Semesterprosjekt

Rapport

PC-basert instrumentering & kommunikasjonsnett

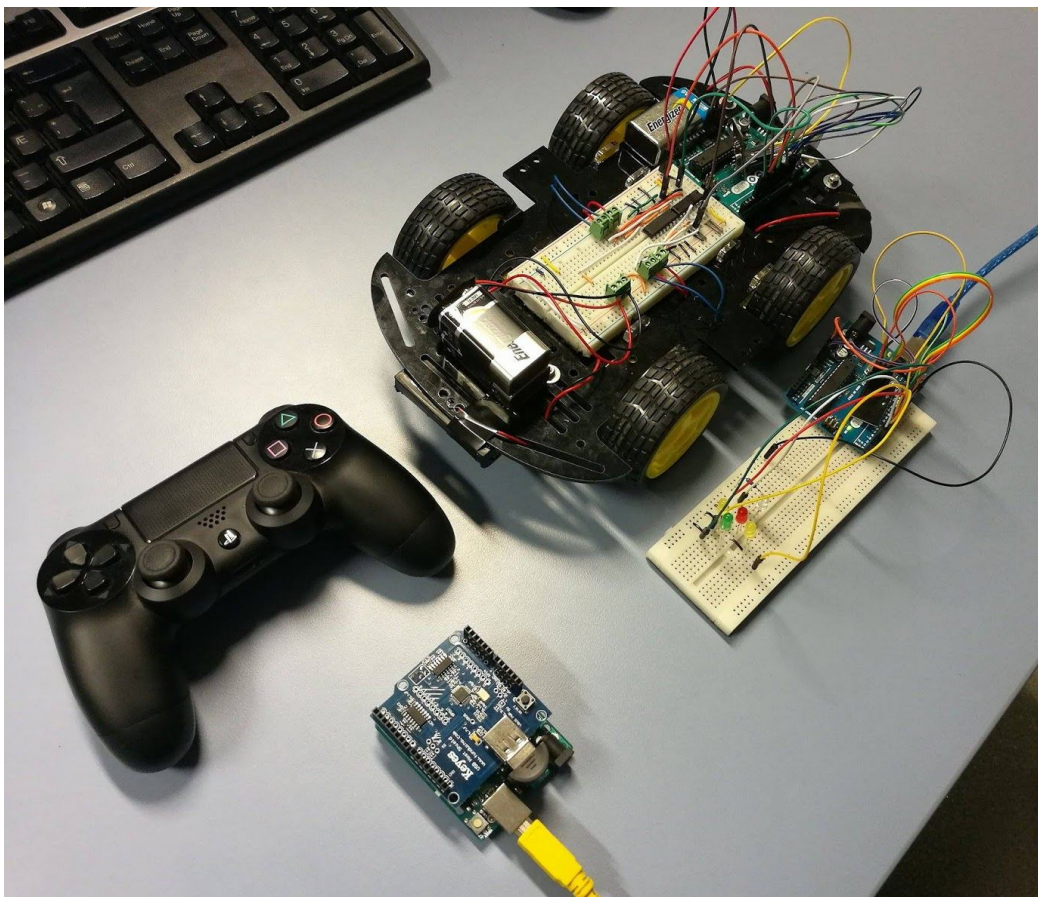
ELTS2200, Høst 2016

Christophe Plaissy, s300344
Mark Strømme s300357,
Kristina Waaler s305155

Forord

Prosjektet går ut på å kunne styre en bil som er koblet til en Server via Klient. Det ble valgt denne problemstillingen for å bli kjent med mye forskjellig utstyr og systemer, samt at det er veldig lærerikt.

Prosjekt og rapport er utført av Mark Strømme, Eivind Tandberg, Kristina Waaler og Christophe Plaissy. Studenter ved 2. klasse ingeniørfag, elektronikk og informasjonsteknologi ved HiOA (HINGELEKTR15H).



