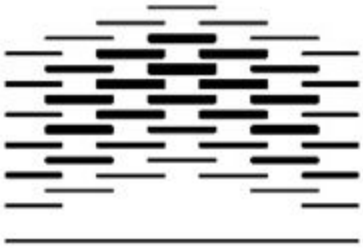


Semesterprosjekt

Vedlegg

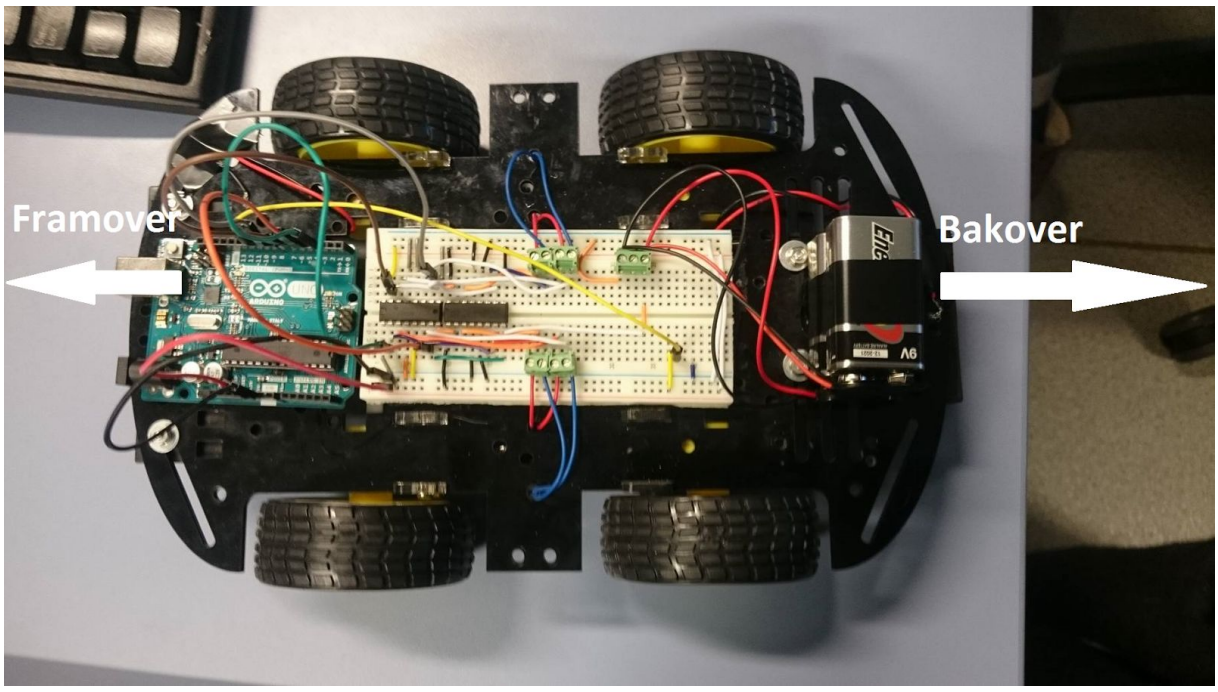
PC-basert instrumentering & kommunikasjonsnett

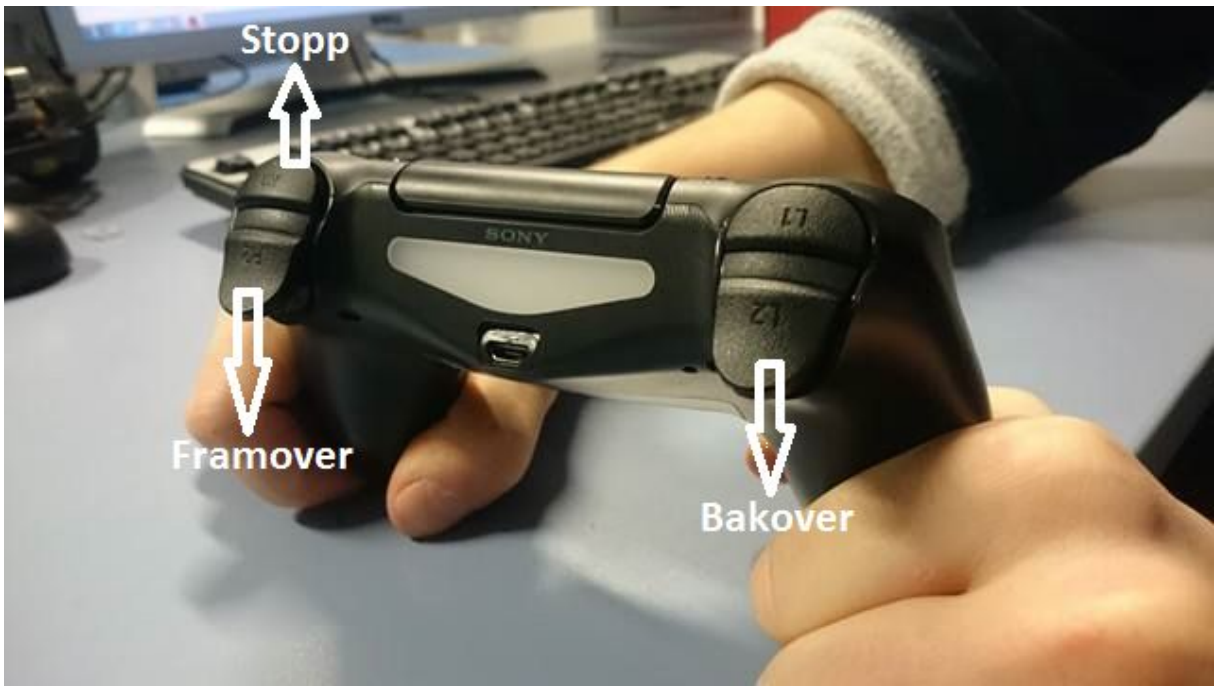
ELTS2200, Høst 2016

Christophe Plaissy, s300344
Mark Strømme s300357,
Kristina Waaler s305155

Innholdsfortegnelse

Bruksanvisning





OBS! Sjekk at baud-rate er satt på 115 200 på begge sider.

