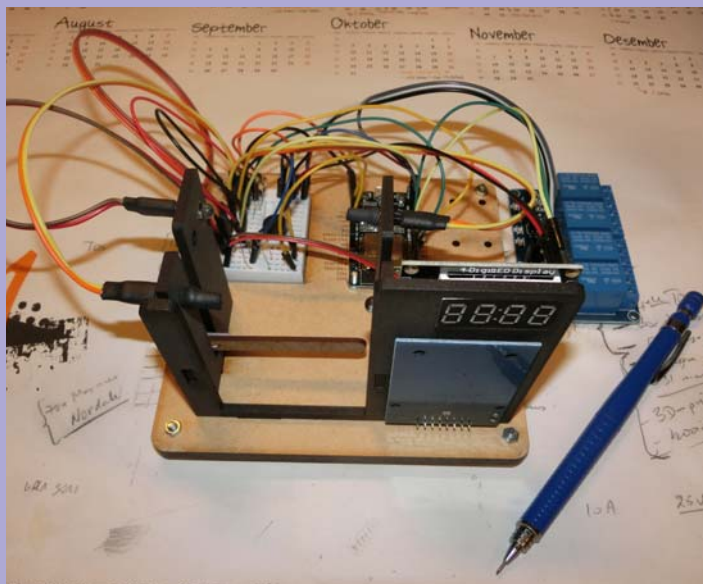


*Nils Kr. Rossing og
Kåre-Benjamin H. Rørvik*
**ESP32 – Grunnkurs
programmering (YFL)**



NTNU



Trondheim

Institutt for fysikk

Skolelaboratoriet

for matematikk, naturfag
og teknologi

Institutt for
elektroniske systemer

August 2020



ESP32

Grunnkurs programmering (YFL)

Nils Kr. Rossing, Skolelaboratoriet og
Kåre-Benjamin H. Rørvik, Institutt for elektroniske systemer

ESP32 – Grunnkurs programmering (YFL)

Trondheim 2020

Layout og redigering: Nils Kr. Rossing, Skolelaboratoriet ved NTNU

Trykk: NTNU Grafisk senter

Tekst og bilder: Nils Kr. Rossing, Skolelaboratoriet ved NTNU

Forsidebilde: Nils Kr. Rossing

Korrektur og uttesting: Kåre-Benjamin H. Rørvik, NTNU

Kursholdere: Nils Kr. Rossing, Skolelaboratoriet

Faglige spørsmål rettes til:

Skolelaboratoriet for matematikk, naturfag og teknologi

Institutt for fysikk

v/ Nils Kr. Rossing nils.rossing@ntnu.no

Skolelaboratoriet ved NTNU

Realfagbygget,

Høgskoleringen 5,

7491 Trondheim

Telefon: 73 55 11 91

<https://www.ntnu.no/skolelab/>

Rev 1.4 – 18.08.24

Undervisningsopplegget er utviklet i samarbeid med Institutt for elektroniske systemer ved Arne Midjo.



Forord

Heftet er et kurshefte laget som en videreføring av kursmodulen: *Arduino – grunnkurs programmering*, og er utarbeidet spesielt med tanke på å brukes overfor yrkesfagelever. Øvingsopplegget som kurset tilbyr er rettet mot problemstillinger som kan være relevant for yrkesfag Elektro og Bygg og anlegg. Likevel ønsker vi å arbeide på grunnplanet der elektronikken ikke er mer kompleks enn at det er mulig å forstå de ulike elementene som blir brukt.

Vi har også valgt å ta i bruk ESP32 som inkluderer WiFi og Bluetooth, og gir mikrokontrolleren mulighet til å fungere som en sensornode som kan kobles opp mot Internett og overføre data begge veier. Kretsen er derfor spesielt egnet til *Internett of Things* (IoT). Samtidig kan den programmeres omtrent som en Arduino ved hjelp av det samme programmeringsverktøyet hvilket gjør overgangen lettere for Arduino-brukere. Vi kommer ikke til å benytte WiFi eller Bluetooth i denne sammenheng, men legger opp til at det blir hovedtemaet i en påfølgende modul.

Utviklingen er delvis finansiert av Udir og Trøndelag fylkeskommune, og er en “spinn off” av et mer omfattende videreutdanningskurs rettet mot yrkesfaglærere både innen Elektro, Bygg og anlegg og Teknisk Industriell Produksjon (TIP) som vil bli gjennomført første gang våren 2021 ved NTNU i Trondheim.

En spesiell takk til Institutt for elektroniske systemer og Arne Midjo som har fulgt prosjektet gjennom hele sommeren og kommet med verdifulle innspill. Videre at instituttet har stilt midler til rådighet slik at Kåre-Benjamin har kunnet gjennomgå undervisningsopplegget og teste ut programvaren. Instituttet har også bidratt med studentassistenter for å lage byggesettene samt stille som veiledere under kurset.

Skolelaboratoriet ved NTNU

August 2020

Nils Kr. Rossing

Kåre-Benjamin H. Rørvik





Innhold

1	Innledning	11
1.1	Fagfornyelsen 2020	11
1.2	Foreløpig plan for kursdagen: ESP32 – Grunnkurs programmering (YFL)	12
2	Bakgrunnen	13
2.1	Mikroprosessorer og mikrokontrollere	13
2.2	ESP 32 - WROOM - 32 med 38 pinner	15
2.3	ESP - WROOM - 32 med 30 pinner	20
3	Programmering	23
3.1	Forskjeller og likheter hos Arduino og ESP32	23
3.2	Installasjon av programvare	24
3.2.1	Arduino programeditor, IDE	24
3.3	Installasjon av ekstra pakke for programmering av ESP32	26
3.4	Programstruktur	30
3.5	Viktige kommandoer	30
3.5.1	Generelle kommandoer	30
3.6	Bruk av I2C og biblioteker	36
3.6.1	Installasjon av pakkede biblioteker generelt	37
3.6.2	Bibliotek for bruk av I ² C	38
4	Oppgavesamling	39
4.1	Kursets prosjektoppgave – Oppdrag	39
4.1.1	Oppdeling i delprosjekter	39
4.2	Programmets grunnstruktur	41
4.3	Oppbygging av målejigg	42
4.3.1	Mekanisk byggeveiledning	42
4.3.2	Elektrisk byggeveiledning	44
4.4	Oppdrag 1 – Oppbygging av en enkel optisk portal	46
4.4.1	Elektronisk virkemåte for en enkel optisk portal	46
4.4.2	Spenningsdeleren	46
4.4.3	Oppkobling	47
4.4.4	Programmering	48
4.5	Oppdrag 2 – Skriv til displayet	50
4.5.1	Oppkobling	50
4.5.2	Programmering	52
4.6	Oppdrag 3 – Tell antall passeringer	55
4.6.1	Oppkobling	55



4.6.2	Programmering	55
4.7	Oppdrag 4 – Montering og styring av servo	55
4.7.1	Servoer	56
4.7.2	Oppkobling	57
4.7.3	Programmering	58
4.8	Oppdrag 5 – Kombiner den optiske portalen, telling av passeringer og servoen	59
4.8.1	Oppkobling	59
4.8.2	Programmering	59
4.9	Oppdrag 6 – Styring av lys ved hjelp av rele	59
4.9.1	Releets funksjon	60
4.9.2	Oppkobling	61
4.9.3	Programmet	62
4.10	Oppdrag 7 – Retningssensitiv optisk portal	62
4.10.1	Oppkobling	63
4.10.2	Programmet	64
4.11	Oppdrag 8 – Suppler avansert optisk sluse med bom	65
4.11.1	Oppkobling	65
4.11.2	Programmet	65
4.12	Oppdrag 9 – Suppler avansert optisk sluse og bom med visning av antall	65
4.12.1	Oppkobling	65
4.12.2	Programmet	66
4.13	Oppdrag 10 – Suppler avansert optisk sluse, bom og visning av antall med et rele	66
4.13.1	Oppkobling	66
4.13.2	Programmet	66
4.14	Oppdrag 11 – Program som leser RFID-kort og viser ID'en i monitoren	66
4.14.1	RFID-brikken	67
4.14.2	Oppkobling	68
4.14.3	Programmet	69
4.15	Program som gir tilgang for akkrediterte og åpner bommen	73
4.15.1	Oppkobling	73
4.15.2	Programmet	73
4.15.3	Tilleggsoppgaver	74
4.16	Oppdrag 13 – Endelig sammenstilling av hele systemet	74
4.16.1	Oppkobling	75
4.16.2	Programmet	75
5	Noen aktuelle sensorer og aktuatorer	78
5.1	Spenningsdeleren	78