Innholdsfortegnelse

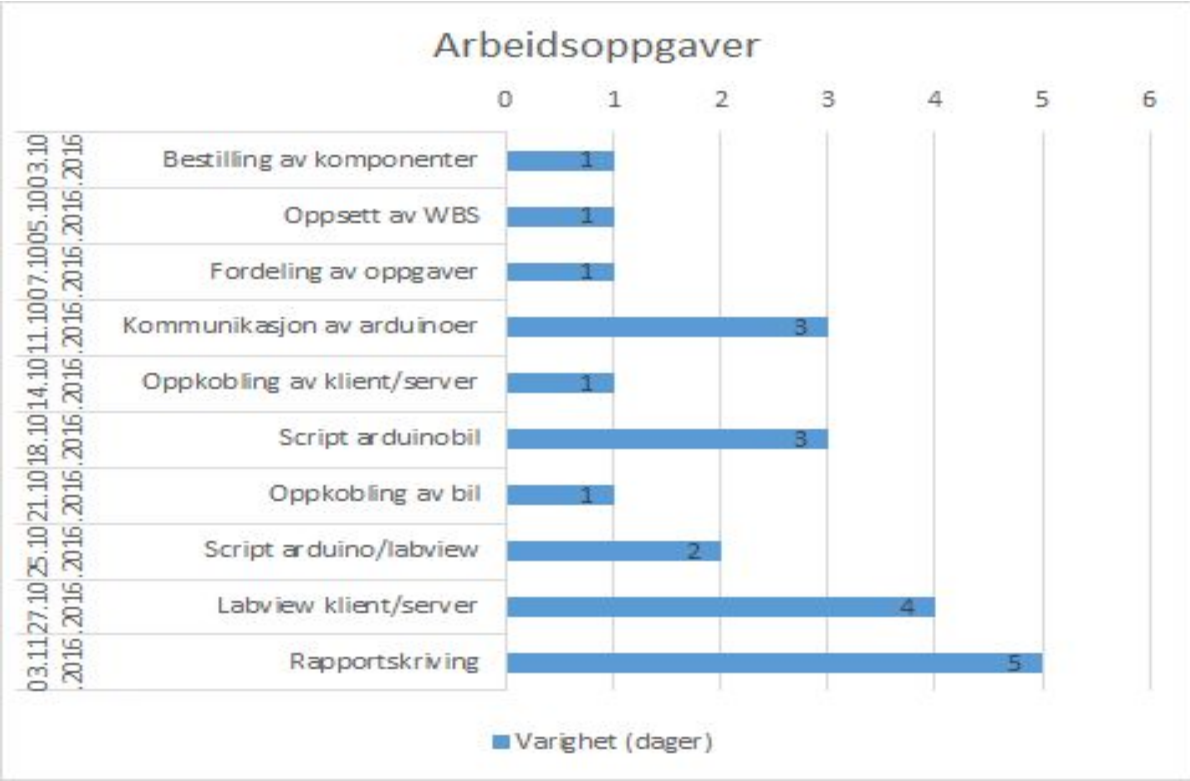
Innledning

Hele prosessen skjer ved at en Arduino på klientsiden får instruksjonsverdier fra en PS4-kontroller som blir videresendt til server ved hjelp av TCP-kommunikasjon. Deretter mottar en annen Arduino instruksjonsverdiene på serversiden serielt, som til slutt blir sendt ved hjelp av en transmitter til den tredje Arduinoen som styrer DC-motorene på bilen. Problemstillingen har en høy vanskelighetsgrad på grunn av bruk av flere mikrokontrollere, i tillegg til LabVIEW «klient-server» og at den har en «wow»-faktor i og med at det skjer så mange kommunikasjoner samtidig.

Første steg i prosjektet var å lage en WBS for å bryte ned problemstillingen i mindre arbeidsoppgaver og se hva som skulle gjøres. Det ble laget en liste over komponentene som var nødvendig for prosjektet, og bestilte dem med en gang. Deretter ble arbeidsoppgavene delt opp og prosjektet var i gang.

Arbeidsoppgavene ble oppdelt i følgende punkter:

- Kommunikasjon mellom klient og server
- Trådløs kommunikasjon mellom to arduinoer
- Programmering av bil
- Kommunikasjon mellom arduino og LabVIEW



Utstyrsliste

Type	Manufacturer	Utstyr	Antall
Microcontroller	Arduino	Arduino UNO R3	3
Microcontroller	Keyes	USB Host Shield	1
Controller	Sony	PS4 Wireless Controller	1
H-bridge	-	L293D/L293DNE	2
Radio Transceiver	-	NRF24L01	2
Karosseri	-	Bil-karosseri	1
Motor	-	DC-motor	4
Batteri	-	9V batteri	2
Brett	-	Koblingsbrett	1 (2)
Kabler	-	USB-USB	2 (3)
Ledninger	-	-	-
Program	National Instruments	LabVIEW 2015	-
Program	Arduino.cc	Arduino IDE	-

Metodebeskrivelse

Generell metode for kommunikasjon

I alle kommunikasjonsprotokoller, enten det er TCP/VISA (i LabVIEW), Serial.read/write (i Arduino) eller NRF Radioprotokoll (i Arduino) sender programmene pakker på forskjellige lengder. Derfor sender hvert program lengden til pakken (i bytes) den kommer til å sende før den sender selve pakken. Dette gjør at programmet som har behov for å få inn data først kan lese antall bytes den kommer til å få inn, og justere sin neste ''read''-funksjon i forhold til dette. Dette er spesielt kraftig innenfor LabVIEW.

Kommunikasjon mellom Arduinoene

Arduino (Klientsiden) til Arduino (Serversiden)

Arduino på klientsiden kommuniserer med Arduino på serversiden gjennom LabVIEW og TCP-kommunikasjon (j.f. Fig.1). Hvordan Arduino kommuniserer med LabVIEW er detaljert i *Kommunikasjon mellom Arduino og LabVIEW* side 11.

