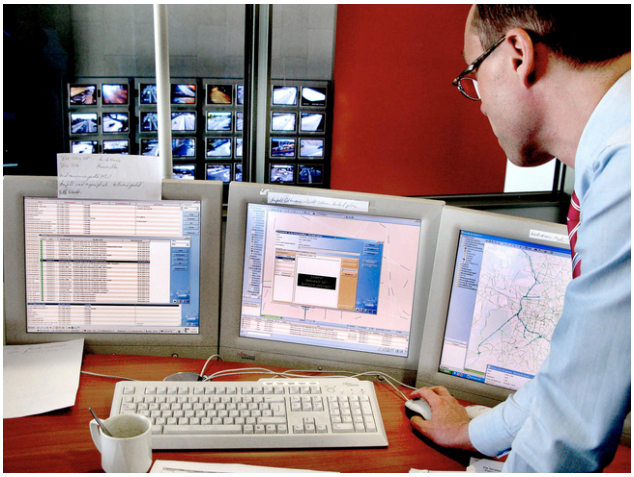
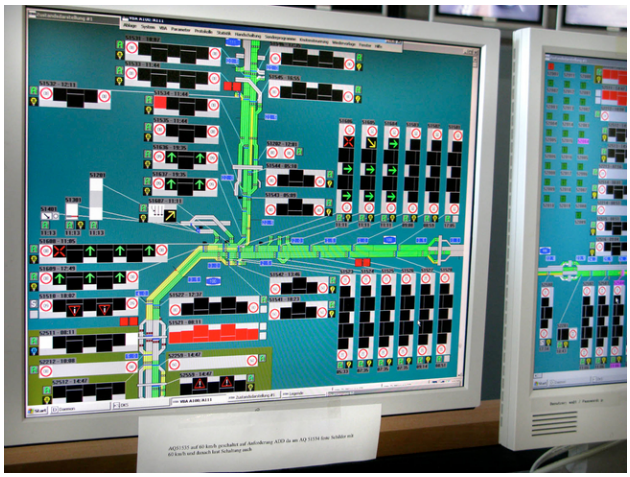
Exercise 3

Trafikstyring

En kæmpe opgave i dagens Danmark (og i resten af verden) er, at få afviklet den stadig stigende trafik på den bedste og hurtigste måde. Dette er en rigtig ingeniør-opgave ☺

Herunder ser du nogle billeder fra trafikstyringscentralen i Berlin





Og herunder ser du hvor galt det kan gå **uden** styring ☺

I denne uge skal du derfor arbejde med trafikstyring.

Du skal desuden lære noget ***meget*** vigtigt, nemlig hvordan man i praksis gør, når man arbejder med software-udvikling (skriver programmer).

Iterativ programudvikling

Når man udvikler software/programmer foregår det ***iterativt***. Det vil sige, at man ***ikke*** skriver ***hele*** programmet på en gang. Man inddeler udviklingen i iterationer. I første iteration skriver man en forud bestemt ***del*** af programmet (f.eks. den del der får nogle lamper til at lyse). Når den del er færdig ***og virker***, forsætter man med anden iteration. Her ***tilføjer*** man den næste forud bestemte ***del*** af programmet (f.eks. den del der spiller en lyd, når den røde lampe lyser). Når den del er færdig ***og virker***, forsætter man med tredie iteration osv. osv. Dette har mindst 3 kæmpe fordele.

1. Du skal ikke kunne overskue ***hele*** programmet på en gang.
2. Hvis noget ikke virker i 3. iteration ***ved*** du, at fejlen ***ikke*** ligger i nogle af de tidligere iterationer – for de ***er*** testet. Dette gør fejlfinding meget nemmere.

3. Hvis chefen, projektlederen eller kunden på et vilkårligt tidspunkt er interesseret i at vide, hvordan det går med projektet, har du ***altid*** noget at vise frem som virker – nemlig den senest færdiggjorte iteration (det har du jo ikke hvis du forsøger at skrive hele programmet på en gang).