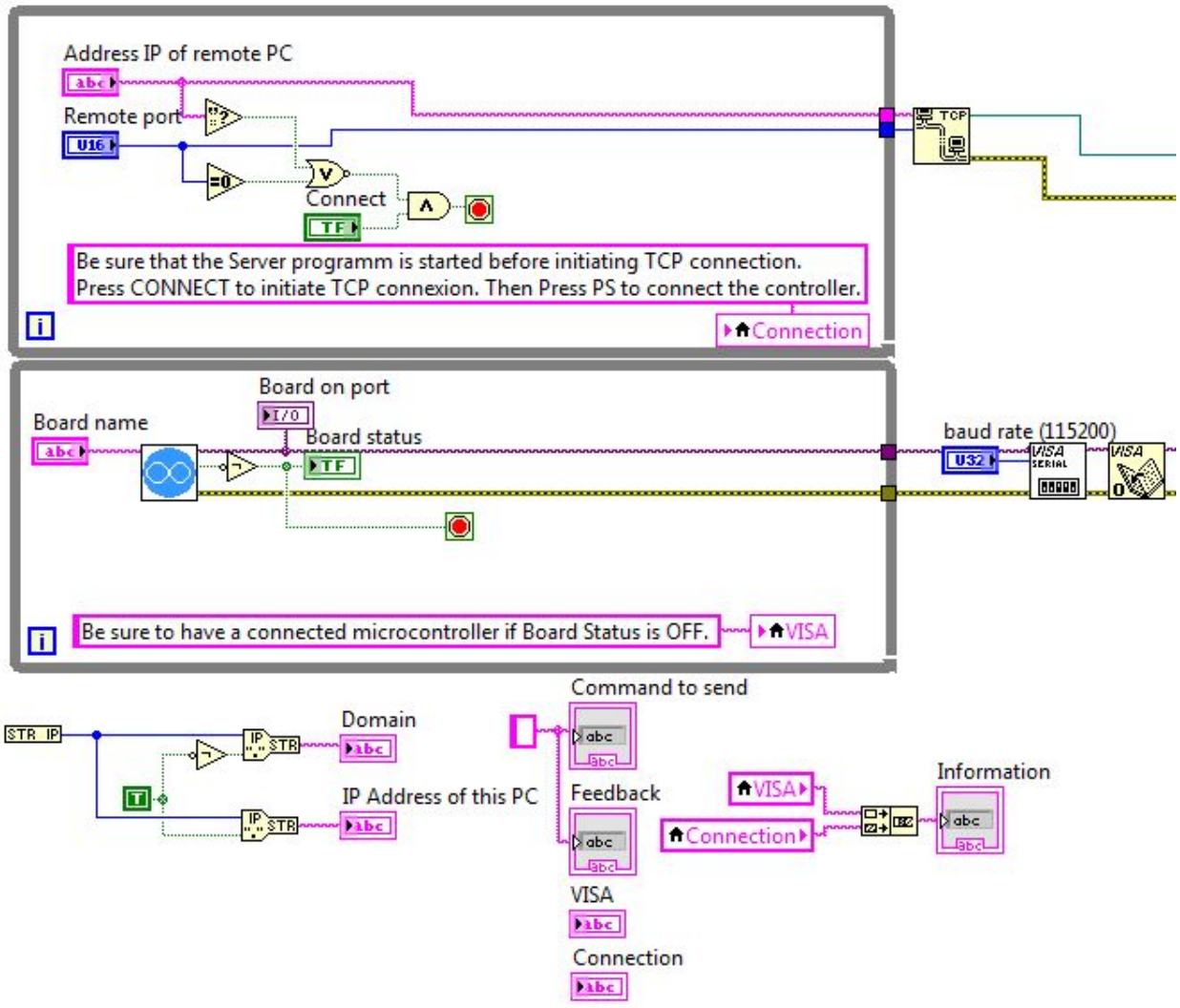
OBS! Sjekk at Controlleren er av før å starte programmet. Er det ikke nok å trykke på PS (Skru på/av kontroll), så trykk samtidig på PS og SHARE.

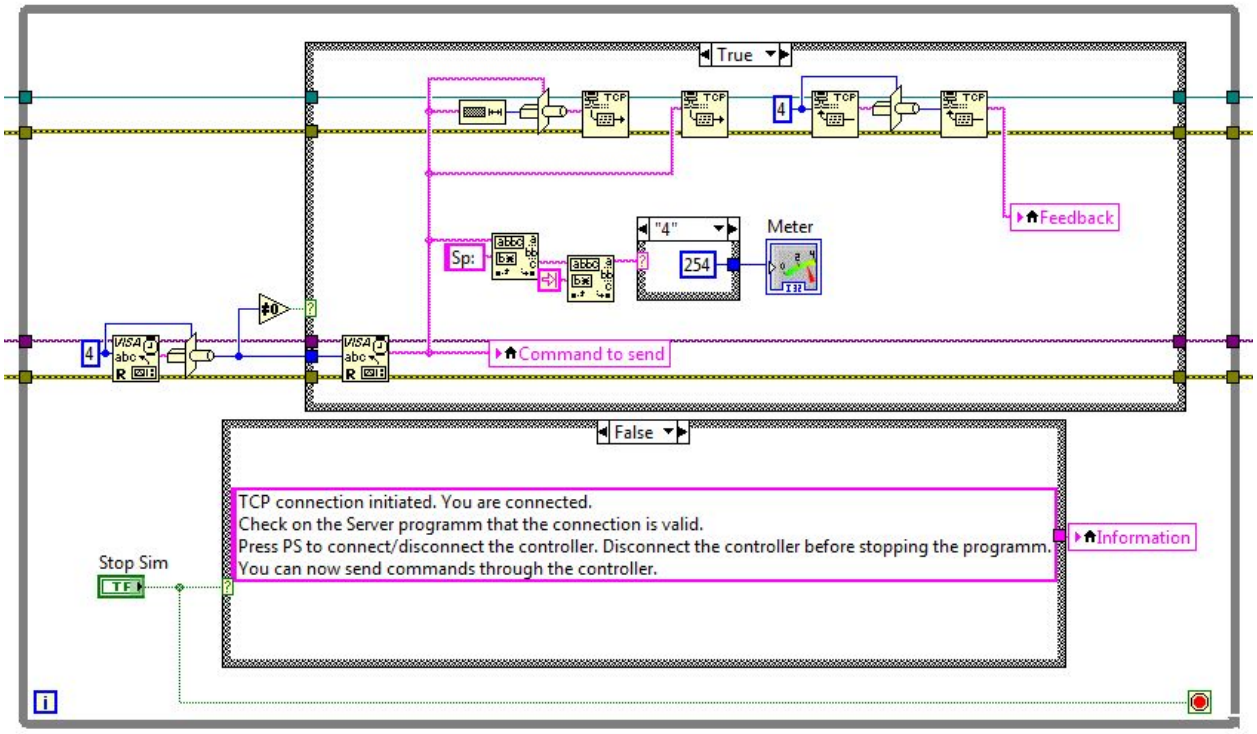
OBS! Velg retning (joystick) og fart før å trykke på Framover/Bakover.