Arduino (Serversiden) til Arduino (Bil)

For å oppnå trådløs kommunikasjon mellom Arduinoen på server sin side og Arduinoen som er på bilen, ble det valgt å bruke NRF24L01 transceiver. Denne transmitteren operer på 2.4 GHz, har lavt strømforbruk og koster lite. I tillegg finnes det et Arduino bibliotek for transmitteren som gjør den enkel og intuitiv å bruke. For oppsett av transmitterne finnes det en veldig bra instruksjon som går litt dypere inn på hvordan transmitteren fungerer (j.f. *NRF24L01* i *Kilder* side 16).

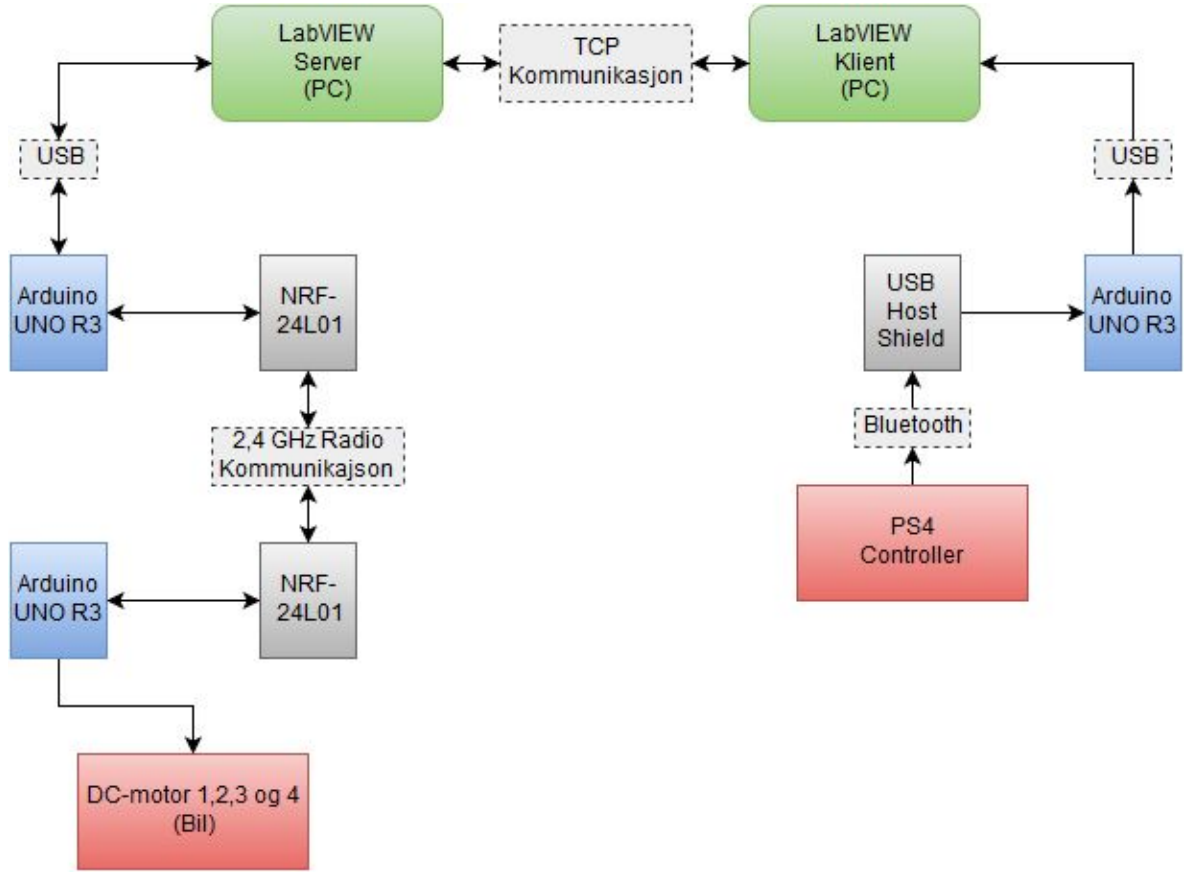


Fig.1 - Prosjektet

NRF24L01 har en kanalavstand på 1MHz som gir 125 mulige kanaler fra 0 til 124. Siden flest WiFi bruker de laveste kanalene har instruksen foreslått at man bruker de øverste 25 kanalene. Det er mulig å velge mellom 4 forskjellige verdier for hvor mye strøm man vil tilføye transmitteren ved å sette inn variablene RF24_PA_MIN, RF24_PA_LOW, RF24_PA_HIGH and RF24_PA_MAX inn i funksjonen "setPALevel(variabelnavn)". Ved å gjøre dette øker man også distansen på hvor langt i mellom transmittere kan kommunisere sammen. Det er også mulig å endre på hvor raskt overføringen av dataen skal være ved å endre på funksjonene "setDataRate()". I scriptet ble det valgt å bruke kanalen 115 for veksling av data, RF24_PA_MAX og en datarate på 250kbps, som var anbefalt for å oppnå lengst mulig kommunikasjons distanse, samt deklarere at den skal sende og ikke motta data:

```
myRadio.setChannel(115); // setter Channel 115
myRadio.setPALevel(RF24_PA_MAX); // max transmitting power
myRadio.setDataRate(RF24_250KBPS); //datarate, laveste, for å få best mulig avstand
myRadio.openWritingPipe( adresses[0]); // Åpner pipen for å kommunisere
```