Denne vigtige iterative arbejdsproces får du mulighed for at prøve i dagens opgave.

Her skal du nemlig udvikle et program til trafikstyring – dvs. et program der kan styre lyskrydsene på en given vejstrækning. For simpelhedens skyld skal du dog kun lave et program, der kan styre ***en*** lysstander – men på en sådan måde, at den på en fornuftig måde ville kunne fungere sammen med de øvrige lysstandere i et lyskryds og de øvrige lyskryds på en vejstrækning (grøn "bølge"). "Din" lysstander er placeret langs hovedvejen. Dit program skal derfor styre lysstanderen på følgende måde:

1. Lysstanderen skal kunne skifte på korrekt måde mellem "rødt lys" og "grønt lys" med et bestemt ***tidsinterval***.
2. Lysstanderen skal skifte til "rødt lys", hvis en fodgænger trykker på en knap.
3. Lysstanderen skal skifte til "rødt lys", hvis der registreres et køretøj på sidevejen.
4. Operatøren på trafikcentralen skal kunne vælge mellem to forskellige tilstande – en dagstilstand og en aftentilstand. I dagstilstanden skal der ***hele tiden*** skiftes mellem rødt og grønt lys med bestemte tidsintervaller (pkt. 1). I aftentilstanden skal der ***kun*** skiftes til "rødt lys" hvis en fodgænger har trykket på knappen eller der registreres et køretøj på sidevejen (pkt. 2 og 3).

Dette er meget at overskue på en gang. Bl.a. derfor skal du selvfølgelig udvikle programmet iterativt.

