Løsningsforslag til eksamen i ITD13019 Datateknikk 13.05.2020

Oppgaver vil kunne løses på andre måter enn det som er vist i dette løsningsforslaget.

Oppgave 1. (8%)

A.
Kan regnes rett ut med posisjonsprinsippet (vekting).
→ 64 + 16 + 8 + 4 + 1 = **93**

Når vi multipliserer med 2 er det bare å flytte bitmønsteret en posisjon mot høyre.
Start: 0000 1011 = 11 . Ganger med 2 og skal få 22
→ **0001 0110** = 22

B.
59. Vi kan prøve oss fram. Vet at vi må benytte 32 som største tall i 2-tallsystemet
Får så: 59 – 32 = 27 . Der kan vi benytte 16. Og slik fortsetter vi.
Finner løsningen: **0011 1011**

Alternativt fortløpende dele på 2. → **59 : 2 = 29 1 i rest (LSB)**
29: 2 = 14 + 1 (rest)
fortsetter på samme vis (se læreboka)

Dette gir også**: 0011 1011**

C.
A5D4 → $10*16^3 + 5*16^2 + 13*16 + 4 =$ **42452**

Oppgave 2. (6%)

1.
1001 0011 (Må være negativt pga MSB er 1.
Tar 2'er komplement og finner det positive tallet.
0110 1100 + 1 → 0110 1101. Og det er tallet: 109
Tallet må følgelig være: **-109**

2.
14 – 21 utregnes som 14 + (-21)
Må følgelig finne 2'er komplement til 21: → 0001 0101
-21: 1110 1010 +1 = 1110 1011

Summerer:

14:	0000 1110	
+(-21):	1110 1011	
-7:	1111 1001	(-7. Kan sjekkes ved å ta 2'er komplementet så får vi 7)

Oppgave 3. (8%)

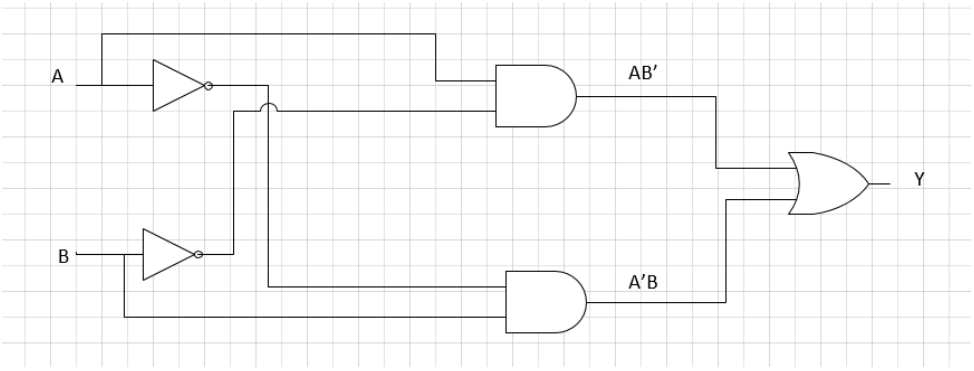
1.
Bruker reglen om at C+ C' = 1
Forenkler og får: $Y = \bar{A}B + A\bar{B}$

2.

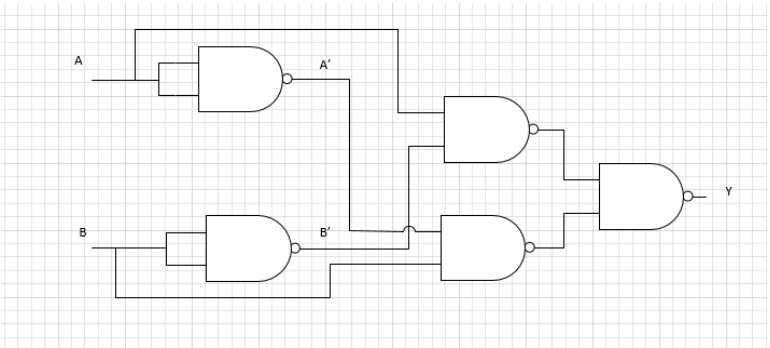
A	B	Y
0	0	0
0	1	1
1	0	1
1	1	0

Dette må være en XOR-krets.