Koden fungerer slik at den leser av “Strings” som den får serielt og deler dem opp og lagrer som forskjellige variabler i en “struct”:

```
struct variabels // Structen som lagrer verdier fra den serielle kommunikasjonen mellom arduino og LabVIEW servere
{
  int forward, backward, gir, dir;
  String forwardString, backwardString, dirString, girString;
};
typedef struct variabels Variabels;
Variabels movment;
```

Disse verdiene blir så skrevet serielt slik at serveren kan sende feedback til klienten, samt sendt videre ved hjelp av transmitteren. Når transmitteren sender, så sender den hele “structen” og pakkens størrelse:

```
myRadio.write(&movment, sizeof(movment));
```

Se vedlegg for hele koden.

TCP-kommunikasjon mellom LabVIEW-programmene

Under hele prosessen kan en bruker både på server- eller klientsidene henholde seg til en ‘’contextual help’’ på Frontpanelet (j.f. Fig.2). Denne informasjonen forklarer hva en har muligheten til å gjøre og hva slags input programmet venter på.

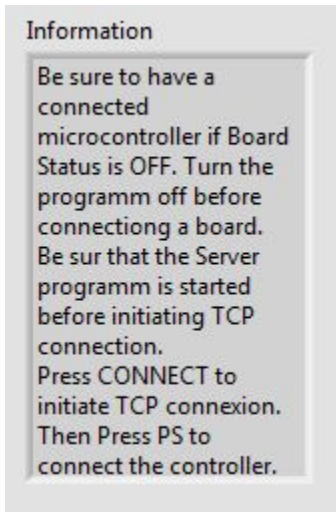


Fig.2 - Contextual help

For å initialisere TCP-kommunikasjon mellom PC-ene trenger brukeren på klientsiden å kunne IP-adressen til Serveren. Med en gang programmet starter skriver derfor LabVIEW IP-address og Domain til denne PC-en, som vist i Fig.3.

Denne funksjonen er implementert både i Klient- og Serverprogrammene. Deretter kan brukeren på serversiden enkelt gi informasjonen til brukeren på klientsiden, sjekke at TCP-connection er gyldig og slutte å overvåke Serveren (Klient får full kontroll over bilen og den generelle prosessen).

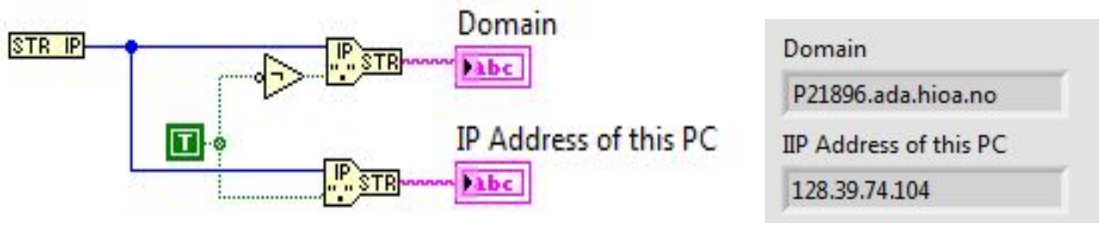


Fig.3 - IP

Klient

Når en bruker på klientsiden starter programmet utfører LabVIEW to funksjoner i parallell. Den ser etter et Arduinobrett (j.f. *Kommunikasjon mellom Arduino og LabVIEW*, side 11 og Fig.13) samtidig som den venter på at brukeren fyller ut ''Remote port'' og ''Address IP of remote PC'' og trykker så på CONNECT. Programmet sjekker at begge input-vinduene ikke er tomme, det vil si at brukeren egentlig har skrevet inn en IP-adresse og en port, satt til 22022 som default (j.f. Fig.4).