Men allerførst en anden lille opgave til at komme i gang på ☺

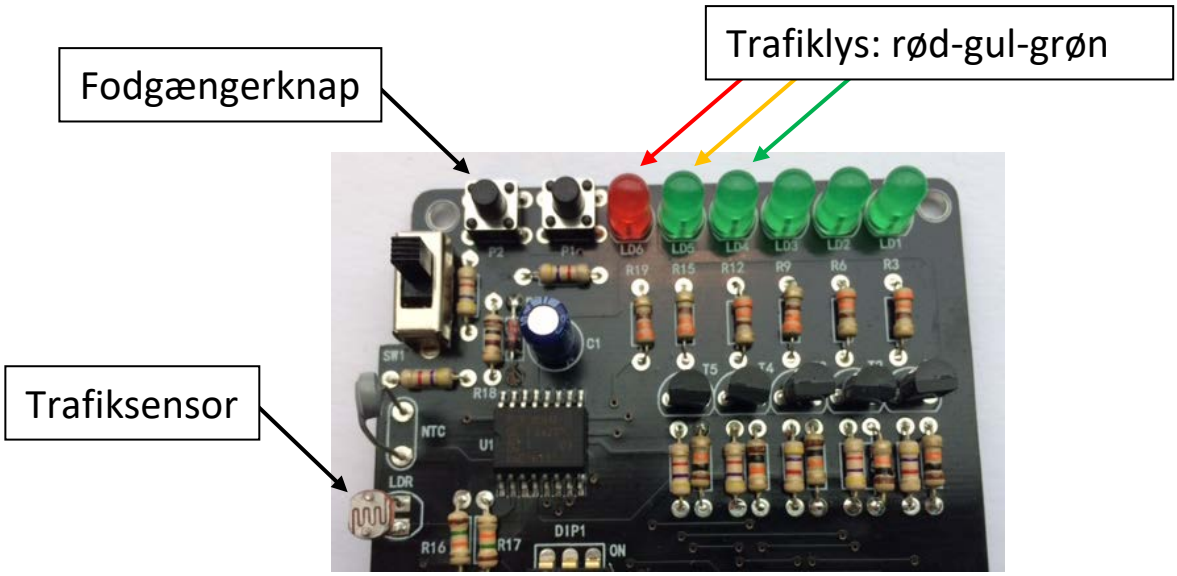
Exercise 3.1 For-løkke

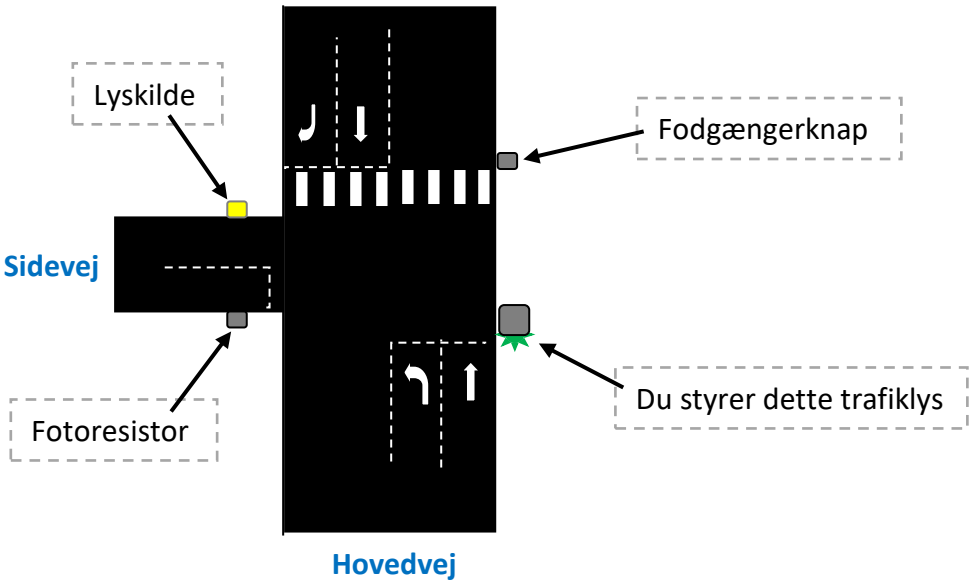
Opret et nyt Visual Studio projekt (Solution Name: "Exercise3" – Project Name: "For-loop"). Skriv heri et lille program der udskriver ***alle*** lige tal fra 50 til 2 i faldende rækkefølge – altså 50 48 46 ……6 4 2. Du skal (selvfølgelig) bruge en for-løkke.

Du kan muligvis godt skrive dette lille program uden at skrive pseudokode først – **MEN gør det alligevel.** For herved øver du dig i at tænke over forudsætninger og betingelser i en for-løkke inden du skriver kode, hvilket er super nyttigt.

Nu skal du så i gang med udviklingen af dit trafikstyringsprogram ☺

Du skal bruge din Raspberry Pi til at simulere trafiklys og fodgængerknap mm.





Exercise 3.2 Trafikstyring – iteration 1

Opret et **nyt** RaspberryPi-projekt og kald det ”Trafikstyring Iteration1”. Skriv her i et program, der får trafiklyset til at skifte på korrekt måde mellem ”rødt lys” og ”grønt lys” med passende tidsintervaller (hint 3A).

Husk at skrive pseudokode før du gør i gang !

Når dette virker, har du et program, der opfylder pkt. 1 ovenfor ☺

Du kan derfor gå videre til iteration 2.

Fortsættes på næste side

Exercise 3.3 Trafikstyring – iteration 2

Tilføj et **nyt** RaspberryPi-projekt til din solution. Kald det ”Trafikstyring Iteration2”. Kopier din kode fra `main()` i iteration 1 til `main()` i iteration 2.

Diskutér med dine medstuderende hvorfor du skal lave et nyt projekt i stedet for bare at fortsætte i samme projekt.

Lav de nødvendige ændringer, så trafiklyset ***kun*** skifter til ”rødt lys”, hvis der kommer en fodgænger og trykker på fodgængerknappen) (hint 3B).

Husk pseudokode først !

