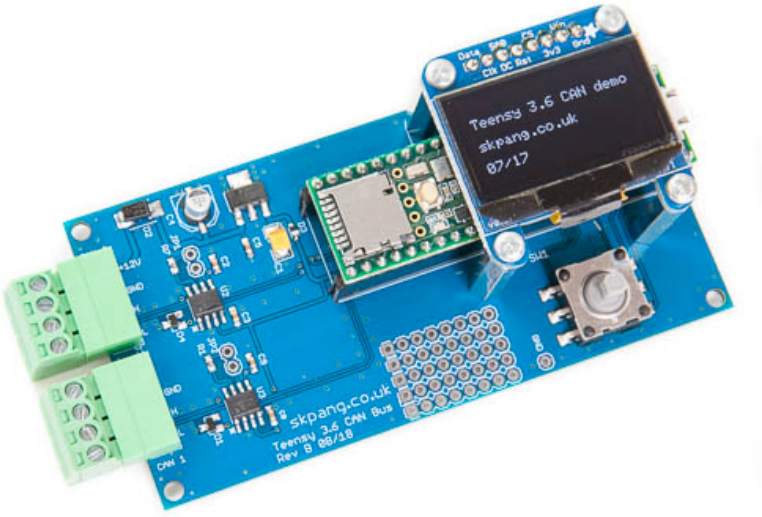


Prosjekt del 2

Arduino / Teensy og CAN-bus.

Om prosjekt del 2

Nødvendig SW for første del av prosjektet er Arduino IDE med Teensy-utvidelser. Autodesk Eagle (eller lignende) kan med fordel benyttes til dokumentasjon.

Dag 1

Oppgave 0 - Installere CAN-bibliotek for Arduino

1. Arduino IDE må installeres hvis du ikke har dette fra før. https://www.arduino.cc/en/software
2. Sjekk at toolchain og IDE fungerer ved å kjøre et eksempel eller to på Teensy-en.
3. https://github.com/collin80/FlexCAN_Library lastes ned og installeres med bibliotek-funksjonen i Arduino IDE.
4. Sjekk at installasjonen fungerer ved å kompilere minst ett av eksemplene for dual CAN-bus på Teensy 3.6.

Oppgave 1 - Koble opp CAN-kort på Teensy

Lab-kofferten inneholder et Teensy 3.6 carrier-board med to CAN-bus-konnektorer + OLED-skjerm.

5. Hvilke pinner på Teensy 3.6 er koblet mot hver av de to CAN-tranceiverene på carrier-kortet? (Teensy 3.6 har både primær og sekunder CAN-utgang for begge CAN-kanalene.) Lag gjerne et kretsskjema f.eks. med Eagle.
6. For tilkobling av CAN-bus mot PCAN-adapter med USB, er det nødvendig å lage en D-sub-adapter. Dere får utdelt en D-sub-konnektor + et breakout-kort. Lodd dette korrekt, slik at CANH og CANL havner på rett sted på breakout-kortet (sjekk pinnenummerering både på PEAK PCAN og ut fra kretskortet).
7. Kontrollmål gjerne at det ikke er kortslutning mellom CANH og CANL og VCC og GND.
8. Må det anvendes termineringsmotstand, og i så fall hvor stor skal denne være? Har carrier-kortet innebygget termineringsmotstand?

Oppgave 2 - Sende en CAN-melding

Bruk et av eksempelprogrammene fra SK Pang, og sjekk at CAN-meldinger kan sendes og mottas mellom de to kanalene på CAN-kortet eller med Peak PCAN-USB CAN-grensesnitt for PC.

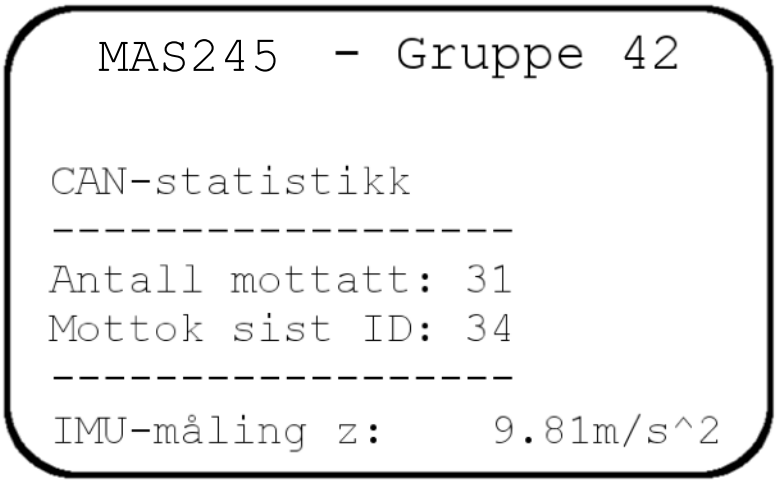
- Send en melding, ta i mot denne, og returnér samme melding til avsenderen.
- Ved bruk av PCAN USB-adapter, se https://www.peak-system.com/PCAN-USB-FD.365.0.html?&L=1 for programvare.
- Lag en beskrivelse av de nødvendige stegene for å sende en melding med CAN-biblioteket (flexcan) på Arduino.
- Hvor store CAN-meldinger kan maksimalt sendes med flexcan-biblioteket?
- Eventuelle CAN-feilkoder fra PCAN-USB når denne er tilkoblet bussen, kan dekodes ved hjelp av header-filen til dette biblioteket: "PCAN-Basic API (Linux)".