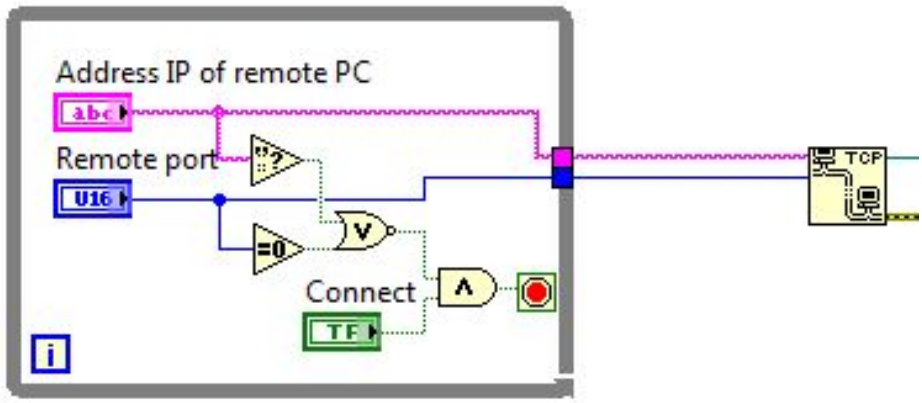


Fig.4 - Klient TCP

Deretter kommer programmet i loopen som skriver til TCP (j.f. Fig.5). Disse funksjonene (de to første blokkene) venter på at programmet fikk lest fra Arduino (j.f. *Kommunikasjon mellom Arduino og LabVIEW*, side 11). Deretter leser programmet feedback-en fra Serveren (j.f. *Feedback* side 10).

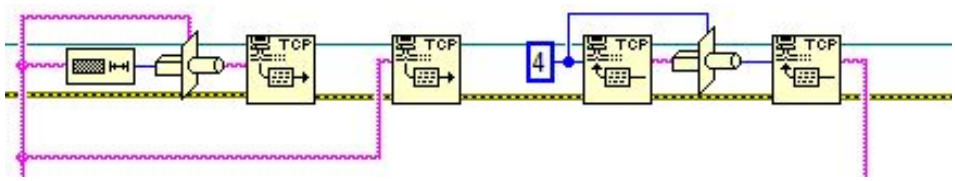


Fig.5 - Klient TCP kommunikasjon

Server

Serveren utfører lignende operasjoner som klienten ved programmets oppstart. Men mens klienten må koble seg til et eksisterende nettverk, initialiserer serveren nettverket og venter på en godkjent TCP connection (Fig.6). For sikkerhets skyld skriver LabVIEW informasjon om ''Domain'' og ''IP Address of remote PC'' til frontpanelet, så brukeren lett kan holde oversikt.

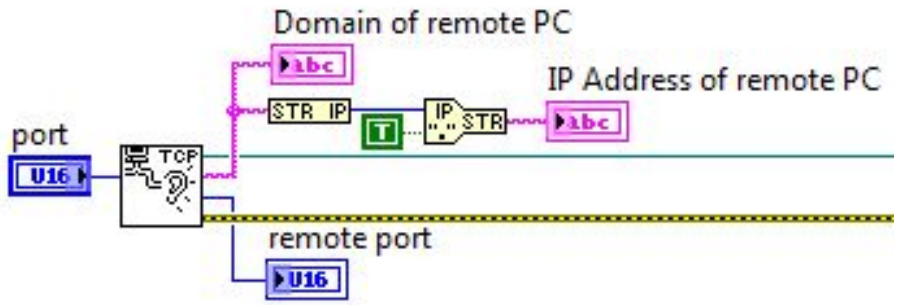


Fig.6 - Server TCP

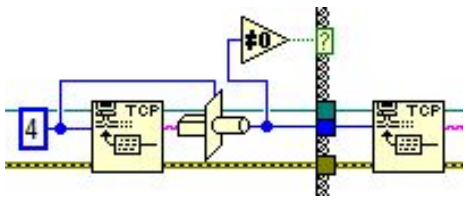


Fig.7 - Server TCP kommunikasjon (1)

Deretter leser Serveren fra TCP og venter på at den har lest inn antall bytes til neste string for å gå videre (Fig.7). Dataen den leser sendes så videre til VISA-protokollen (j.f. *Kommunikasjon mellom Arduino og LabVIEW*, side 11).

Senere venter programmet på at den har fått lest fra VISA for å skrive feedback-en til TCP (j.f. *Feedback* side 10). På samme måte som før sendes først antall bytes den leser og så selve feedback-en (Fig.8).

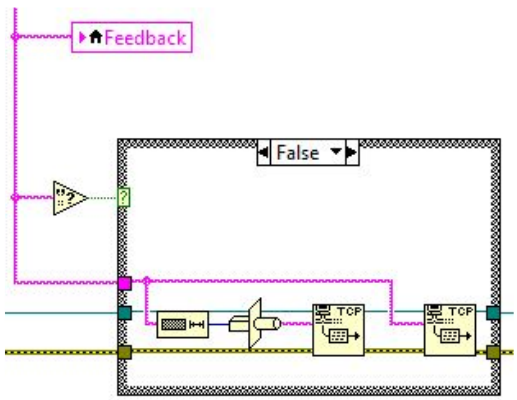


Fig.8 - Server TCP kommunikasjon (2)

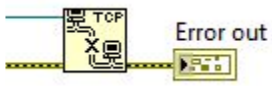


Fig.9 - TCP Close

Når programmene avslutter (etter at brukeren stoppet programmene med knappen STOP PROGRAM) går TCP-protokollen til TCP-close og lukker kommunikasjonen (Fig.9).

