte for operativsystemet vi setter sammen.
- mas234-rootfs-overlay inneholder nettverkskonfigurasjon (setter statisk IP for imaget vi genererer). Noe som gjør at vi vet hvilken IP raspberry-en starter opp med når vi booter vårt image.