5.2	Lysfølsom sensor – LDR	80
5.2.1	Fotomotstand (LDR - Light Dependent Resistor) 81	
5.3	Trådløs identifikasjon – RFID	83
5.4	Lysdioder	87
5.4.1	Ordinære lysdioder	87
5.4.2	Flerfargede lysdioder – RGB-dioder	90
5.4.3	Sykliske RGB-dioder	90
5.4.4	Digitalt styrte RGB-dioder ved WS2811	91
5.5	Numeriske 7 segment display – TM1637	93
5.5.1	Kort omtale:	93
5.5.2	Programmering	94
5.6	Servomotorer	96
5.6.1	Virkemåte	96
5.6.2	Viktige parametere for servoer	97
5.6.3	Programmering av servoer	98
5.7	Releer	100
5.7.1	Releets funksjon	100
5.7.2	Transistordriveren	100
5.7.3	Reed releet	101
5.7.4	FET-transistorer for styring av store likestrømmer.	101
5.7.5	Releer med innebygget driver	102
6	Referanser	103
Vedlegg A	Komponentliste	104
Vedlegg B	Programmer løsningsforslag	105
B.1	Oppdrag 1 – Enkel optisk portal	105
B.2	Oppdrag 2 – Skriv til display	106
B.3	Oppdrag 3 – Tell antall passeringer og vis på display	107
B.4	Oppdrag 4 – Sett opp servoen slik at den åpner og lukker bommen	108
B.5	Oppdrag 5 – Sammenstill enkel optisk portal, teller og åpning av bommen ...	109
B.6	Oppdrag 6 – Styring av rele	111
B.7	Oppdrag 7 – Avansert optisk port, registrering av inngang og utgang	113
B.8	Oppdrag 8 – Avansert optisk portal med servo bom	116
B.9	Oppdrag 9 – Avansert optisk portal med bom og display	119
B.10	Oppdrag 10 – Avansert optisk portal med bom, display og rele	123
B.11	Oppdrag 11 – Lesing av ID-kort informasjon	127
B.12	Oppdrag 12 – Lesing av ID-kort informasjon og åpning av bom	128
B.13	Oppdrag 13 – Endelig sammenstilling – Løsning av det totale oppdraget	130



Vedlegg C	Maler for laserkutting	138
C.1	Målejigg	138
Vedlegg D	Administrative tiltak	142
D.1	Oversikt over arbeidsoppgaver byggesett	142
D.2	Legges ut på hjemmesiden til Skolelaboratoriet	143



1 Innledning

Kurset er en påbygningningsmodul til kurset: *Arduino – Grunnkurs programmering* og har som målsetning å gi kursdeltagerne bredere programmeringserfaring. Videre viser kurset hvordan et system kan brytes ned til enklere enheter som så kan settes sammen til et funksjonelt hele. Tanken er at deltagerne skal arbeide selvstendig med arbeidsheftet under veiledning og nå delmål etter delmål samtidig som de under veis kombinerer enkeltdeler til et større system. Dermed blir det ikke så viktig at de når alle delmålene, men at de heller kan få ro til fordypning innen hvert delmål. Hovedmålet eller den endelige prosjektideen kan likevel gi arbeidet med delmålene en retning og være en viktig motivasjon. At de i dette tilfellet arbeider med en målejigg som gjensker inngangskontroll til et rom gir arbeidet en viss autentisitet samtidig som det peker mot det endelige målet med arbeidet.

Videre vil kurset være en introduksjon til kursets tredje kursdag der deltakerne lærer å utnytte ESP32s egenskaper som trådløs sensornode koblet opp i et trådløst nettverk, f.eks. med tanke på fjernstyring og overvåking via Internett.

1.1 Fagfornyelsen 2020

Programmering er relevant i mange fag. Vi har hovedsakelig fokusert på grunnleggende programmering av Arduino for de som ikke har programmert Arduino tidligere, og som opp til et visst punkt er uavhengig av hvilken yrkesfaglig retning deltagerne kommer fra.

En kan tenke seg at det er mulig å delta fra dag 2 for den som behersker grunnferdighetene. Vi har i første rekke sett på Elektrofag.

Kursets innhold bygger på følgende punkter i LK20:

Skoleåret 2020/21 er det Vg1 som har læreplaner etter Fagfornyelsen (LK20).

Læreplanen for Vg1 Elektrofag fremhever programmering både i fagets kjerneelementer og i flere kompetansemål. Fra læreplanens kjerneelementer kan vi lese at: *"Kjerneelementet komponenter, kretser og utstyr handler om å regne på og utføre målinger på elektriske og elektroniske kretser og å kunne anvende utstyr og komponenter i helhetlige systemer. Det handler også om å kunne programmere utstyr og komponenter."*

Eksempler på kompetansemål fra læreplan på Vg1 i emnet Elektroniske kretser og nettverk:

Elektroniske kretser og nettverk

Elevene skal kunne ...

- bygge og programmere et selvvalgt produkt som består av mikrokontroller, analoge kretser, relevante sensorer og aktuatorer for å oppnå ønsket virkemåte
- koble sammen ulike datateknologiske enheter til et system, konfigurere aktuelle komponenter ved hjelp av programvare og opprette kommunikasjon mellom enhetene for å oppnå ønsket virkemåte
- montere og konfigurere et mindre datanettverk med internettilkobling, utføre relevante målinger og gjøre rede for enkle tiltak for å sikre nettverket



- velge og bruke egnede instrumenter og programvare for å utføre målinger og feilsøking, og vurdere måleresultatet opp mot forventede verdier

1.2 Foreløpig plan for kursdagen: ESP32 – Grunnkurs programmering (YFL)

Program kursdag – Oppmøte

09.00 – 09.15	Velkommen, praktisk og forventninger
09:15 – 10:15	Økt 1 Introduksjon og oppstart prosjekt: <i>Lag adgangskontroll</i>
10:15 – 10:30	Kaffepause
10:30 – 12:00	Økt 2 <i>Forsett med prosjekt: Lag adgangskontroll</i> Korte avbrekk med viktig kunnskap
12:00 – 12:30	Lunsj
12:30 – 13:30	Økt 3 <i>Forsett med prosjekt: Lag adgangskontroll</i> Korte avbrekk med viktig kunnskap
13:30 – 13:45	Kaffepause
14:00 – 15:30	Økt 4 <i>Forsett og avslutt prosjekt: Lag adgangskontroll</i> Korte avbrekk med viktig kunnskap
15:30 – 16:00	Oppsummering, refleksjon og diskusjon om hvordan ta erfaringene inn i klasserommet

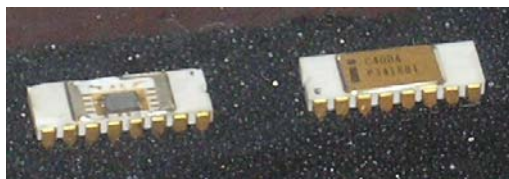
2 Bakgrunnen

2.1 Mikroprosessorer og mikrokontrollere

Vi er alle klar over at vi omgir oss med store mengder elektronikk både i skolen, på jobben, i bilen, hjemme, ja, overalt hvor vi ferdes. De fleste av oss bærer på svært avansert elektronikk og kommunikasjonsutstyr. Tenker vi etter så har de fleste med seg en kalkulator som er en liten datamaskin. Mange vil ha en smarttelefon som ikke bare er ett kommunikasjonsmiddel for sending av tekstmeldinger og telefonsamtaler. De fleste mobiltelefoner tilbyr stadig flere tilleggsfunksjoner som f.eks. musikkanlegg, navigasjonsutstyr, kart, kamera for opptak av bilder og film, kalendere og arkivsystemer for å holde orden på hverdagen, spill og underholdning, bibliotek med både romaner og lærebøker, og utallige andre funksjoner. I tillegg går flere av oss rundt med bærbar PC-er eller nettbrett for større oppgaver, gjerne med nett-tilgang.



Figur 2.1 HTC Smartphone



Figur 2.2 Intel mikroprosessor Intel 4004 regnes for den første mikroprosessen.

Lansert 15. nov. 1971

Sentral i alt slikt utstyr er mikroprosessen som er en miniaturisert datamaskin som satt på spissen, stort sett bare kan addere, sammenligne, flytte på og lagre binære tall. Dessuten kan den “Den lille multiplikasjonstabell” for binære tall som er $0 \times 0 = 0$ og $1 \times 0 = 0$ og $1 \times 1 = 1$. I tillegg må den kommunisere med omverdenen ved å omdanne fysiske størrelser, som lyd, lys, temperatur, radiosignaler, gravitasjon osv. til binære tall.

Selv om mikroprosessorer i dag er blitt langt mer avanserte og raskere enn den første mikroprosessen som Intel lanserte i 1971, så er grunnfunksjonen omtrent den samme.

En mikroprosessor er derfor en temmelig enfoldig og dum elektronisk “duppe dings”, men den har noen svært nyttige egenskaper:

Den arbeider svært fort. I løpet av et sekund kan den utføre mange hundre millioner, ja noen ganger milliarder av små enkle oppgaver i sekundet. Setter man sammen mange nok små enkle oppgaver, kan man gjøre store komplekse oppgaver i løpet av en brøkdel av et sekund.

Den glemmer aldri. Når noe er lagret i hukommelsen til en mikroprosessor så går det ikke “i glemmeboka”. Så lenge den virker som den skal så huskes det til “evig tid”. Selv om “strømmen går” husker den det som er skrevet inn i hukommelsen.