Feedback

Feedback er en viktig funksjon for at brukeren på Klientsiden skal ha full kontroll over bilen, særlig når man tenker på fjernkontroll der brukeren ikke ser bilen direkte (noe som er et viktig poeng i videreutvikling av prosjektet). Derfor er det slik at det sendes en beskjed når Arduino på Serversiden får inn data som forklarer hva kortet har fått inn (j.f. Fig.10 & 11) og hvilken funksjon som utføres. På Serveren er det to informasjonsvinduer ‘’Remote Command‘‘ og ‘’Remote Information’’ (j.f. Fig.10). LabVIEW sorterer ikke dataen den får gjennom TCP- eller VISA-kommunikasjon, i ‘’Remote Information’’ viser den alt den får gjennom TCP og i ‘’Remote Command’’ alt som er formet som en gyldig kommando til bilen, men sender videre begge deler uansett. Med tanke på videreutvikling er dette viktig, siden det er fullstendig mulig å lage nye bilfunksjoner som tar imot andre type argumenter - og da er det nyttig at LabVIEW ikke hindrer informasjonen.

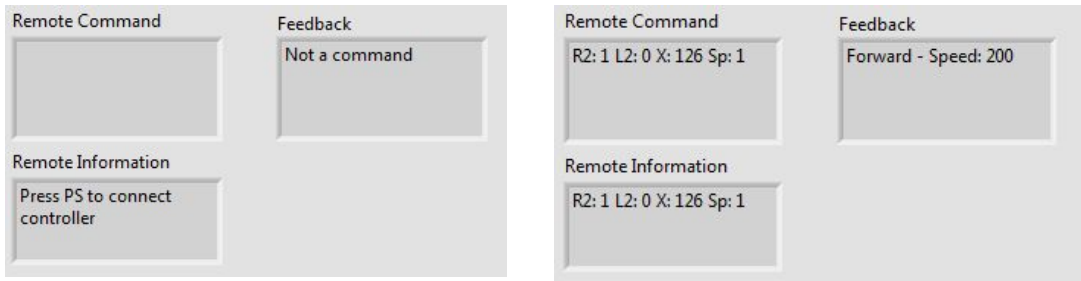


Fig.10 - Server Feedback

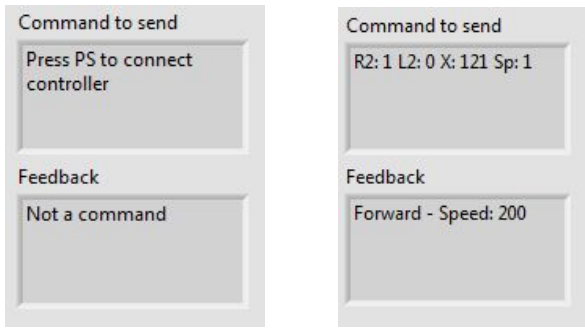
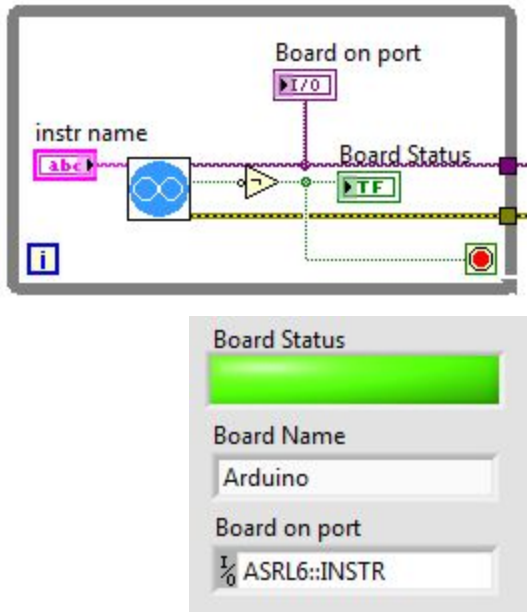


Fig.11 - Klient Feedback

Kommunikasjon mellom Arduino og LabVIEW

I LabVIEW-programmene er det en SubVI som ser etter et eksternt kort på en USB-port (j.f. Fig.12). Programmet skriver ut statusen (ON hvis Arduino er funnet) og går videre kun når kortet er funnet. Deretter bestemmes baud rate og VISA-kommunikasjon åpnes. SubVI er detaljert i Fig.13.