Henhold deg til ‘’Information’’ fra LabVIEW for å vite hvor du er og hva neste steg er.

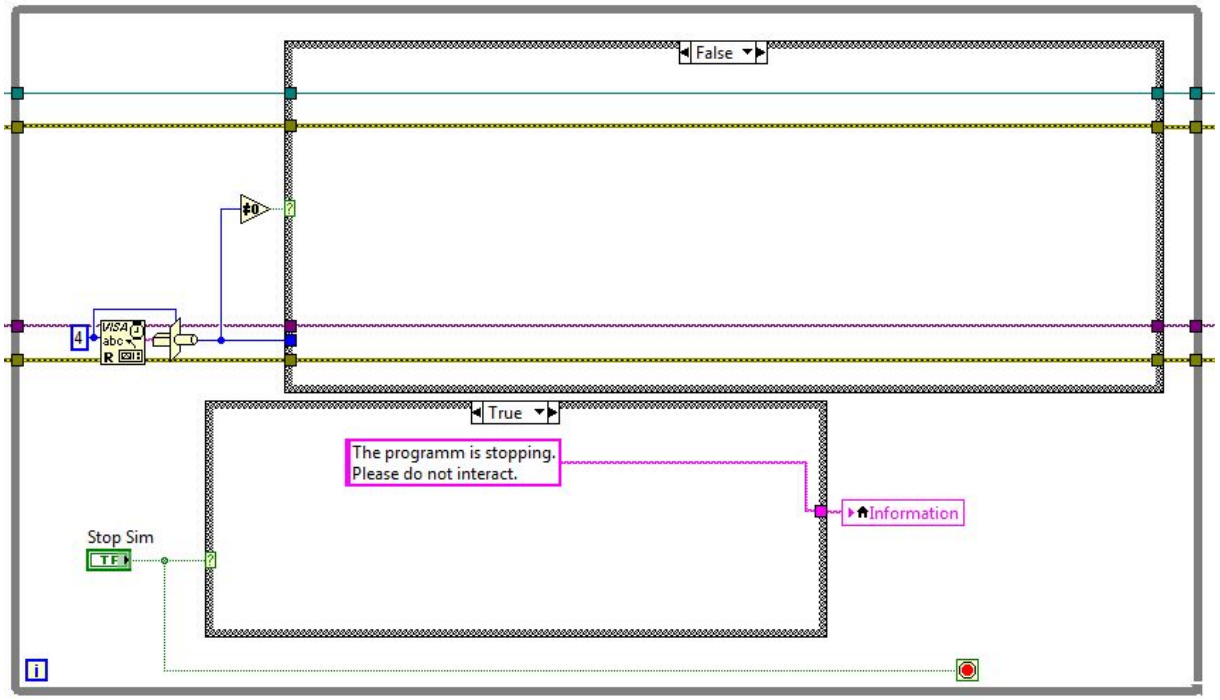
LabVIEW

LabVIEW-Klient

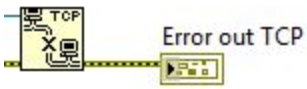




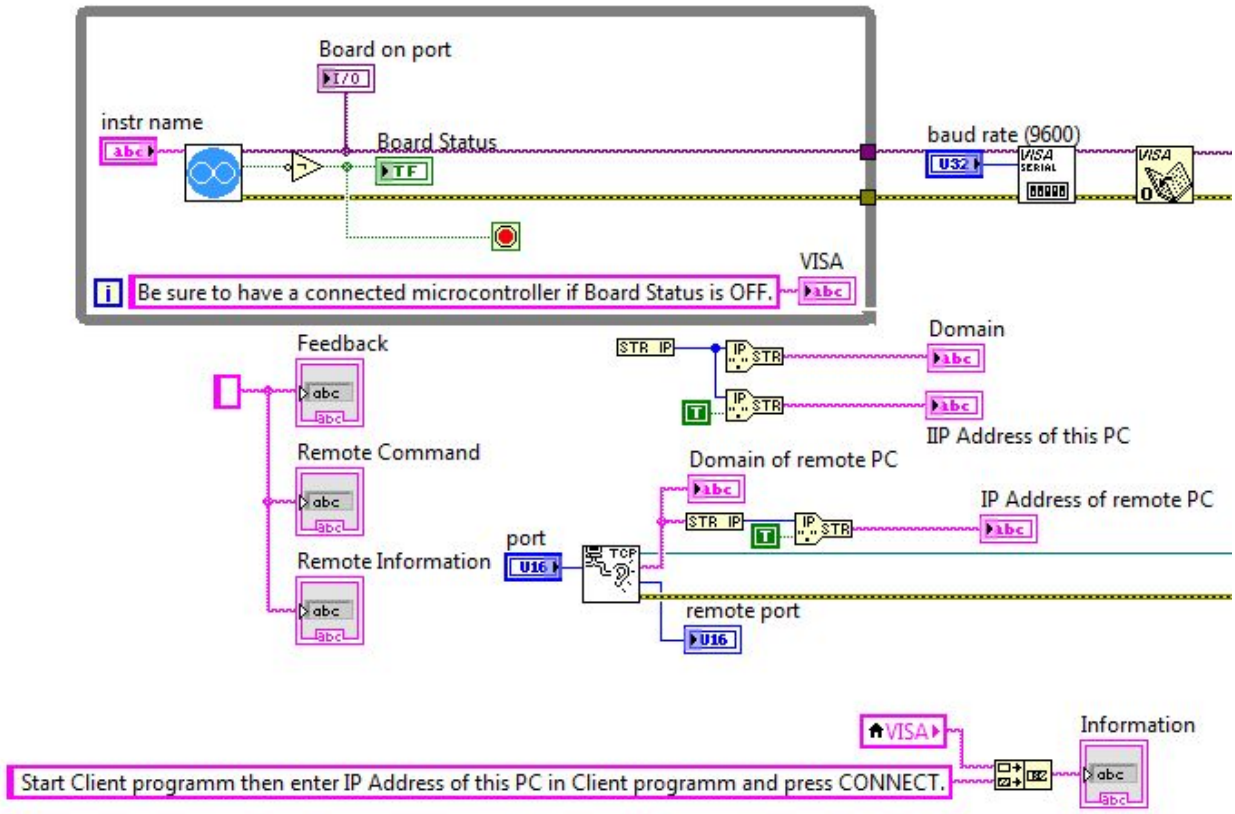
False-tilstand:



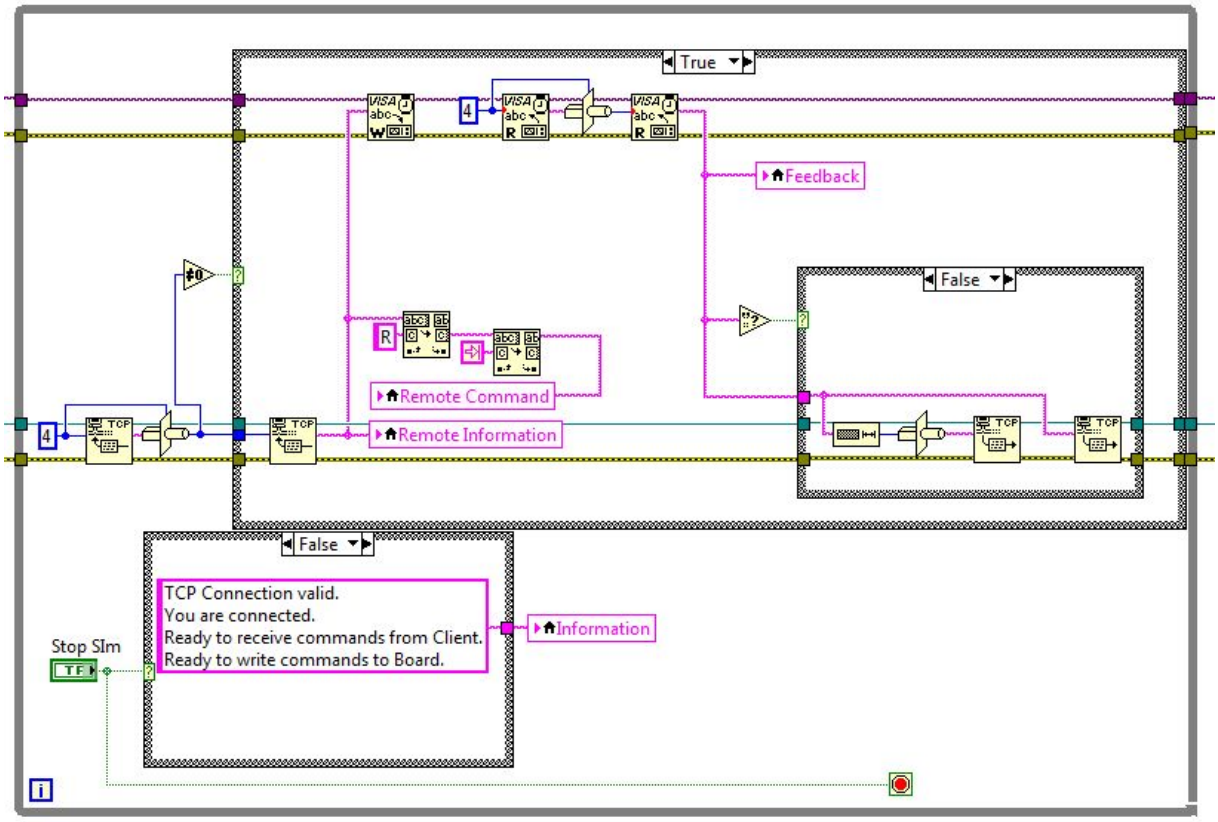
Close:

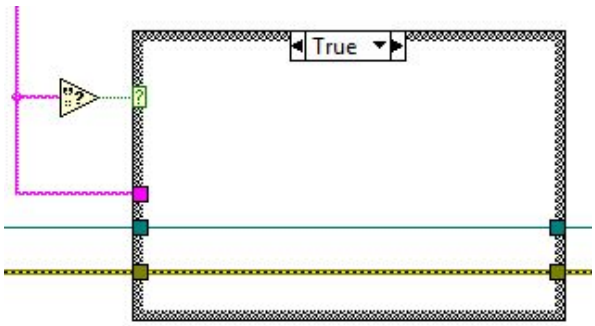


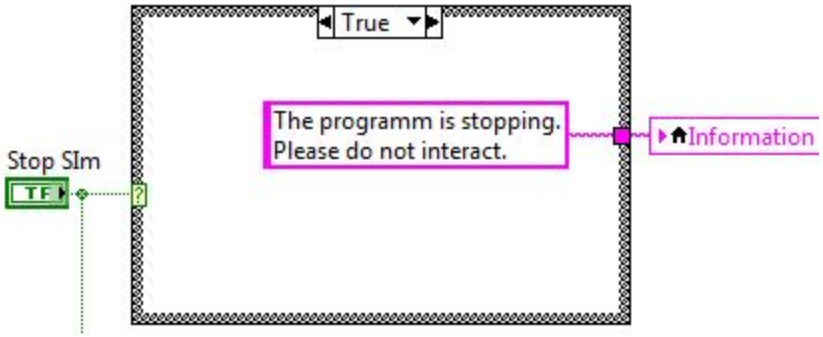
LabVIEW-Server



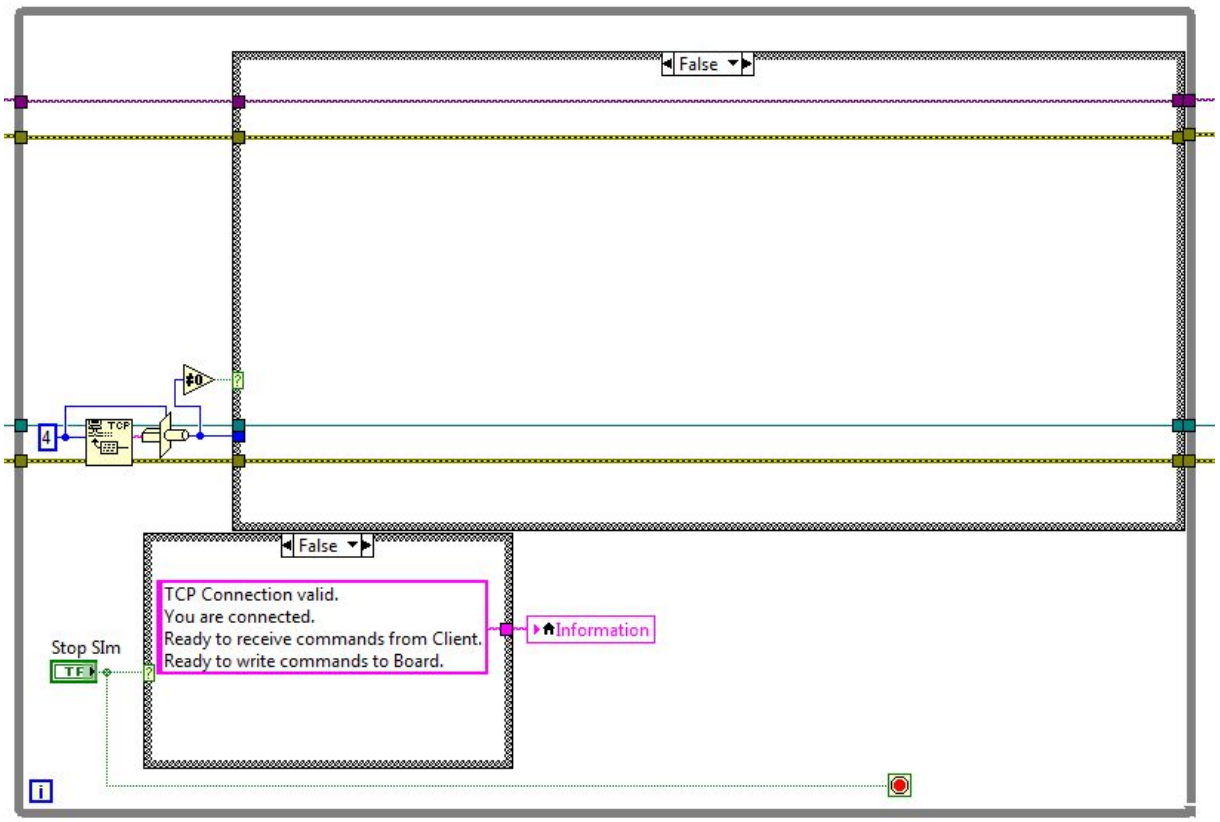
True-tilsand:



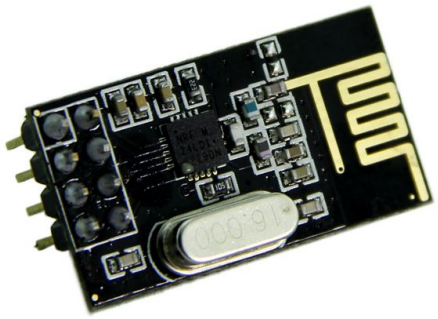




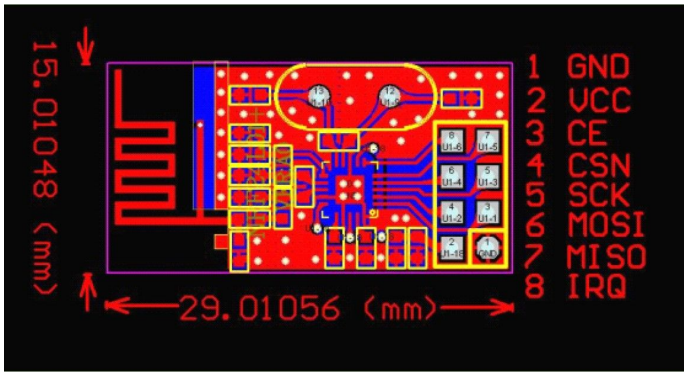
False-tilstand:


False
TCP
4
≠0
False
TCP Connection valid.
You are connected.
Ready to receive commands from Client.
Ready to write commands to Board.
Information
Stop SIm
TF
i

NRF24L01



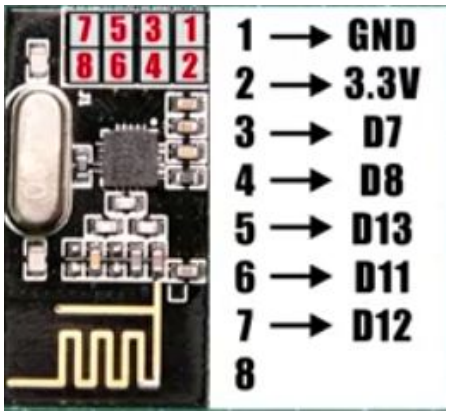
Datasheet for NRF24L01:



Library for NRF24L01:

https://github.com/TMRh20/RF24

Oppkobling for NRF24L01:



Kode Arduino (Server siden)

```
#include <SPI.h>
#include "RF24.h"

RF24 myRadio(7, 8);
byte adresses[][6] = {"0"}; // PIPE adress til 0

String stringToSend = ""; //Lagrer feedbacken som skal sendes fra brettet til LabVIEW serveren
const String frwrd = "Forward - ", frwrdLft = "Forward left - ", frwrdRght = "Forward right - ";
//Variablene som skal skriver til feedback
const String bckwrd = "Backward - ", bckwrdLft = "Backward left - ", bckwrdRght = "Backward
right - ";
const String spd = "Speed: ", stp = "Stop", error = "To many inputs", notacmd = "Not a
command";

int i, motorSpeed;
const int Direction[] = {2, 3, 4, 5, 6};
const int Back = Direction[0], Left = Direction[1], Stop = Direction[2], Right = Direction[3], Go =
Direction[4];

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

void setup()
{
  for (i = 0; i < 5; i++)
          pinMode(Direction[i], OUTPUT);

  Serial.setTimeout(5);
  Serial.begin(115200);

  myRadio.begin();
  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
```

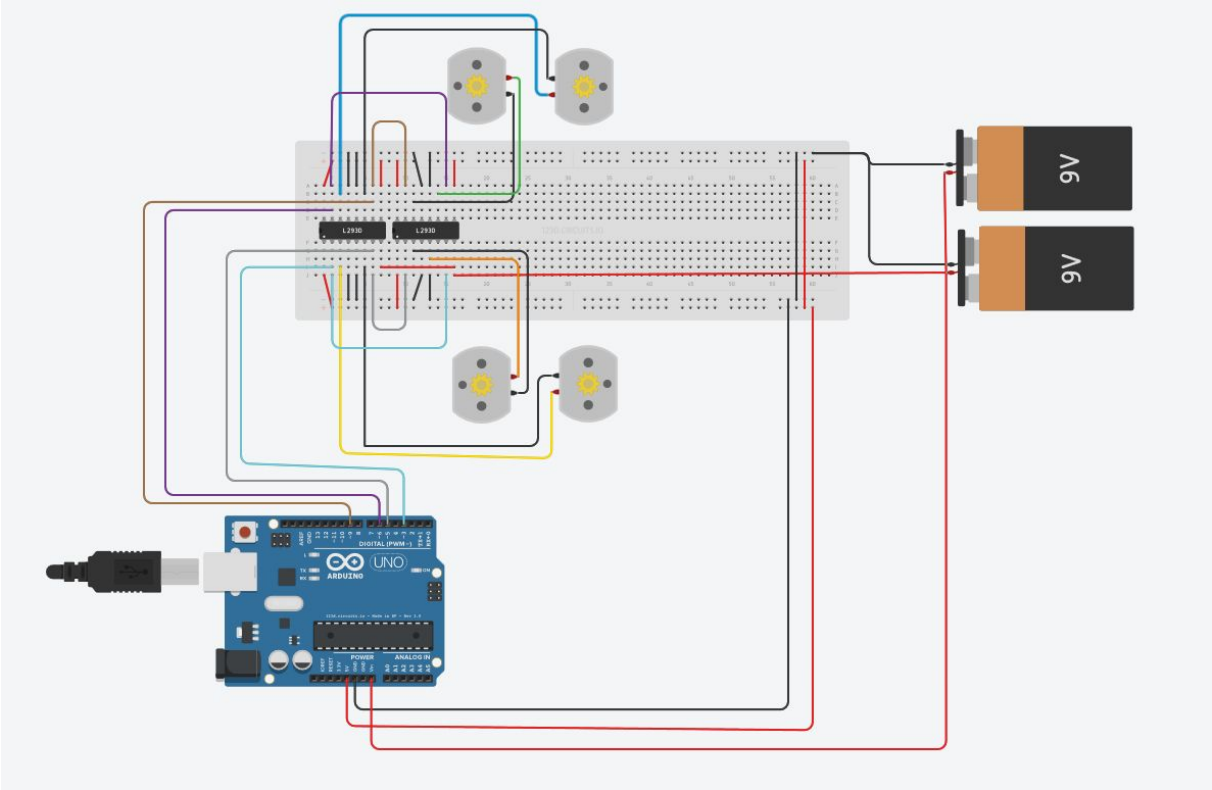
```
  myRadio.openWritingPipe(adresses[0]); // Åpner pipen for å kommunisere
}

void loop()
{
  movment.forwardString = Serial.readStringUntil(' '); // Skjekker om første ord i seriellen er R2
  if (movment.forwardString == "R2:")
  {
        serial_read();
        myRadio.write(&movment, sizeof(movment));
        motorSpeed = motor_speed(movment.gir);
        if (movment.forward == 1 && movment.backward == 0) // hvis R2 på PS4 kontrollen er
trykket
        {
                if (movment.dir > 50)
                {
                        lights_out();
                        digitalWrite(Go, HIGH);
                        digitalWrite(Right, HIGH);
                        stringToSend = frwrdRght + spd + String(motorSpeed) + "\t"; // lagrer
stringen som skal sendes i feedback
                }

                else if (movment.dir < -50)
                {
                        lights_out();
                        digitalWrite(Go, HIGH);
                        digitalWrite(Left, HIGH);
                        stringToSend = frwrdLft + spd + String(motorSpeed) + "\t";
                }

                else
                {
                        lights_out();
                        digitalWrite(Go, HIGH);
                        stringToSend = frwrd + spd + String(motorSpeed) + "\t";
                }
        }

        if (movment.backward == 1 && movment.forward == 0) // skjekker om L2 er presset på
kontroller
        {
                if (movment.dir > 50)
```

```
                {
                        lights_out();
                        digitalWrite(Back, HIGH);
                        digitalWrite(Right, HIGH);
                        stringToSend = bckwrdRght + spd + String(motorSpeed) + "\t";
                }

                else if (movment.dir < -50)
                {
                        lights_out();
                        digitalWrite(Back, HIGH);
                        digitalWrite(Left, HIGH);
                        stringToSend = bckwrdLft + spd + String(motorSpeed) + "\t";
                }

                else
                {
                        lights_out();
                        digitalWrite(Back, HIGH);
                        stringToSend = bckwrd + spd + String(motorSpeed) + "\t";
                }
        }

        if (movment.backward == 0 && movment.forward == 0) // når begge L2 og R2 verdiene
er lave stopper alle motrene
        {
                lights_out();
                digitalWrite(Stop, HIGH);
                stringToSend = stp; // lagrer stringen som skal sendes til feedback
        }

        if (movment.backward == 1 && movment.forward == 1) // når begge L2 og R2 verdiene
er høye så vil motorene også stoppe
        {
                lights_out();
                digitalWrite(Stop, HIGH);
                stringToSend = error;
        }
 }

  if (movment.forwardString != "R2:" && movment.forwardString != "") // Hvis stringen som blir
sendt i seriell ikke er gyldig
  {
```

```
        lights_out();
        digitalWrite(Stop, HIGH);
        stringToSend = notacmd; //feedbacken sender stringen Not a command
 }

 if (stringToSend != "") // så lenge tilbakesending av stringen ikke er tom, så vil den returne lengden til stringen samt selve stringen
 {
        Serial.println(stringToSend.length());
        Serial.println(stringToSend);
        stringToSend = ""; // også setter den til ingenting
 }
}

void serial_read() // funksjonen som leser seriell kommunikasjon og lagrer som int variabler
{
        movment.forward = Serial.readStringUntil(' ').toInt();
        movment.backwardString = Serial.readStringUntil(' ');
        movment.backward = Serial.readStringUntil(' ').toInt();
        movment.dirString = Serial.readStringUntil(' ');
        movment.dir = Serial.readStringUntil(' ').toInt() - 124; // Siden X verdiene er 124 i midten, så vil vi sette den til 0 istedet
        movment.girString = Serial.readStringUntil(' ');
        movment.gir = Serial.readStringUntil('\0').toInt();
}

int motor_speed(int  var)  // Funksjon som returenerer PMW verdien til motor utifra hvilket gir
{
 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;
  }
}

// funksjonen som tar lysene av
void lights_out()
{
  for (i = 0; i < 5; i ++)
  {
            digitalWrite(Direction[i], LOW);
  }
}
```