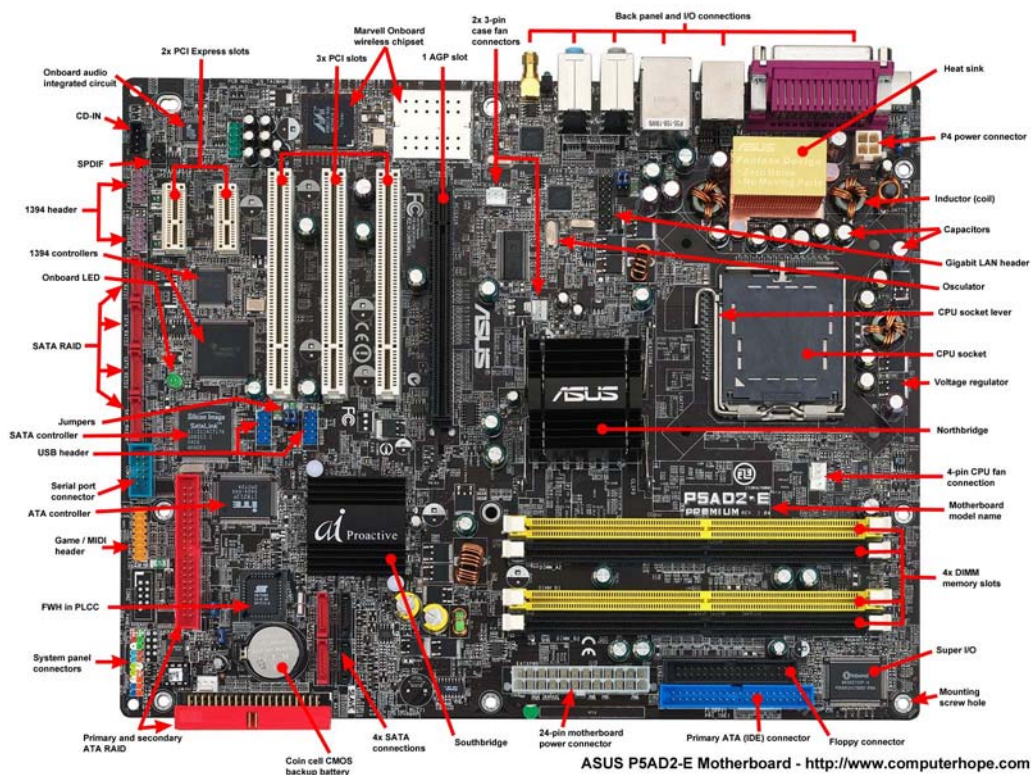
Den går ikke lei. Dersom den er instruert til å utføre en regneoperasjon riktig, så gjør den det



riktig hver gang uten å bomme en eneste gang. Dersom noe går galt så skyldes det som regel at de som har laget mikroprosessen eller oppskriften (programmet) den følger, har gjort feil eller ikke tenkt på alt. Dessuten er en slik gjenstand sårbar slik at den kan bli ødelagt av feilaktig eller uforsiktig bruk.

Ofte benyttes begrepene *mikroprosessor* og *mikrokontroller*. I dette heftet skal vi bruke mikrokontrollere. Men hva er egentlig forskjellen?

En mikroprosessor er en meget generell regne- og datamanipuleringsenhet som oftest er plasseres i et miljø med egne kretser for lagring av data og program på utsiden av mikroprosessen. Likeså foregår all spesialisert kommunikasjon med omverdenen med egne spesialiserte kretser. Åpner du en datamaskin vil du gjerne se et stort kretskort (moderkort) som inneholder mange ulike kretser (se figuren under).

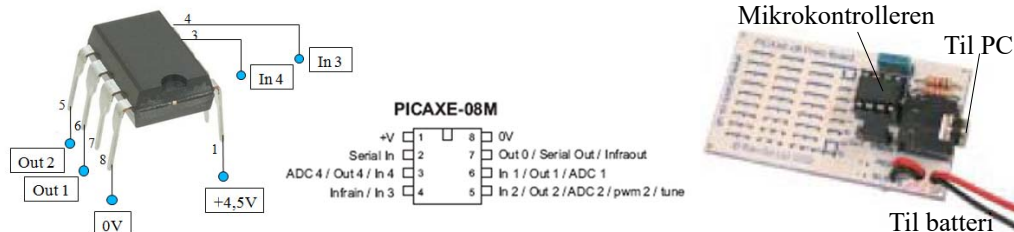


En mikrokontroller inneholder regne- og datamanipuleringsenheten, lager for program og data og det som skal til for å kommunisere med omverdenen, enten det er analoge spenninger eller digitale inn- og utganger, så finnes alt dette inne i den integrerte kretsen. Mikrokontrolleren er med andre ord en “komplett” datamaskin inkludert i én integrert krets. Prisen en må betale er at den ikke er så kraftig og generell som en mikroprosessor, men som oftest er den kraftig nok til vårt bruk. Den mangler dessuten muligheter for direkte å koble til PC-tastatur og skjerm. Skal vi kommunisere med en mikrokontroller er vi derfor som oftest avheng av å bruke en PC med tilpasset programvare for å programmere den. Når den først er programmert kan den frigjøres fra

tastatur, mus, skjerm og PC.

Mens mikroprosessorer gjerne står i datamaskiner, finner vi mikrokontrollere i alt mulig annet utstyr som vaskemaskiner, biler, overvåkning og styringsystemer og annen småelektronikk. Det er innen dette markedet at Atmel Norge¹ gjorde seg bemerket.

Det finnes en rekke forskjellige familier med mikrokontrollere. Som et eksempel på hvor enkel en mikrokontroller kan være er det på bilde under vist en PIC-kontroller. Så og si en liten datamaskin med 8 bein (til venstre).



Denne vesle kretsen har i alt 5 porter (bein) som kan programmeres som digitale inn- eller utganger, med og uten puls-bredde-modulasjon, analoge innganger, eller innganger som kan fungere som mottakere for IR-kommunikasjon eller som berøringsbrytere. En av utgangene er forberedt for å levere toner fra skalaen. Til høyre på figuren over er vist kretsen montert på et lite kretskort, med tilkoblingsplugg for batteri og programmering fra PC.

På 1980-tallet oppdaget man at det var mer effektivt å lage mikrokontrollere som hadde få og enkle kommandoer som kunne utføres svært raskt og effektivt, enn å ha mikrokontrollere med mange spesialiserte og ofte kompliserte kommandoer. Denne teknologien kalte man RISK (Reduce Instruction Set Computer) i motsetning til CISK (Complex Instruction Set Computer). Atmels mikrokontrollerfamilier som brukes i det fleste Arduino-kort er stort sett av RISK type.

2.2 ESP 32 - WROOM - 32 med 38 pinner

ESP32 er et kontrollerkort bygget opp omkring en kombinert WiFi og Bluetooth enhet som benytter 2,4 GHz båndet som kommunikasjonskanal.

Kortet er bygget opp omkring to prosessorer, en ESP-WROOM-32 og en CP2102N, sistnevnte tar seg av kommunikasjon med PC'en via USB-tilkoblingen. Lagerkapasiteten er som følgende: 448 kB ROM (Read Only Memory), 520 kB SRAM (Static Random Access Memory), i tillegg til 16 kB SRAM i en RTC (Real Time Clock).

Senderen har en maksimal effekt på +12 dBm (8 mW) og mottakeren en følsomhet på -94 dBm. Kortet har følgende porter:

- **Analoge og Digitale I/O-porter**

Kortet har 34 digitale² inn/utporter (I/O-porter) som kan programmeres til enten å være inn-

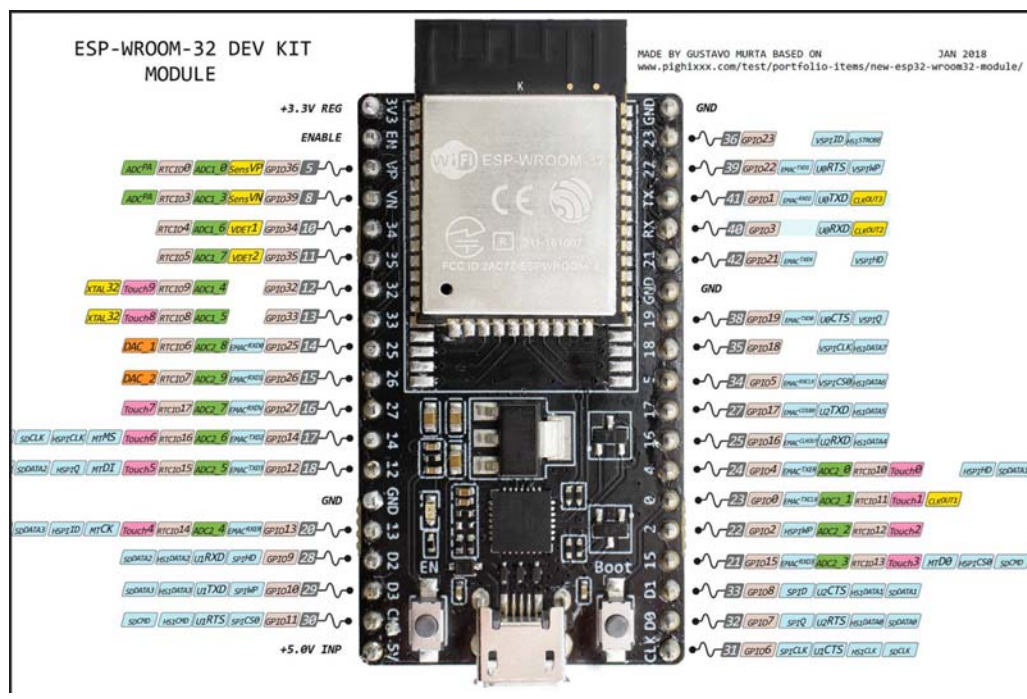
1. I dag er Atmel oppkjøpt av firmaet Microchip Technology, men har fortsatt en stor utviklingsavdeling i Norge.
2. Dette kan variere noe fra versjon til versjon