Fig.12 - Arduino board detect



For å få en kommunikasjon mellom arduino og LabVIEW brukes VISA-read/write. Arduino sender seriell kommunikasjon til PC-en via USB som LabVIEW kan lese av med VISA-read. I samme forstand sender PC-en informasjon tilbake til arduinoen med seriell kommunikasjon med VISA-write.

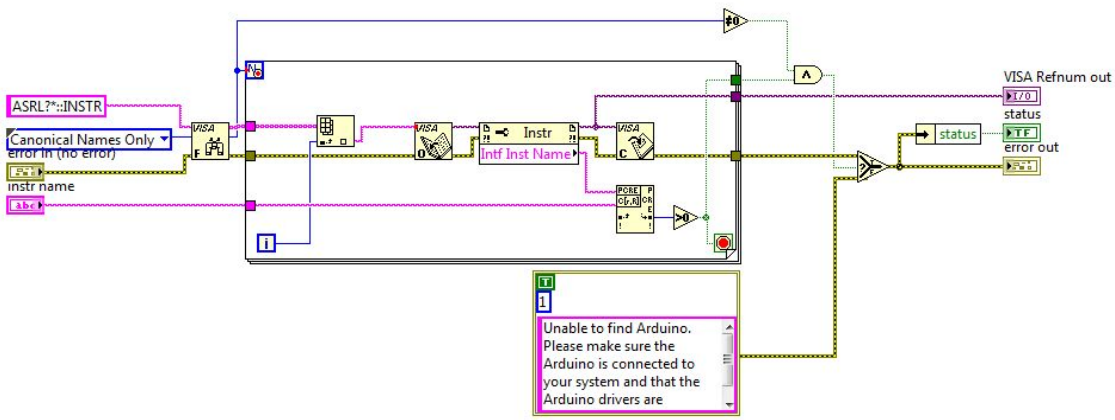


Fig.13 - Arduino board detect (SubVI)

Programmering av bilen

Arduinobilen er bygget opp av fire DC-motorer som er styrt av to H-Bridger (L293D) (Fig.14). For å få nok strøm inn i DC-motorene er det koblet to 9V batterier i serie.

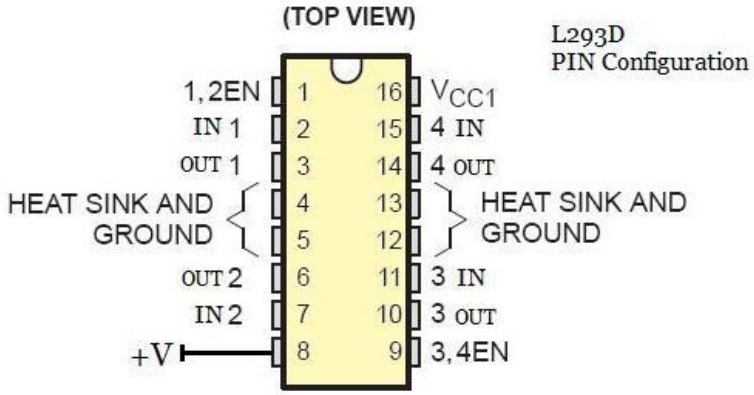


Fig.14 - H-Bridge L293D

9V-strømmen ble så viderekoblet inn i +V-inngangen i serie på begge H-Bridgepinnene. Alle jordpinnene ble koblet til Ground, og VCC1-pinnene til 5V på arduino. Siden motorene hele tiden skal være på når batteriet er koblet til, ble det valgt å direktekoble "enable"-pinnene på H-Bridgen til 5V på koblingsbrettet. Dette sparte ledningsplass og komprimerte Arduinokoden. IN1-pinnene på den ene H-Bridgen er seriekoblet til IN2 på den andre H-Bridgen og motsatt. Med dette er det mulig å styre begge retninge til alle DC-motorene ved hjelp av kun fire PMW-pinner istedenfor for 8. Grunnen til at IN1 og IN2 er koblet til PMW-pinnene er for å kunne styre farten til motorene som vist i arduinokoden her.

```
int motor_speed(int  var)  // Funksjon som re
{
  switch (var)
  {
    case 1:
      return 200;
      break;
    case 2:
      return 220;
      break;
    case 3:
      return 240;
      break;
    case 4:
      return 254;
      break;
    default:
      return 0;
      break;
  }
}
```

NRF24L01 transmitteren er koblet på samme måte som Aruinoen på server sin side (j.f. Fig.15).