3.
Tegner et krets-skjema. 2 NOT + 2 AND + 1 OR krets.



4.
Kan løses på flere måter. Men rett fram blir det med 5 NAND-porter.
(Kan også løses med 4 NAND.)



5.
NAND kalles en universell port fordi den kan benyttes for å lage**; AND, OR, NOR, NOT**

Oppgave 4. (8%)

1.
Sannhetstabell.

A	B	C	Y
0	0	0	1
0	0	1	0
0	1	0	0
0	1	1	1
1	0	0	1
1	0	1	0
1	1	0	1
1	1	1	1

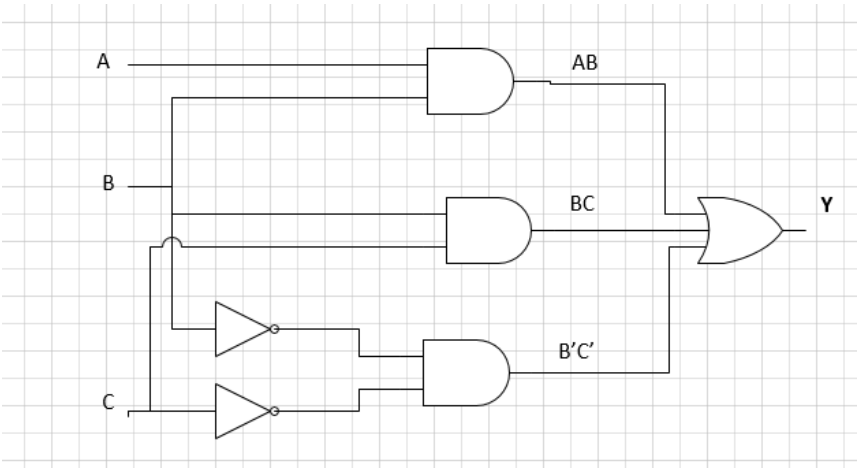
Logisk uttrykk: $Y = \bar{A}\bar{B}\bar{C} + \bar{A}BC + AB\bar{C} + ABC + A\bar{B}\bar{C}$

2.
Forenkler fra K-diagrammet. Lager tre grupper av 2.
Her er 2 svar mulig.

Alt 1. Y = $AB + BC + \bar{B}\bar{C}$

Alt 2. Y= $A\bar{C} + BC + \bar{B}\bar{C}$

Krets-skjema vil bestå av; 3 OG-porter og en ELLER-port som samler utgangene fra disse. Må også ha med invertering der det er nødvendig. Tegner opp for alt 1.

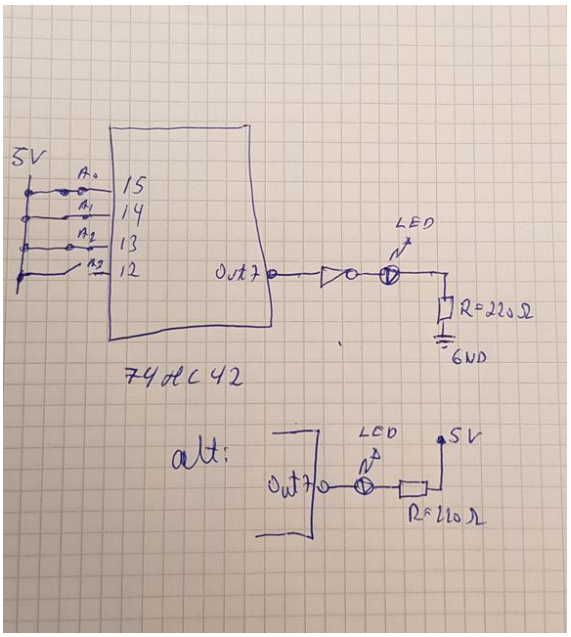


Oppgave 5. (7%)