Koblingsskjema bil



Kode bil

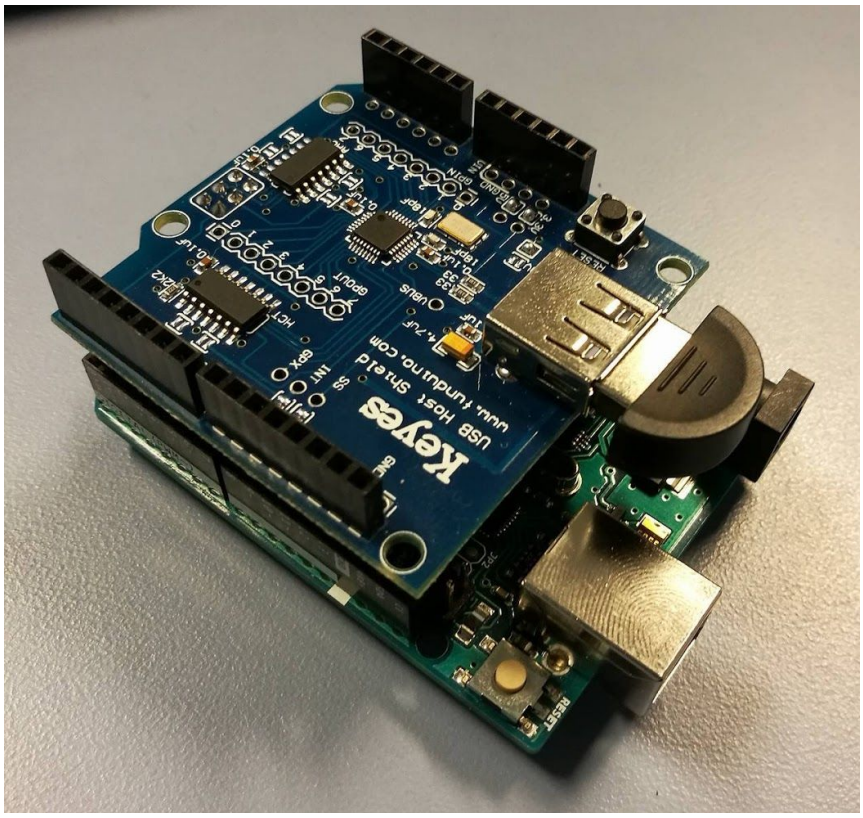
```
#include <SPI.h>
#include "RF24.h"

RF24 myRadio(7, 8);
byte adresses[][6] = {"0"}; // PIPE adress til 0

int motors[] = {3, 5, 6, 9}; // PMW pinenen som motorene er koblet til
int f1 = 3, f2 = 9, b1 = 5, b2 = 6; // f1 = side 1 forover, f2 = side 2 forover, b1 = siden 1 bakover, b2 = side 2 bakover
int motorSpeed; // Variabelen som bestemmer farten til motorene
int i; // teller

struct variabels // Structen som lagrer verdier fra den serielle kommunikasjonen mellom arduino og LabVIEW serveren
{
  int forward, backward, gir, dir;
```

```
  String forwardString, backwardString, dirString, girString;
};
typedef struct variabels Variabels;
Variabels movment;

void setup()
{
  Serial.setTimeout(5); // Setter dalayen for seriell lesing til 5ms
  Serial.begin(115200); // Setter baudrate til 115200

  myRadio.begin();
  myRadio.setChannel(115); // setter kanalen til 115
  myRadio.setPALevel(RF24_PA_MAX); // maksimal veksling kraft
  myRadio.setDataRate(RF24_250KBPS); // datarate til den laveste verdien å oopnå best mulig avstand
  myRadio.openReadingPipe(1, adresses[0]); // Åpner "pipen" for å kommunisere
  myRadio.startListening(); // starter å "lytte" på den oppkoblede transmittoren

  for (i = 0; i < 5; i++)
        pinMode(motors[i], OUTPUT); // setter alle motorpinnene som output

  for (i = 0; i < 5; i++)
        digitalWrite(motors[i], LOW); // setter alle motorene av
}

void loop()
{
  if (myRadio.available())
  {
        myRadio.read( &movment, sizeof(movment));
        motorSpeed = motor_speed(movment.gir); // setter hastigheten til motorene ved å hente gir
variabelen i movment structen og utføre den i motor_speed funksjonen
        if (movment.forward == 1 && movment.backward == 0) // hvis R2 på PS4 kontrollen er trykket
        {
              if (movment.dir > 50) // skjekker om det er noe retningsverdi
              {
                    analogWrite(f1, motorSpeed);
                    digitalWrite(f2, LOW); // svinger mot høyre
                    digitalWrite(b1, LOW);
                    digitalWrite(b2, LOW);
              }
              else if (movment.dir < -50)
              {
                    digitalWrite(f1, LOW); // svinger mot venstre
```

```
                analogWrite(f2, motorSpeed );
                digitalWrite(b1, LOW);
                digitalWrite(b2, LOW);
        }
        else
        {
                analogWrite(f1, motorSpeed);
                analogWrite(f2, motorSpeed); //default kjører ratt fram
                digitalWrite(b1, LOW);
                digitalWrite(b2, LOW);
        }
}

if (movment.backward == 1 && movment.forward == 0) // skjekker om L2 er presset på kontroller
{
        if (movment.dir > 50)
        {
                analogWrite(b1, motorSpeed); // svinger høyre
                digitalWrite(b2, LOW);
                digitalWrite(f1, LOW);
                digitalWrite(f2, LOW);
        }
        else if (movment.dir < -50)
        {
                digitalWrite(b1, LOW); // svinger venstre
                analogWrite(b2, motorSpeed);
                digitalWrite(f1, LOW);
                digitalWrite(f2, LOW);
        }
        else
        {
                analogWrite(b1, motorSpeed); // kjører direkte bakover
                analogWrite(b2, motorSpeed);
                digitalWrite(f1, LOW);
                digitalWrite(f2, LOW);
        }
}
if (movment.backward == 0 && movment.forward == 0) // når begge L2 og R2 verdiene er lave stopper alle motrene
{
        for (i = 0; i < 5; i++)
        {
```

```
                digitalWrite(motors[i], LOW);
            }
        }
        if (movment.backward == 1 && movment.forward == 1) // når begge L2 og R2 verdiene er høye så vil motorene også stoppe
        {
            for (i = 0; i < 5; i++)
            {
                digitalWrite(motors[i], LOW);
            }
        }
 }
}

int motor_speed(int  var)  // Funksjon som returenerer PMW verdien til motor utifra hvilket gir
{
 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;
 }
}
```