eller utganger. 12 bit ADC (Analog til Digital Converter) med opp til 18 kanaler. I tillegg har den 2 x 8 bit DAC (Digital til Analog Converter). 10 av inngangen kan fungere som touch sensor. I tillegg har den mulighet til å styre motorer med PWM og opp til 16 PWM for styring av LED.

- **Serie kommunikasjon:** Kortet har 4xSPI kanaler. Dessuten har det 2 x I²C, 2 x I²S³ og 3 x UART⁴. Den har også mulighet for å kommunisere med IR (Infrared) (Tx/Rx).
- **USB-kontakt** for direkte tilkobling av PC, for programmering av kortet. Under programmeringen tilføres kortet spenning fra USB-kontakten. Dersom denne belastes med mer enn 500 mA vil strømforsyningen bli brutt inntil strømtrekket reduseres under denne grensen. USB C er designet for høyere strømtrekk, men det er PC'en som til syvende og sist bestemmer hvor mye den kan levere.
- **Strømtilførselen**⁵ skjer via USB-kontakten som leverer 5V og reguleres ned til 3,3 V. Prosessorene har arbeidsspenning fra 2,2 – 3,3 V. Typisk strømtrekk er 80mA, maksimalt 500 mA. Kretsen har imidlertid flere nivåer av dvale tilstand fra “lett sovende” (0,8 mA) til “vinterdvale” (5 µA) hvor bare “real time” klokka (RTC) er aktiv.

Figuren under viser pinningen for ESP-Wroom-32 DEV KIT med 38 pinner. Vi legger merke til at hver pinne har flere funksjoner.



3. I²S er en enkel buss for overføring av digitale audio mellom kretser, med en klokkefrekvens på 44.1 kHz × 16 × 2 = 1.4112 Mbit/s som egner seg for overføring av 2 x 16 bits kanaler. <https://en.wikipedia.org/wiki/I2S>
4. UART – Universal Asynchronous Receiver/Transmitter, linje for toveis overføring av data.
5. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf

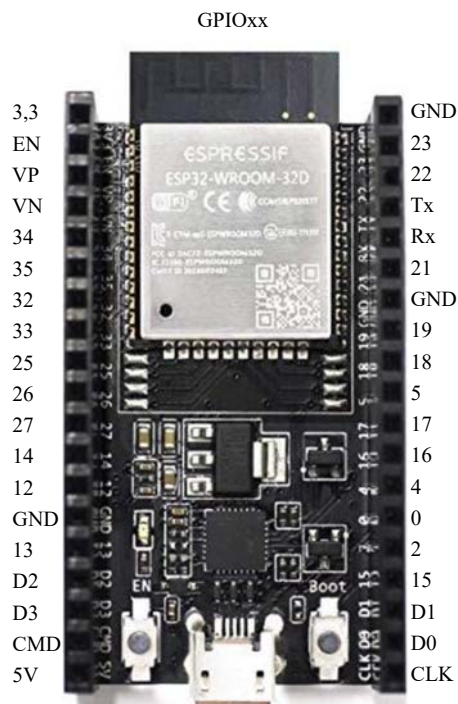


- *Enable-knappen*
Denne knappen resetter programmet. Skjønt det er ikke alltid erfaringen.
- *Boot-knappen*
NB! Noen varianter av ESP32 krever at man bruker denne ved opplasting av programmet. Dette gjelder den varianten vi bruker i dette kurset. Andre varianter har innebygget elektromekanikk som laster opp programmet uten bruk av Boot. Ved å trykke Boot-knappen sammen med EN-knappen går kretsen inn i en tilstand der det er mulig å installere ny firmware via serieporten.

Innbygde sensorer og ADC og DAC⁶

Under brukes betegnelsen GPIOxx, tallet (xx) angir nummeret som er påført kretsen.

- *Hall sensor*
En hall magnetfelt sensor er integrert på kortet og kan tilsluttes en av de analoge inngangene.
- *Temperatursensor*
- *Kapasitive berøringssensorer*
10 av inngangene kan fungere som kapasitive berøringssensorer. Dette er Touch 0–9: GPIO4, 0, 2, 15, 13, 12, 14, 27, 33 og 32.
- *ADC (Analog til digital omvandler)*
Følgende innganger kan knyttes til de 18 12 bits ADC-inngangene, disse er ordnet i to grupper
ADC1 0, 3–7: GPIO 36, 39, 32, 33, 34 og 35.
ADC2 0–9: GPIO 4, 0, 2, 15, 13, 12, 14, 27, 25 og 26. I tillegg har vi to som går under betegnelsen ADC^{PA}: GPIO36 og 39.
- *DAC (Digital til analog omvandler)*
ESP32 har to 8 bits DA-omvandlerne disse betegnes DAC_1 og DAC_2 er tilkoblet GPIO35 og 26.
- *IR-kontroll*
I alt 8 kanaler kan betjene ulike IR-protokoller.

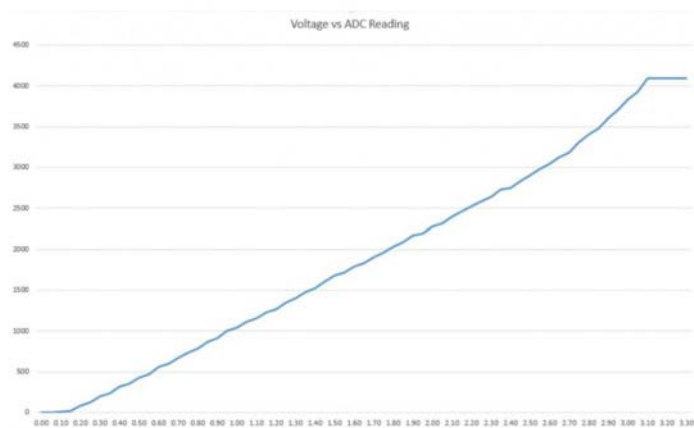


6. https://www.espressif.com/sites/default/files/documentation/esp32_datasheet_en.pdf



Merk:

1. Alle GPIO som er skravert i tabellen finnes ikke tilgjengelig på pinner
2. Alle A2-0-9 kan ikke brukes sammen med WiFi
3. GPIO-0 har intern pullup når den brukes som digital inngang
4. Alle GPIO kan konfigureres for interrupt
5. Alle GPIO kan konfigureres som PWM (Pulsbredde modulering)
6. ADC (Analog til Digital Converter) er ikke lineær i endene som vist i figuren under



Hvordan lese og skrive til de ulike pinnene koblet til funksjon

Nå har vi funnet ut hvilke pinner og GPIO nummer som kan brukes til digitale, analoge og touch formål. Så er spørsmålet hvordan vi når disse pinnene med de ulike funksjonene fra programmet? Det er `pinMode()`-funksjonen eller funksjonen vi bruker for å avlese eller skrive til GPIO-porten som definerer dens funksjon:

Digital (inngang)

Vi kan lese verdien på den digitale inngangen slik:

```
int verdi                // Kan holde heltalsverdier med fortegn
pinMode(16, INPUT);      // Definerer GPIO 16 som en digital inngang
verdi = digitalRead(16); // Leser GPIO 16 som en digital inngang
```

Deklarasjonene sår gjerne i starten av programmet, `pinMode()` står vanligvis i `setup()`-funksjonen definerer GPIO 16 som en digital utgang og avlesning av porter er gjerne plassert i `Loop()`-funksjonen.



Digital (utgang)

Tilsvarende kan vi definere en GPIO-port som en utgang slik:

```
pinMode(17, OUTPUT); // Definerer GPIO 17 som en digital utgang
digitalWrite(17, HIGH); // Gir GPIO-port 17 verdien høy
```

Som vi ser så forholder vi oss til GPIO-portnumrene.