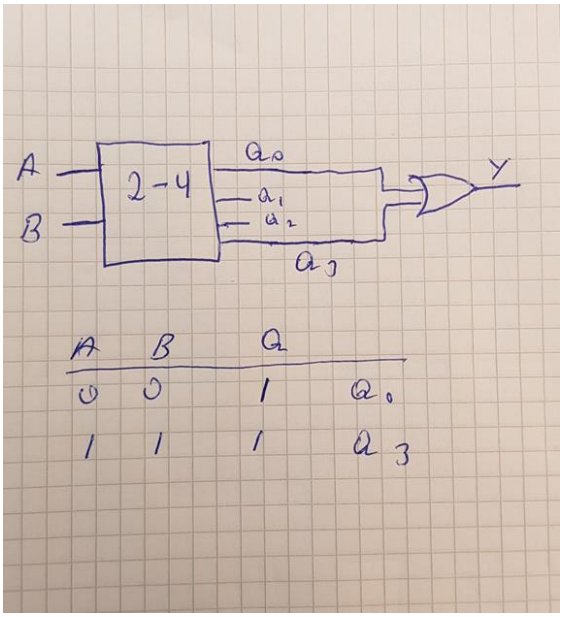
1.
Legger inn 4 brytere på A_0 til A_3. Lar dem bryte 5V inn på A_0, A_1 og A_2.
Tar signalet ut fra out7 (pinne 9.) Dette signalet skal vise at vi har valgt BCD for tallet desimalverdi 7.

2 måter å koble Led-dioden på.
Signalet fra out7 kan vi koble mot en Led med motstand på 220 Ohm.
Rett mot 5V vil det funke.
Kobler vi må jord (GND) må vi benytte en inverter, slik det ble gjort i lab 2.
Vcc er tilførsel-spenning. I dette tilfelle 5Volt. GND er jord eller – (minus-signalet.)

Figuren under viser en skisse.



2.
Dekoderen omformer fra 2 bit (A og B)
Det betyr 4 valgmuligheter inn.
Ut velges 1 av fire utganger Q_0 til Q_3.
Dekoderen omformer fra en binær-kode til en ønsket utgang.
Figuren viser hvordan vi kan lage en XNOR-port.



Oppgave 6. (6%)

1.
Et oscilloskop benyttes for å undersøke tidsvariasjoner av elektriske signaler.
Voltverdien vil være langs Y-aksen. Tiden vil være langs X-aksen.
Signaler fra kretser kan tas inn på 2 eller flere kanaler, slik at vi kan se dem i sammenheng.
Mye brukt for å se på svingninger, pulser.
Vil se ut som et tids-diagram for signalene som kobles inn.

Trigge-funksjonen benytter vi for å «låse» signalene på skjermen.
Hvis vi trigger riktig på et periodisk signal vil det se ut som det står fast på skjermen.
Gjør at vi lett kan avlese variasjoner i volt og tid.
Oscilloskopet har en egen bryter (hjul) som vi kan justere triggepunktet på et signal med.

2.
Ser at J-K-vippa trigger på negativ flanke.
CLK-signalet vil ha en frekvens lik: $F = 1/T = 1/0.02 = 50$ Hz.
Utgangen Q har halvparten av frekvens til CLK. Det vil si 25 Hz.

Oppgave 7. (7%)

1.
En SR-vippe er en vippe som kan sette et utgang høy (Set-funksjon) . S=1
Den kan resette utgangen ved å benytte R=1. (Reset-funksjon).
SR-vippe virker følgelig som et slags minne.
Vi kan sette utgangen høy, og holde den høy inntil vippa resttes.

2.
Det er pull-down motstander.
De sikrer at inngangen til S og R holdes på 0 Volt (GND) når bryteren tilsier det.
Slik vippa er tegnet holdes S på 0 Volt (GND), og R er lagt til en positiv spenning V.
Sikrer at vi ikke får flytende signalinnganger til SR-vippa.

3.
Når vi ønsker å sette vippa Q=1, så legger vi S til høy.
Selv om det er prelling i bryteren så vil utgangen fra SR-vippa låse seg til høy (1)
når S=1, selv om S senere skulle prelle (variere mellom 0 og 1).
Tilsvarende vil vippa resettes når det første høye signalet kommer inn på R.
Utgangen Q vil dermed følge bryteren entydig mellom A og B punktene, og hindre prelling i signalet ettersom den låser seg til første overgang fra 0 til 1.