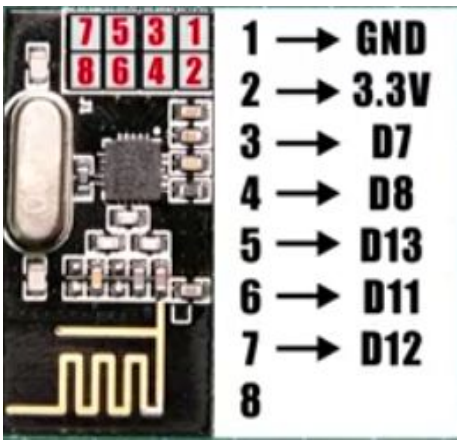


Fig.15 - NRF24L01 koblingsskjema

I motsetning til koden på server sin side, skal transmitteren bare motta data. Dette defineres i koden ved å bytte ut myRadio.openWritingPipe, med myRadio.openReadingPipe, og bestemme at den skal begynne å lytte til transmisjonen med myRadio.startLIstening:

```
myRadio.openReadingPipe(1, adresses[0]); // Apner "pipen" for å kommunisere
myRadio.startListening(); // starter å "lytte" på den oppkoblede transmittoren
```

Arduinoen lagrer så disse verdiene i “Structen” lik definert som i Arduinoen på Server sin side, og utfører funksjoner ut ifra hva som blir lagret. Se vedlegg for koden til Arduino-bilen.

Programmering av PS4-kontroller

PS4-kontroller kommuniserer trådløst og sender data gjennom et Bluetoothanlegg. Derfor kobles det en KEYES USB Host Shield til Arduinokortet på klientsiden: USB Host Shield tar imot Bluetoothsignalet via en USB/Bluetoothpinne som er koblet til PS4-kontrolleren. For å rekke å lese signalene fra kontrolleren må baud-raten settes høyere enn den vanlige 9 600 (115 200 i dette tilfelle).

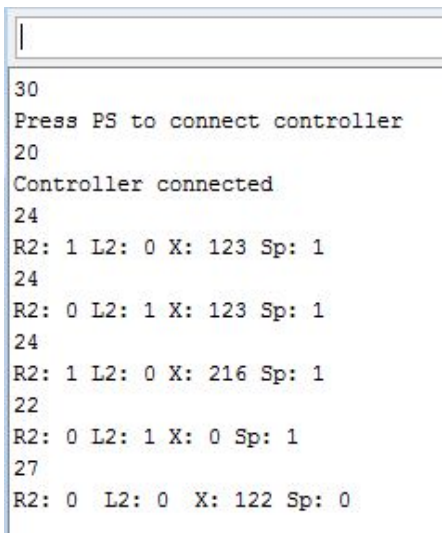


Fig.16 - Arduino Serial port

Ved hjelp av PS4BT.h-biblioteket brukes det mange ferdigbygde funksjoner som lar oss bestemme hvilke knapper fra kontrolleren vi ønsker å bruke. For dette prosjektet brukes R1, R2, L2, venstre joystick, kryss, firkant, trekant og sirkel. Dette Arduinoprogrammet ‘’bestemmer’’ ingenting i seg selv, den sender bare videre til PC-en som tar av seg å sende videre til bilen. Arduino sender da først en string som inneholder antall bytes den kommer til å sende og så en string som inneholder verdien til de forskjellige kontrollene (j.f. Fig.16).

Diskusjon

I dette prosjektet brukes det mange forskjellige utstyr og systemer, fra TCP- til Bluetooth- og Radio-kommunikasjon, både software og hardware. Ingen av de ulike stegene, enten det er trådløst kommunikasjon eller server/klient oppsett, kan inneholde eller føre til feil, da alle programmene må samarbeide for at ønsket funksjonene skal kunne utføres riktig. Til sammen er dette prosjektet veldig relevant med studieløpet innen Elektronikk og Informasjonsteknologi og ikke minst lærerikt.

Dessverre benyttes det ikke analoge verdier fra PS4 Controlleren siden server/klient oppsett ikke klarte å håndtere en kontinuerlig strøm av data, og feedbacken er tregere enn optimalt. Dessuten utfører ikke bilen mange funksjoner.

Konklusjon

Noen av bestilte komponenter som var avgjørende for prosjektet kom ganske sent i prosessen (Derfor er det enveiskommunikasjon mellom Arduinoene med NRF modullen). Med tanke på prosjektgjennomføring er det viktig å bestille komponentene tidsnok.

Med tanke på videreutvikling av prosjektet kan det være interessant å forske på hvordan man kan få sendt en kontinuerlig strøm av data i et toveis kommunikasjonssystem slik at man styre bilen med analoge verdier i sanntid for bedre styring. På samme måten kan bilen i fremtiden utføre flere funksjoner, som f.eks. inneholde et kamera sånn at brukeren ikke trenger å se bilen direkte, men kan kontrollere den på avstand, eller ha med seg f.eks. målingsinstrumenter, armer, kraftigere motorer...

Kilder

NRF24

https://github.com/TMRh20/RF24

https://arduino-info.wikispaces.com/Nrf24L01-2.4GHz-HowTo

L293D:

http://www.ti.com/lit/ds/symlink/l293.pdf

PS4BT biblioteket

https://github.com/felis/USB_Host_Shield_2.0

Arduino i LabVIEW

https://github.com/marcomauro/Arduino-LabVIEW/tree/master/arduinoLabVIEW