Lese en analog inngang

En analog port trengs ikke defineres, heller ikke trenger vi å spesifisere om det gjelder ADC1 eller ADC2 vi ønsker å bruke. Det er kun GPIO-nummeret som gjelder. Vi må imidlertid bruk en GPIO som kan konfigureres som en analog inngang:

```
int verdi; // Vil ha verdier 0 - 4096, 12 bit
verdi = analogRead(34); // Hvor GPIO-34 angir ADC1_CH6)
```

Touch – funksjon (inngang)

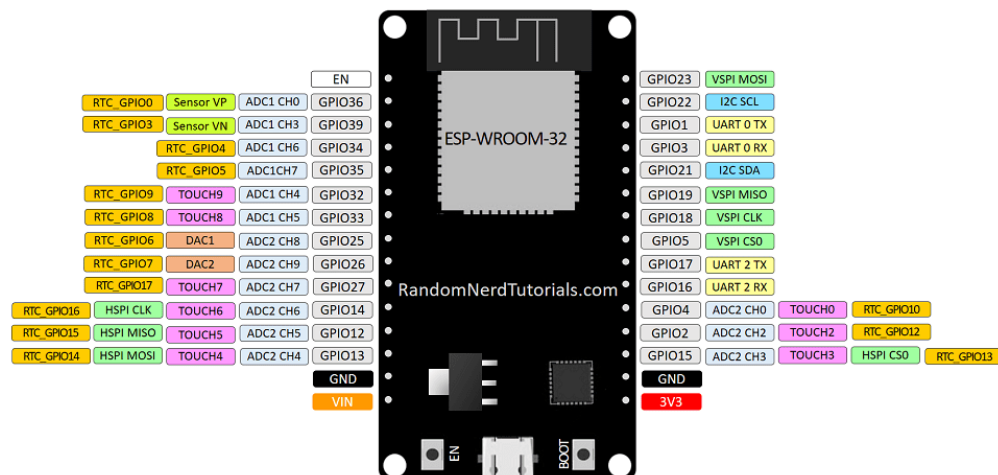
Verdien fra en Touch pinne kan nås i programmet ved følgende kommando:

```
int verdi; // Vil ha verdier typisk mellom 10 og 100
verdi = touchRead(4); // Leser tilstanden til GPIO 4, Touch 0
```

2.3 ESP - WROOM - 32 med 30 pinner

Den samme kretsen finnes i flere utgaver vi har av ulike årsaker bl.a. pris og tilgjengelighet valgt å bruke en 30 pinner utgave.

ESP32 DEVKIT V1 – DOIT version with 30 GPIOs





	EN	D23	GPIO23
GPIO36	VP	D22	GPIO22
GPIO39	VN	TX0	GPIO1
GPIO34	D34	RX0	GPIO3
GPIO35	D35	D21	GPIO21
GPIO32	D32	D19	GPIO19
GPIO33	D33	D18	GPIO18
GPIO25	D25	D5	GPIO5
GPIO26	D26	TX2	GPIO17
GPIO27	D27	RX2	GPIO16
GPIO14	D14	D4	GPIO4
GPIO12	D12	D2	GPIO2
GPIO13	D13	D15	GPIO15
	GND	GND	
	VIN	3V3	



Forskjellen på ESP32 (38 pin) og ESP32 (30 pin) er følgende:

GPIO	Merk	Br. 1	Br. 2	Br. 3	GPIO	Merk	Br. 1	Br. 2	Br. 4	GPIO	Merk	Br. 1	Br. 2	Br. 3	GPIO	Merk	Br. 1	Br. 2	Br. 3
0	0	I/O	A2-1	T4	10	D3	x	x	x	20	-	-	-	x	30	-	-	-	
1	Tx	Tx			11	CMD	x	x	x	21	21	I/O	SDA (I ² C)		31	-	-	-	
2	2	I/O	A2-2	T2	12	12	O	A2-5	T5	22	22	I/O	SCL (I ² C)		32	32	I/O	A1-4	T9
3	Rx	Rx			13	13	I/O	A2-4	T4	23	23	I/O		MOSI	33	33	I/O	A1-5	T8
4	4	I/O	A2-0	T0	14	14	I/O	A2-6	T6	24	-	-	-		34	34	I	A1-6	
5	5	I/O		CS0	15	15	I/O	A2-3	T3	25	25	I/O	A2-8	DAC1	35	35	I	A1-7	
6	CLK	x	x		16	16	I/O			26	26	I/O	A2-9	DAC2	36	VP	I	A1-0	
7	D0	x	x		17	17	I/O			27	27	I/O	A2-7	T7	37	-	-	-	
8	D1	x	x		18	18	I/O		CLK	28	-	-	-		38	-	-	-	
9	D2	x	x		19	19	I/O		MISO	29	-	-	-		39	VN	I	A1-3	



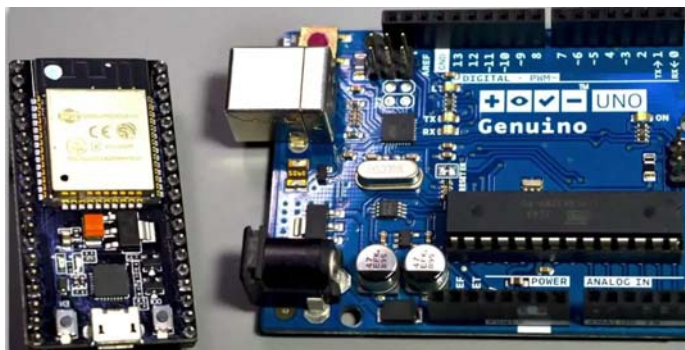
Forskjellen på ESP32 (38 pin) og ESP32 (30 pin) er følgende: Alle GPIO som er skravert i tabellen over finnes ikke tilgjengelig på pinner hos ESP32 (30 pin). Dvs. at ESP32 (30 pin) mangler følgende i forhold til ESP32 (38 pin): En GND, pin D0 – D3, CLK, CMD i tillegg til GPIO 0. Dvs. alt-i-alt så er det meste med hos ESP32 (30 pin) og vi velger å betrakte de to variantene som likeverdige til våre formål.

3 Programmering

Dette kapitlet gir noen grunnleggende tips til programmering av Arduino. De av dere som har erfaring med denne type programmering kan nøye dere med lese avsnitt 3.1, og 3.3, side 26, som omtaler installasjon av ekstrapakken for å kunne bruke Arduino-grensesnittet for ESP32.

3.1 Forskjeller og likheter hos Arduino og ESP32

La oss ganske kort sammenligne ESP32 og en “vanlig” Arduino, f.eks. UNO⁸.



Dersom vi ser på de to kortene er de svært forskjellige, dette gjelder på de fleste områder som f.eks. innebygde funksjoner, lager- og prosesseringskapasitet, antall GPIO-porter og kommunikasjonsmuligheter, ikke minst det siste siden ESP32 både har Bluetooth og WiFi (selv om noen Arduino varianter også har det). At prosesseringshastigheten er så høy skyldes dels den høye klokkefrekvensen på 240 MHz (16 MHz på de fleste Arduino) og at ESP32 har to prosessorkjerner.

Hva som likevel gjør ESP32 så attraktiv for Arduino-brukere er programvaren som er utviklet av Espressif, som har gjort det mulig å bruke mesteparten av de samme kommandoene og det samme programmeringsmiljøet (IDO) som Arduino. Likeså kan over 90% av bibliotekene som er utviklet for Arduino også brukes direkte på ESP32. Her må man imidlertid være litt oppmerksom og sjekke om det finnes spesialtilpassede biblioteker utviklet spesielt for ESP32. Det trengs imidlertid å legge inn noe tilleggsprogramvare som vi snart skal komme tilbake til (se avsnitt 3.3, side 26).

Naturlig nok finnes det noen tillegg i språket som er knyttet til de tilleggsfunksjonene som ESP32 forsyner oss med, ellers er det omtrent total overlapp. Programmer som f.eks. er skrevet for Arduino UNO kan godt lastes inn i editoren og kompileres og overføres til ESP32. Det man imidlertid må passe på er *at IO-portene er plassert annerledes*. Dette er jo et svært viktig poeng, dog er det håndterbart.

Normalt vil prisen for ESP32 ligge under Arduino. Man kan derfor se det slik at man får den økte lagerkapasiteten, hastigheten og WiFi/Bluetooth gratis.

8. Stoffet til denne sammenligningen er hentet fra <https://techexplorations.com/guides/esp32/begin/esp32ard/>



ESP32 er likevel ganske kompleks dersom man ønsker å utnytte mulighetene. Det anbefales derfor ikke å begynne med denne, men gjerne starte med en Arduino UNO. Vi velger likevel å gjøre det for at skrittet over til Internett of Things (IoT) skal være lettere.

3.2 Installasjon av programvare

Om du har installert Arduino IDE tidligere kan du hoppe over dette avsnittet og gå direkte til avsnitt 3.3, side 26.

La oss først se hvordan vi kan installere: Arduino programeditor, IDE.

3.2.1 Arduino programeditor, IDE

Nedlasting av programvare

Arduino programeditor og kompilator hentes fra:

<http://arduino.cc/hu/Main/Software>

Versjonen som er brukt i denne sammenheng er 1.8.13. Filen som har navnet *arduino-1.8.13-windows*, er pakket som en zip-fil. En tilsvarende fil er tilgjengelig for Mac fra samme nettsted.