Link til eksempel-sketch fra SKpang på github: https://github.com/skpang/Teensy-3.6-Dual-CAN-Bus-Breakout-Board . Merk at selv om dere benytter dette for å teste/komme i gang, så skal selve løsningen på oppgaven lages i et eget sketch hvor dere kun har med det som er nødvendig for å løse oppgaven (altså: unngå klipp og lim!).

Oppgave 3 - MAS245 CAN message receiver

Carrier-kortet fra SK Pang er utstyrt med OLED-display. I denne oppgaven skal dere tegne opp en ramme (f.eks. et avrundet rektangel) på skjermen, med tittel f.eks. MAS245 og/eller Gruppe X øverst, altså tilsvarende figuren.

Informasjonen om antall meldinger sendt og mottatt skal reflektere det som skjer på CAN-bus.



Adafruit graphics-library tilbyr funksjonalitet for å tegne linjer, rektangler etc. på skjermen på en enkel måte: https://learn.adafruit.com/adafruit-gfx-graphics-library

Velg oppgave 4a eller 4b

Oppgave 4a - Rapportere IMU-verdier på CAN-bus hvert sekund og vise dem frem på skjerm

Målet med oppgaven er å lese inn IMU-verdier, for deretter å sende dem ut på CAN-bus og i tillegg presentere dem på skjermen.

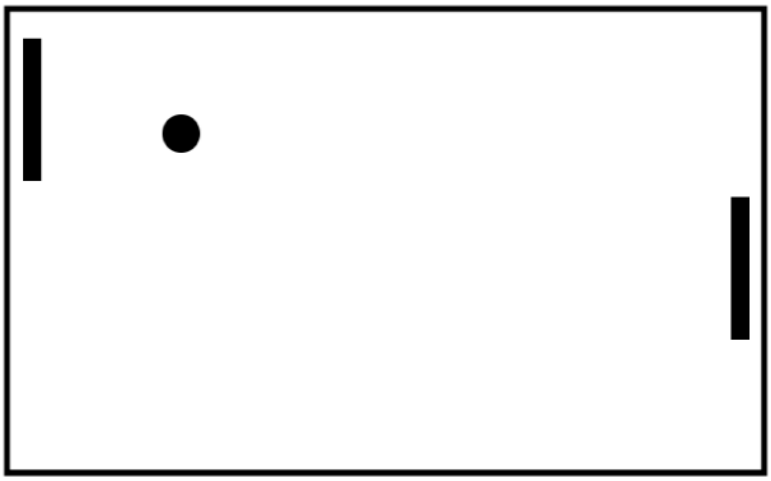
IMU med breakout-kort utleveres ved behov. Dersom det ikke er loddet på pin-header, så må dette gjøres.

1. Tegn kretsskjema og koble opp IMU mot Teensy 3.6.
2. Les ut IMU-målinger vha. passende bibliotek.
3. Sørg for å vise frem IMU-måling på OLED-skjermen. Minimum akselerasjon i Z-retning. Merk at målt akselerasjon skal vises i SI-enhet m/s^2.
4. IMU-målingene også distribueres på CAN-bus:
    I. Meldings-ID 0x21 benyttes for å starte utsending av IMU-verdier. Denne sendes f.eks. av PC-en (via PCAN-adapter).
    II. Meldings-ID 0x22 benyttes for å sende de faktiske IMU-verdiene. Verdiene skal ligge i payload felt 0-5 eller 0-11 alt etter om det benyttes 8 eller 16 bit signed integer per verdi.
5. Meldings-ID 0x22 skal sendes på fast rate 1 Hz.

    Tips: https://www.pjrc.com/teensy/td_libs_Metro.html

Oppgave 4b - CAN-ping-pong

Oppgaven går ut på å lage et enkelt spill for to spillere. Hver spiller styrer en plate med lengde 20 pixler på sin side av spillebrettet. Egen spiller/plate er platen på høyre side.



1. Egen plate skal kunne styres opp og ned med joystick-en. Posisjonen på egen plate skal rapporteres over CAN-bus med meldings ID etter formelen ID = Guppenr + 20.
2. Motspillers plate skal tegnes på siste rapporterte posisjon fra motspiller.
3. Første spiller som trykker på joystick-en etter oppstart, vil være master i spillet, og er ansvarlig for å sende ut en ball i form av CAN-melding med ID = Gruppenr + 50. Oppdatert ball-posisjon skal sendes ut hvert 10. millisekund (altså med rate 100 Hz).
4. Eventuelle ball-meldinger som mottas, skal medføre at en ball tegnes opp på oppgitt posisjon. Husk at motstander har motsatt koordinatsystem, og den må derfor tegnes opp speilvendt!

Struktur på CAN-meldinger for posisjon (datatyper etc.) må avtales med motstander-gruppe. Eventuelt så kan dere samarbeide slik at dere låner en ekstra Teensy fra nabo-gruppen for å demonstrere eget spill, og nabo-gruppen låner deres Teensy når de skal demonstrere/teste sitt spill.

Det forventes at det som et minimum kan demonstreres fungerende visning av posisjonen på egen plate og motstanders plate. Poengberegning, enkel simulering av ballens fysikk etc. er ønskelig.