Når programmet virker, har du opfyldt pkt. 2. Du kan derfor gå videre til iteration 3.

Exercise 3.4 Trafikstyring – iteration 3

Tilføj igen et **nyt** RaspberryPi-projekt og kald det ”Trafikstyring Iteration3”. Kopier din kode fra `main()` i iteration 2 til `main()` i iteration 3.

Tilføj så følgende til dit program: Trafiklyset skal ***også*** skifte til ”rødt lys”, hvis der detekteres et køretøj på sidevejen. Du kan simulere trafiksensoren ved at lyse på fotoresitoren med din mobiltelefon. Når du så sætter en hånd ind mellem mobilen og resistoren, svarer det til at der kommer en bil og ”bryder lysstrålen” (hint 3C).

Begynd med at rette din pseudokode.

Når dette virker, har du et program, der opfylder pkt. 3 ovenfor ☺ Du kan derfor gå videre til iteration 4.

Exercise 3.5 Trafikstyring – iteration 4

Tilføj et igen nyt RaspberryPi-projekt og kald det ”Trafikstyring Iteration4”. Skriv nu et program, der opfylder punkt 4 ovenfor. Brugeren (ham/hende på trafikcentralen) skal - ved tryk på 3 forskellige taster på tastaturet - kunne skifte mellem:

1. dagstilstand (brug koden fra iteration 1)
2. aften/weekendtilstand (brug koden fra iteration 3)
3. afslutte programmet

Mens programmet er i en bestemt tilstand skal brugeren selvfølgelig kunne trykke på en tast og derved ”afbryde” tilstanden og få mulighed for at skifte tilstand eller afslutte programmet (hint 3D).

Nu har du virkelig brug for pseudokode for at kunne overskue flowet i programmet.

Det anbefales, at du skriver et **helt nyt** pseudoprogram (i stedet for at forsøge at rette i det/de ”gamle”) og at du heri **henviser** til iteration 1 og 3 (så du ikke skal gentage den pseudokode, du allerede har skrevet tidligere).

Hint 3A: Kig på det du lavede i exercise 2.1 sidste gang, og husk, at du kan indsætte pauser med funktionen `Wait(int ms)`.

Det vil være en rigtig god ide, hvis du skriver den kode, der får trafiklyset til at skifte mellem rødt og grønt lys, inde i en løkke som denne:

```
do{ …… }while( !kbhit() );
```

Herved kan du bryde ud af løkken ***efter*** hver endt cyclus – altså ved ***slutningen*** af din do/while.

Hint 3B: Efter at det grønne lys er tændt erstatter du `Wait()` med en tom while-løkke. While-løkken skal fungere således:

Sålænge der ***ikke*** er trykket på fodgængerknappen ***og*** der ***ikke*** er trykket på en tast på tastaturet gør ingenting (tom body: { });

Hint 3C: I din while-løkke fra før tilføjer du dette:

og den lysintensitet, som fotoresistoren måler, er større end 50

Hint 3D: Begynd med at skrive et program som, ved valg af dagstilstand hhv. aftentilstand, ***kun*** *udskriver* hvilken tilstand der er valgt (brug en `switch`).

Når dette virker, skal du tilføje en løkke i ***hver*** tilstand, som fungerer således, at brugeren skal trykke på en tast for at afbryde løkken, så programmet går ud af den valgte tilstand og tilbage til valg af tilstand (brug en `do{ …… }while(!kbhit());` i ***hver*** tilstand).

Når dette virker, kan du kopiere koden fra iteration1 til den ene tilstand (dag) og koden fra iteration 3 til den anden tilstand (nat). Nu burde du have et program der (næsten) virker som ønsket.

NB! Når du tester dit program, kan brugeren så bryde ud af aftentilstanden? Hmm…nej sikkert ikke. Tænk over ***hvor*** der testes for, om der er trykket på en tast. Ret det så det brugeren ”altid” kan afbryde en tilstand.