Det er også helt greit å bruke Windows installer versjonen som finnes på samme side. Det er imidlertid erfart noen uregelmessigheter ved bruk av Windows app-versjonen så unngå gjerne den.

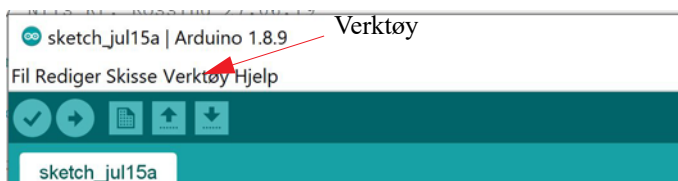
Installasjon av programvaren:

1. Klikk på *Windows installer* og følg anvisningen under installasjonen. For MAC-brukere klikk på *MAC OS X*

2. Programmet startes ved å klikke på programikonet:  .

3. Koble USB-kabelen til ønsket port.

4. Klikk på *Verktøy (Tools)* på menylinjen og velg *Kort*. Her velges hvilken variant i Arduino-familien du ønsker å jobbe med.




Etter at vi har installert tilleggsprogramvare for DOIT ESP-32 DEVKIT V1 så velger vi denne (se avsnitt 3.3).

5. Klikk på *Verktøy* på menylinjen og velg *Port*. Sjekk at riktig port (Com?) er valgt. Normalt står det hvilken Arduino-variant porten er koblet til eller velg det høyeste nummeret.







Programmet er nå klart til bruk og du kan skrive inn programlinjene. Når programmet er ferdig skrevet, skal det *kompile*res, dvs. overføres til en binærkode som mikrokontrolleren forstår. Der som programmet inneholder ulovlige kommandoer eller skrivefeil, vil kompilatoren varsle om det og vise på hvilken linje feilen er avslørt. Det er ikke nødvendigvis alltid der feilen befinner seg.

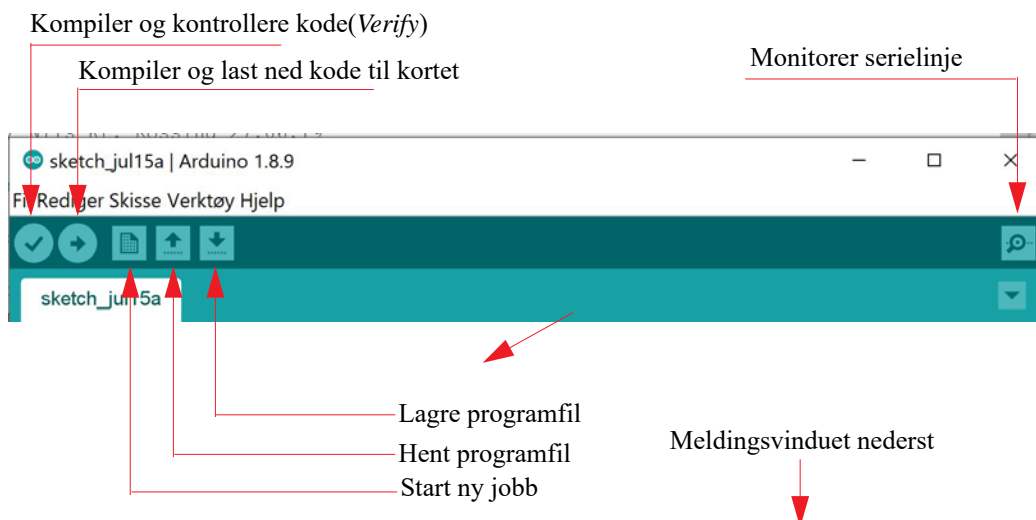


Dernest skal programmet lastes ned til kontrollereens minne (Arduino-kortet). Dette gjøres ved å trykke på knappen .

Kort oversikt over Arduino-editoren

Man finner følgende kommandoikoner på den grafiske menylinjen:

-  Kompiler og verifiser at koden er riktig
-  Kompiler og last ned programmet til kontrollenheten
-  Hent nytt “arbeidsark”
-  Hent en eksisterende programfil
-  Lagre programfil
-  Monitorer data sendt tilbake fra kontrollerkortet på serielinjen



Manglende kontakt med kortet

Det hender at en ikke oppnår umiddelbar kontakt med Arduino-kortet når en forsøker å laste ned et program. Feilmeldingen: **avrdude: stk500_getsync(): not in sync: resp=0x00** i meldingsvinduet betyr at det ikke oppnås kontakt med kortet. Dette kan skyldes flere ting:

- Kabelen er ikke tilkoblet, eller ødelagt
- Feil port er valgt av programeditoren, som kan endres ved å velge: *Verktøy* og *Port* fra menylinjen i editoren

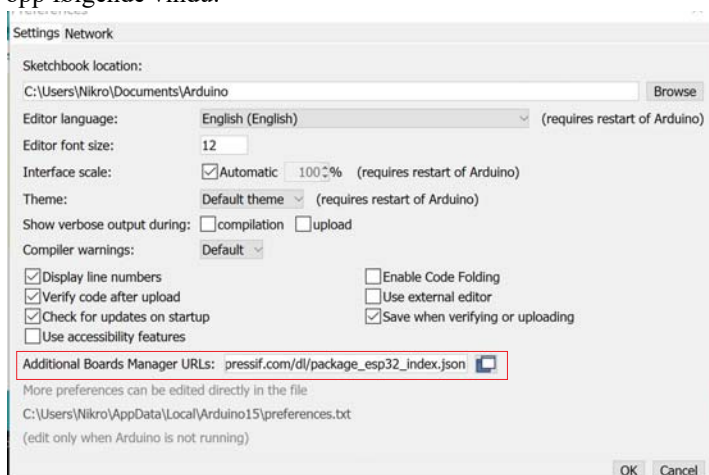


- Feil type kontroller-kort valgt
Endres ved å velge: *Verktøy* og *Kort* fra menylinjen for så å velge rett kort, i vårt tilfelle *Arduino/Genuino UNO*.
- Manglende drivere, i så fall må driverne installeres manuelt
- Eller rett og slett at programmet har hengt seg opp. I så fall kan man løse problemet med å lukke og starte programmet på nytt.

3.3 Installasjon av ekstra pakke for programmering av ESP32

I dette avsnittet skal vi se hvordan vi skal installere tillegget som gjør det mulig å bruke IDE til å programmere ESP32⁹.

1. Gå til menylinjen på Arduino IDE og velg: *File* → *Preferences*.
Du får da opp følgende vindu:

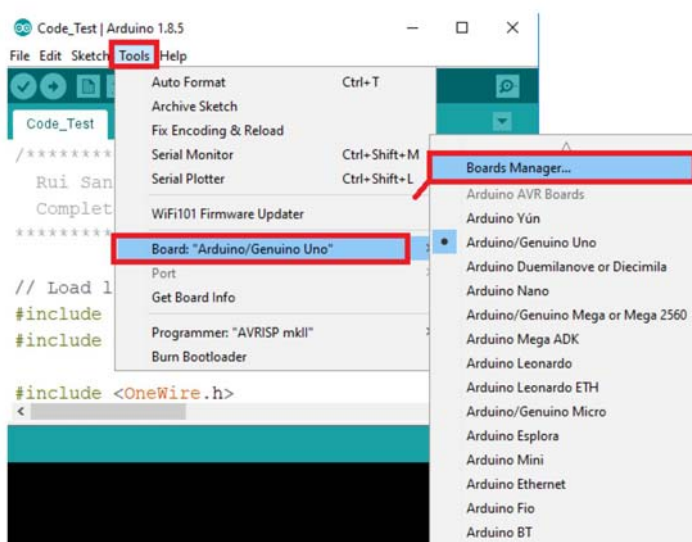


2. Legg inn følgende nettadresse https://dl.espressif.com/dl/package_esp32_index.json i feltet merket “*Additional Board Manager URLs*”. Dersom det alt ligger noe der, gå til slutten av linjen skriv komma og legger inn den nye adressen bak de eksisterende. Klikk deretter på “OK” knappen.

9. Beskrivelsen er delvis hentet fra Santos, Learn ESP32 with Arduino IDE (<https://randomnerdtutorials.com/learn-esp32-with-arduino-ide/>).