Koblingsskjema Controller



Kode PS4 (Klient siden)

```
//inkludering av bibliotekene for styring av PS4 Bluetooth Controller
#include <PS4BT.h>
#include <usbhub.h>

#ifdef dobogusinclude
#include <spi4teensy3.h>
#include <SPI.h>
#endif

int forward = 0, backward = 0, halt = 0;
int valX, speedVal = 1;
int connection = 0;
const String connectON = "Controller connected", connectOFF = "Controller disconnected", request =
"Press PS to connect controller";
const String r2 = "R2: ", l2 = " L2: ", x = " X: ", zero = "0 ", speedString = " Sp: ";
String stringToSend = “”;


USB Usb;
```

```
BTD Btd(&Usb);
PS4BT PS4(&Btd);

void setup()
{
  // Seriell kommunikasjon må skje raskere enn vanlig, for å lese fra Controlleren
  Serial.begin(115200);
  Serial.setTimeout(5);

  #if !defined(__MIPSEL__)
  #endif
  if(Usb.Init() == -1)
  {
          while (1); // Halt
  }
}

void loop()
{
  Usb.Task();
  // Hvis Controlleren er connected
  if (PS4.connected())
  {
          // Skriver det bare en gang
          if (connection == 0 || connection == 2)
          {
                  Serial.println(connectON.length());
                  Serial.println(connectON);
                  delay(1000);
                  connection = 1;
          }

          // Vi bestemmer at R2 og L2 skal sende digitale verdier, men joysticken analoge verdier
          valX = PS4.getAnalogHat(LeftHatX);
          forward = PS4.getButtonClick(R2);
          backward = PS4.getButtonClick(L2);

          // Verdien på farten bestemmes
          if (PS4.getButtonClick(CROSS) == 1)
          {
                  speedVal = 1;
          }
          if (PS4.getButtonClick(SQUARE) == 1)
          {
                  speedVal = 2;
          }
          if (PS4.getButtonClick(TRIANGLE) == 1)
          {
```

```
                speedVal = 3;
        }
        if (PS4.getButtonClick(CIRCLE) == 1)
        {
                speedVal = 4;
        }

        // Sender commandoen
        if (forward > 0 || backward > 0)
        {
                stringToSend =  r2 + String(forward) + l2 + String(backward) + x + String(valX) +
speedString + String(speedVal) + "\t";
                Serial.println(stringToSend.length());
                Serial.println(stringToSend);
        }

        // stopper bilen
        if (PS4.getButtonClick(R1) > 0)
        {
                stringToSend =  r2 + zero + l2 + zero + x + String(valX) + speedString + zero + "\t";
                Serial.println(stringToSend.length());
                Serial.println(stringToSend);
        }

        // stopper bilen og kobler Controlleren av
        if (PS4.getButtonClick(PS))
        {
                stringToSend =  r2 + zero + l2 + zero + x + String(valX) + speedString + zero + "\t";
                Serial.println(stringToSend.length());
                Serial.println(stringToSend);
                Serial.println(connectOFF.length());
                Serial.println(connectOFF);
                connection = 0;
                PS4.disconnect();
        }
 }

 // Hvis Controlleren ikke er connected skriver Arduino dette bare en gang
 else if (connection == 0 || connection == 1)
 {
        Serial.println(request.length());
        Serial.println(request);
        connection = 2;
 }
}
```