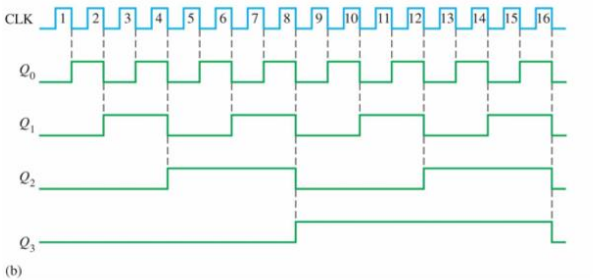
Oppgave 8. (7%)

1.
De trigger på negativ flanke.
Q_0 vil være halvparten ac CP_0 . Det vil si: **2000 Hz**

2.
Q_3 vil være 1/16 del av CP_0. Det vil si: 4000/16 = **250 Hz**

3.
De 4 utgangene Q_0 til Q_3 vil variere med et bitmønster som teller **fra 0 til 15.**
Den har følgelig 16 telle-tilstander.
Kan vises med en figur som den under.

4-bit asynkron binær teller. 16 tilstander.



Teller
0 → 15

4.
Skal den telle 0 til 8 må den resettes når den kommer til 9.
Tallet 9 er binært**: 1001**
Må følgelig koble Q_3 og Q_0 inn på henholdsvis MR1 og MR2.
Dermed vil vippene resettes når de forsøker å telle 9.
Vil dermed starte fra 0 igjen. Teller følgelig 0, 1, 2….8

Oppgave 9. (6%)

1.
En ALU er enheten i en CPU som kan utføre instruksjoner gitt av programkode.
Det kan være addering av tall, sammenligning, manipulering av bit osv…..
Utfører **aritmetiske** og **logiske** operasjoner på en eller to operander.
Består av logiske porter som kan utføre instruksjoner.

2.
CPU'en jobber med data, adresser og instruksjoner.
For å holde operander, adresser m.m trengs en rekke registre nært ALU'en og instruksjonsdekoderen.
CPU'en vil kunne jobbe vesentlig raskere når den har «info» den trenger nært seg.
Denne infoen bør derfor mellomlagres i registre i CPU'en.
Typisk har en CPU fra noen titalls registre til flere hundre.

Oppgave 10. (7%)

1.
Det digitale verdien vil være et heltall.
Desto flere bit vi benytter desto større heltall kan den digitale verdien lagres i,
Heltallet er gitt av: $0 \rightarrow 2^N-1$. (N er antall bit)
F.eks vil vi med 10 bit dekke området 0 til 1023.
Oppløsningen er gitt av $\mathbf{Q = 1/2^N}$

2.
12 bit ADC gir en oppløsning på **1/4096**.
Feilen pga ADC vil være i området; +/- 0.5 *(1/4096) * 100% = **+/- 0.012 %**
Sensorer som vi henter det analoge signalet fra vil normalt ha mye større feil.
Det betyr at en 12 bit ADC ofte er mer en god nok i slike systemer hvor en ikke har sensorer med ekstrem nøyaktighet.

3.
Den skal gi ut : 2.9/5 * 4096 = **2375**
Utregningen gir 2375.68. Men bitmønsteret må være et heltall. Fjerner desimaler.