3. Gå så til *Tools* → *Boards* → *Boards Manager* som vist på figuren under:



4. Når du åpner Boards Manager får du opp et nytt vindu, hvor du skriver esp32 i søkevinduet, etter kort tid vil du få opp følgende:



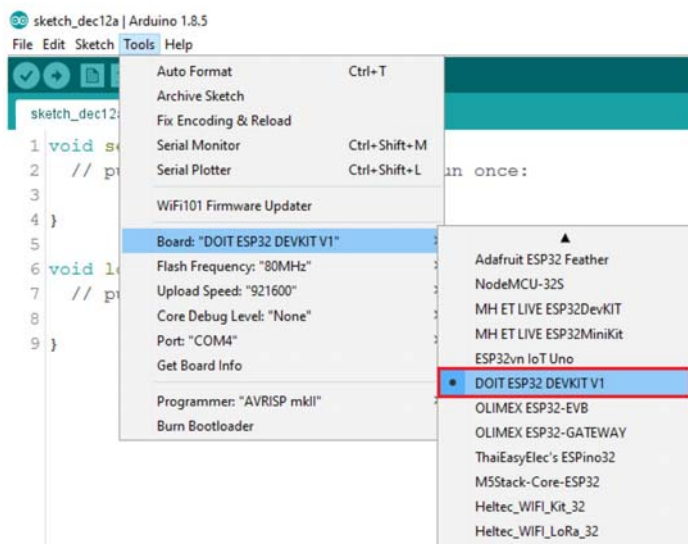
Dersom du ikke får opp et forslag til installasjon, så skyldes det sannsynligvis at du skrev inn feil adresse under *Preferences*, da *det* er adressen der *Boards Manager* leter etter tillegget.

Vi skal nå teste at tillegget fungerer. Gjør følgende:

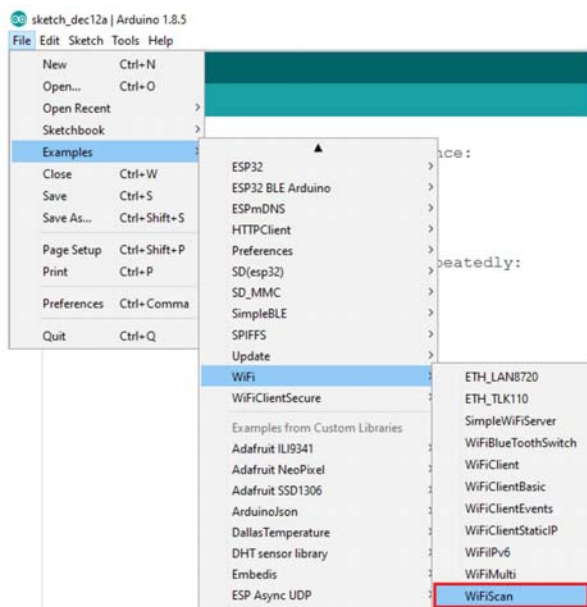
5. Åpne Arduino IDE (om du ikke alt har den åpen)




6. Gå til *Tools* på menylinjen og velg: *Tools* → *Boards* → *ESP32 Arduino* → *DOIT ESP32 DEVKIT V1* fra lista til høyre som vist på figuren under (legg merke til at det i nyere versjoner finnes et nivå til: *ESP32 Arduino*). Du har nå gitt IDE'en beskjed om hvilket kort du har tenkt å bruke.



7. Gå til *File* på menylinja og velg: *File* → *Examples* → *WiFi* (under *DOIT ESP32 DEVKIT V1*) → *WiFiScan*. Dette er et program som scanner hvilke nettverk som finnes i nærheten.





8. Last opp programmet og velg *kompiler* og *overfør* til mikrokontrolleren ved å trykke følgende symbol på menylinja: 

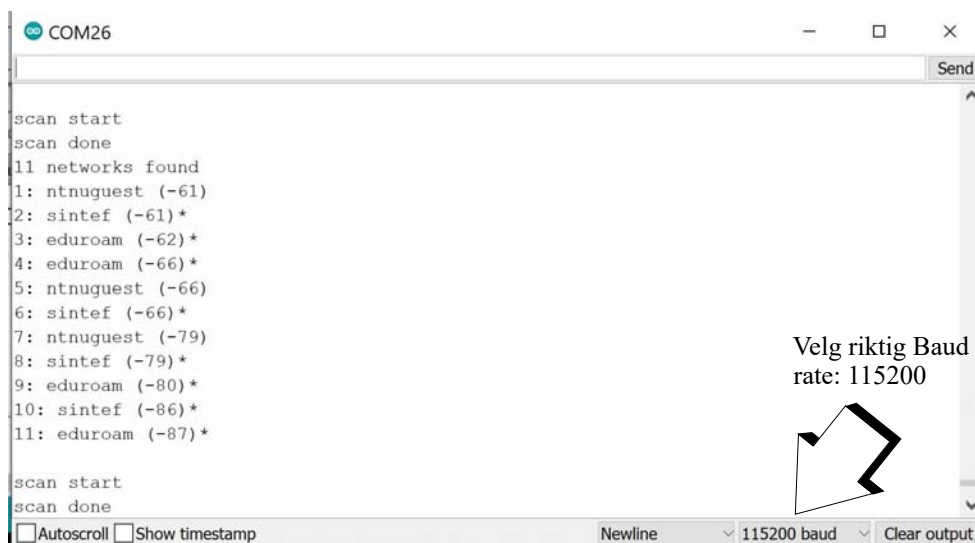


9. Kompileringen tar ganske lang tid og når opplastingen skjer er det normalt at det dukker opp rød tekst i meldingsvinduet nederst.

Programmet skal nå kjøre i mikrokontrolleren ESP32 og returnere informasjon om mulig nettverk som finnes i nærheten. Disse skrives ut i monitoren.

NB! Dersom opplastingen venter og ikke kommer videre ev. gir feilmelding: TRYKK BOOT-knappen og hold den nede til programmet begynner å lastes, dette gjelder vår 30-pins variant. Noen ganger kan det også være nødvendig å trykke ENABLE-knappen for å starte programmet.

10. Åpne monitoren ved å trykke på symbolet . Pass på at overføringshastigheten (Baud rate) er satt til 115 200, som vist på figuren under.





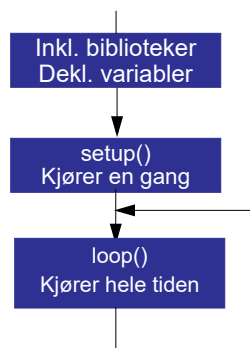
11. Du skal nå se tilgjengelig trådløse nettverk. I vårt tilfelle hele 11 stykker som vist på figuren over.

Du er nå klar til å programmere i ESP32.

3.4 Programstruktur

Alle Arduino-programmer består av to hovedfunksjoner:

- **void setup()** som kjøres bare *en* gang hver gang programmet starter. Dette skjer etter at programmet er lastet opp, når man trykker på restartknappen eller åpner monitoren.
- **void loop()** er en funksjon som gjentas så lenge Arduino'en har spenning.
- **Biblioteker og deklarasjon** av globale variabler plasseres gjerne helt i starten.
- Det er også vanlig å skrive *egne funksjoner* som gjerne legges på slutten av programmet, utenfor og etter loop-funksjonen(), men kan i prinsippet legges hvor som helst utenfor setup() og loop() funksjonene.

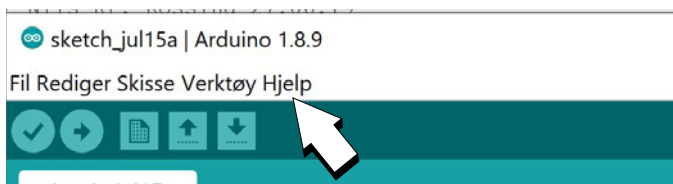


3.5 Viktige kommandoer

Referansemanualen til Arduino C++ for bruk ved programmering av Arduino-prosessorer finnes på følgende nettside:

<http://arduino.cc/en/Reference/HomePage>

Referanse manualen ligger også under *Hjelp* på menylinjen i programeditoren som vist på figuren til høyre.



3.5.1 Generelle kommandoer

Programstruktur

Som nevnt foran så består programmet hovedsakelig av to funksjoner omsluttet av *klammeparenteser*. I void setup()-funksjonen initieres mikrokontrolleren, mens selve programmet legges under void loop()-funksjonen som vist under.

```
void setup()  
{  
    <initiering>  
}
```



```
void loop()  
{  
    <programkode>  
}
```

Initiering av dataoverføring til PC

Under uttestingen kan det være praktisk at data leses tilbake til monitoren i Arduino editoren. Datahastigheten settes opp i setup-funksjonen med kommandoen: `Serial.begin(9600)`; her satt til 9 600 baud¹⁰ (her ca. 9600 *bit* pr. sekund) som vist under:

```
void setup()  
{  
    Serial.begin(9600);  
}
```

I noen tilfeller kan det være hensiktsmessig å øke overføringshastigheten. 115 200 tegn pr. sekund er vanlig å bruke. Pass på at mottakerhastigheten for monitoren må settes til samme hastighet. Dette gjøres i monitoren som vist på figuren under.



Kommentarer:

Kommentarer kan skrives hvor som helst og begynner med

// Dette er en kommentar

Disse blir fjernet under kompilering og overføres ikke til mikrokontrolleren. Kommentarene er derfor kun en hjelp for den som skriver programmet, ev. andre som skal lese og forstå programmet senere.