Oppgave 11. (14%)

```
/*  Oppgave 11. Eksamen 13.mai 2020
    Fil: oppgave11_eksamen_mai2020.io   RR mai 2020    */

// lager noen konstanter og globale variabler
const int bryterA = 11;
const int bryterB = 12;
const int bryterC = 13;
bool startA = false;
bool startB = false;
bool startC = false;

void setup()
{
  // initialiserer digitale utganger. 9 Led-lys
 for ( int pin=2; pin < 11; pin++)
 {
  pinMode(pin,  OUTPUT );
  digitalWrite( pin , LOW );   // slår av alle Led ved start
 }

 // initierer innganger.  Inn er default så dette er ikke nødvendig.
  pinMode( bryterA, INPUT );  // tilkobling for bryter A.
  pinMode( bryterB, INPUT );
  pinMode( bryterC, INPUT );
}


void loop()
{
 // sjekker om systemet skal stoppes. Sjekker Bryter C.
 if ( startC || digitalRead(bryterC) )
 {
   // slår av alle lys
   for ( int pin=2; pin < 11; pin++)
   {
      digitalWrite( pin , LOW );
   }
   startC = false;
 }

  // Leser av bryterA  for sjekk om bruker ønsker syklus A.
  if ( startA || digitalRead(bryterA) )         // Leser av om bryetr A er trykket
  {
    startA = true;
```

```
    while ( startA )
    {
     // Utfører denne syklus A så lenge det er ønsket
     for ( int pin=2; pin < 11; pin++)
     {
      digitalWrite( pin , HIGH );
      delay(500);            // venter 0.5 sek
      digitalWrite( pin , LOW );
     }

     // sjekker bryter B og C
     startB = digitalRead(bryterB);
     startC = digitalRead(bryterC);
     if ( (startB || startC) == true )
       startA = false;
    }                        // end while
  }                          // end if


  // Leser av bryterB  for sjekk om bruker ønsker syklus B.
  if ( startB || digitalRead(bryterB) )        // Leser av om bryetr B er trykket
  {
     startB = true;
     while ( startB )
     {
     // Utfører denne syklus så lenge det er ønsket
      for ( int pin=2; pin < 5; pin++)
      {
       // Setter rød, grønn og blå Led høy og lav i riktig rekkefølge
       digitalWrite( pin , HIGH );              // slår Led på
       digitalWrite( pin + 3 , HIGH );
       digitalWrite( pin + 6 , HIGH );
       delay(2000);                              // venter 2 sek
       digitalWrite( pin , LOW );               // slår Led av
       digitalWrite( pin + 3 , LOW );
       digitalWrite( pin + 6 , LOW );
      }
       // sjekker bryter A og C
       startA = digitalRead(bryterA);
       startC = digitalRead(bryterC);
       if ( startA || startC == true )
           startB = false;
     }                                 // end while
    }                                  // end if
}                                      // end loop
```

Oppgave 12. (7%)
A.

```
/*
   Oppgave 12. Eksamen mai 2020
   Fil: oppgave12_eksamen_mai2020.ino    */

const int tempSensor = A0;      // Analog inngang
const int trykkSensor = A1;     // Analog inngang
const int vindSensor = A2;      // Analog inngang


void setup()
{
 Serial.begin(9600);    // Initierer seriellporten for kommunikasjon med monitor.
}


void loop()
{
 int sensorVerdi = 0;
 float tempVerdi = 0.0;
 float trykkVerdi = 0.0;
 float vindVerdi = 0.0;

 sensorVerdi = analogRead(tempSensor);
 Serial.print("Sensor Verdi A0 = ");              // Sensorverdien i heltall.
 Serial.println(sensorVerdi);

 tempVerdi = (sensorVerdi/1024.0)*180.0 - 100.0;  // Beregner verdi for temperatur
 Serial.print("Temperaturen er = ");
 Serial.println(tempVerdi,2);
 Serial.println("*---------------------------*");

 sensorVerdi = analogRead(trykkSensor);
 Serial.print("Sensor Verdi A1 = ");              // Sensorverdien i heltall.
 Serial.println(sensorVerdi);

 trykkVerdi = (sensorVerdi/1024.0)*400.0 + 900.0;          // Beregner verdi for trykk
 Serial.print("Luft-trykket = ");
 Serial.println(trykkVerdi,2);
 Serial.println("*---------------------------*");

 sensorVerdi = analogRead(vindSensor);
 Serial.print("Sensor Verdi A2 = ");              // Sensorverdien i heltall.
 Serial.println(sensorVerdi);
```

```
 vindVerdi = (sensorVerdi/1024.0)*50.0;      // Beregner verdi for vindhastighet
 Serial.print("Vindhastigheten = ");
 Serial.println(vindVerdi,2);
 Serial.println("*----------------------------*");

 Serial.println("\n\n\n");
 delay(5000);                //    venter 5 sek
}
```

B.

Vi kan legge loggingen av sensorene i en funksjon som vi så setter opp som et **periodisk program**.
Vi kan benytte en Timer-funksjon som går med et fast tids-intervall. F.eks hvert 5 sekund.
Vi benytter en Timer med initiering som setter opp det periodiske programmet.
På Arduino Uno kan inntil 3 Timere benyttes.
Tar kun med initieringskode som kan benyttes. Ble testet i en lab-oppgave.
Ikke krav at kode skal tas med.

```
#include <TimerOne.h>       // Må være med
Timer1.initialize(tid)      //  angir periodetiden i mikrosekunder
Timer1.attachInterrupt(timer_rutine)        // angir navnet på funksjonen
                                            //   som skal kjøres periodisk
```