Ønsker man å kommentere bort flere linjer kan dette gjøres slik:

10. Baud rate - Angir hvor mange symboler som kan overføres pr. transmisjonsperiode. For et binært system vil baud raten tilsvare bit pr. sek. For mer komplekse former for modulasjon (f.eks. QPSK) vil hver transmisjonsperiode kunne overføre flere bit (2, 4 eller flere), dog på bekostning av følsomheten for støy.



```
/*  
Alle tre linjene  
vil bli betraktet  
som kommentarer  
*/
```

Deklarasjon av lokale og globale variable:

I programspråkene C og C++ må alle variabler deklarereres før de kan brukes. Deklarasjonene må inneholde *type* og *navn* på variabelen og gjøres gjerne i starten av programmet før `setup()` rutinen. Siden den deklarereres utenfor funksjonene, så blir de **globale**, dvs. at variablene kan brukes i alle funksjonene og beholder innholdet i variabelen.

Deklarering kan også gjøres *innenfor hver* funksjon. Slike variabler gjelder da bare innenfor den funksjonen og kalles da **lokale**. Under er vist deklarasjon av de vanligste typer variabler. I dette tilfelle vil de bare gjelde innen funksjonen `loop()`:

```
void loop()  
{  
    Int a;           // deklarasjon av 16 bit heltallsvariabel (word)  
    char b;          // deklarasjon av 8 bit karakttervariabel (byte)  
    char c, d;        // deklarasjon av to 8 bits karakttervariabler (byte)  
    float e;          // deklarasjon av variabelen e som et desimaltall f.eks. 1,65 (32 bit, dobbel word)  
    unsigned long f;  // deklarasjon av 4 byts heltallsvariabel f (32 bit) uten fortegn  
    boolean g;        // deklarasjon av en boolsk variabel g som kan ha verdiene 0 eller 1  
    <dernest følger programkoden>  
}
```

Skriv tilbake til PC – til monitoren i programeditoren:

Følgende kommandoer skriver en variabel eller en tekst tilbake på terminalvinduet (monitoren) i programeditoren.

```
Serial.print(a);      // Skriver variabelen a til en linje på skjermen,  
                      // neste skrivekommando skriver på samme linje  
Serial.println(a);    // Skriver variabelen a til en linje på skjermen og skifter linje,  
                      // neste skrivekommando skrives derfor på ny linje  
Serial.println("Hallo"); // Skriver teksten Hallo til en linje på skjermen,  
                      // neste skrivekommando skriver på ny linje
```

Det er også mulig å kombinere tekst og variabler i samme printtkommando:

```
Seriel.println("Trykk:", a); // Skriver teksten Trykk: til en linje på Arduino monitoren,  
                             // etterfulgt av innholdet i variabelen a, og skifter deretter til ny linje  
Seriel.println(f, 2);        // Skriver desimalvariabelen f til terminal på PC med to desimaler
```




Definer digitale porter som inngang eller utgang:

En *port* er et fysisk tilkoblingspunkt på kretsen som kan kobles til eksterne krets-elementer, sensorer eller aktuatorer f.eks. LED. Disse kan enten være analoge eller digitale, innganger eller utganger. Siden en strømkrets må være sluttet, vil alle porter forholde seg til jord på kortet (dvs. minus på batteriet).

Kontrolleren ATmega 328 (Arduino UNO) har en rekke *porter*, 14 digitale (0 – 13) og 6 analoge (0 – 5), det samme har ESP32, se oversikt i avsnitt 2.3, side 20. De digitale portene kan både være innganger eller utganger og hver port må derfor defineres som en inn- eller utgang. Dette gjøres i `setup()`-funksjonen:

```
void setup()
{
    pinMode(8,OUTPUT);           // Definerer pinne 8 som utgang
    pinMode(7,INPUT);            // Definerer pinne 7 som inngang
    pinMode(6,INPUT_PULLUP);     // Definerer pinne 6 som inngang med pullup-motstand, dette er nyttig
                                // for at inngangen ikke skal henge fritt (sveve)
                                // når den ikke er tilkoblet noe
}
```

Dersom en inngang defineres med *pullup-motstand* betyr det at utgangen er koblet til pluss på batteriet ved en “stor” motstand internt i kretsen. Det betyr at dersom porten ikke er tilkoblet noe annet på utsiden så vil den ha verdi 1 (dvs. 5V for Arduino og 3,3 V for ESP32).

Lese og skriv til en digital port:

Digitale porter som er definert som utganger, kan enten settes til høy eller lav spenning. Digitale porter som er definert som innganger, kan lese av om spenningen på porten er høy eller lav. Dette gjøres med følgende kommandoer:

```
boolean bolsk;                // Definerer den boolske variabelen bolsk kan ha verdien 0 eller 1
int heltall;                  // Definerer en heltalls variabel heltall, kan meget vel brukes for 0 og 1
void loop()
{
    digitalWrite(8, HIGH);     //Setter port 8 høy (5 V ev. 3,3 V)
    digitalWrite(8, LOW);      //Setter port 8 lav (0 V)
    bolsk = digitalRead(7);    // Leser den digitale verdien på port 7 og setter i variabelen bolsk
    heltall = digitalRead(6);  // Leser den digitale verdien på port 6 og setter i variabelen heltall
}
```

Det er imidlertid vanligere å bruke heltallsvariabler (krever 2 byte) for å holde digitale verdier lest fra innganger. Man sparer imidlertid litt lagerplass dersom man bruker boolske variabler (krever 1 byte lagerplass).

Vent-kommando:

Dersom vi ønsker at programmet skal ta en pause kan vi skrive følgende:

```
delay(1000);                  //Stopper programmet i 1000 msek (1 sek)
delayMicroseconds(100);      //Stopper programmet i 100 µs, 0,1 msek (0,0001 sek)
```



Dette medfører imidlertid at heller ingenting annet kan gjøres i denne tiden. Har man ikke råd til å miste denne tiden bør man finne andre løsninger, f.eks. bruk av `millis()`;

Aritmetiske operasjoner

```
sum = a + b;           // Summen av a + b settes i variabelen sum
diff = a - b;          // Differansen av a - b settes i variabelen diff
prod = a * b;          // Produktet av a * b settes i variabelen prod
kvo = a / b;           // Koeffisienten av a / b settes i variabelen kvo
```

Variabelnavnene *sum*, *diff*, *prod* og *kvo* er bare valgt som eksempler.

Dersom man ønsker å endre innholdet i en av variablene, kan dette gjøres slik:

```
a = a + b;             // Summen av a + b settes i variabelen a
```

Dette ser ikke lengre ut som en fornuftig “ligning” slik vi kjenner den fra matematikken. Her legges verdien i *a* til *b* før den så legges tilbake til *a*.

Les verdi fra en analog inngang og skriv til monitor (PC)

Syntaksen for lesing fra en analog inngang kan skrives som:

```
<variabel> = analogRead(<analog port>); //Den analoge porten kan ha verdier fra 0 til 5 V hos UNO
                                           // Ev. 0 til 3,3 V hos ESP32
```

Eksempel 1:

```
Int verdi;                // Deklarerer variabelen verdi
verdi = analogRead(0);     // Digitale verdien fra analog inngang 0 leses inn i variabelen verdi
```

Eksempel 2:

```
void loop()
{
    int pressure;          //Deklarerer pressure som en heltalls-variabel
    pressure = analogRead(1); //Leser av trykksensoren på AD-kanal 1
    Serial.println(pressure); //Skriv resultatet tilbake til monitoren (PC)
}
```

Legg merke til *Serial.print()*; skriver *uten* påfølgende linjeskift, mens *Serial.println()*; skriver *med* påfølgende linjeskift.

Eksempel 3:

```
void loop()
{
    int temperature;        //Deklarerer temperatur som heltallsvariabel
    temperature = analogRead(5); //Leser av temperatursensoren på AD-kanal 5
    Serial.println(temperature,2); //Skriv resultatet tilbake til Arduino monitoren (PC) med 2 desimaler
}
```



De analoge portene er på innsiden tilkoblet en analog til digital omvandler (AD-konverter) som gjør om spenningen 0 – 5 V til en digitalverdi fra 0 – 1023 (0000000000 – 1111111111). Dvs. AD-konverteren har 10 bits nøyaktighet. Tilsvarende kan ESP32 håndtere analoge spenninger fra 0 – 3,3 V og har AD-konvertere med en oppløsning på 12 bit.

Sløyfer

Noen ganger har man behov for å gjenta en operasjon et vist antall ganger da vil en *for-loop* egnet. Andre ganger vil man ha behov for å gjenta en operasjon så lenge en betingelse er oppfylt, i så fall kan man benytte en *while-loop*.

For-loop – For å gjenta mange like operasjoner (sløyfer)

For-loop'en egner seg spesielt godt til å gjenta den samme operasjonen et bestemt antall ganger, kanskje med ganske små forandringer mellom gjentakelsene. Denne skrives slik:

```
for(int x = 0; x < 100; x++)
{
    // Her skrives koden som skal gjentas
}
```

x er en heltallsvariabel (int x) som brukes som teller. Denne starter på verdien 0 (int x = 0;) økes med 1 (x++) for hver runde i loopen og stopper ikke før den når opp til 100 (x < 100;). De ulike uttrykkene skilles med semikolon.

Eksempel:

```
for(int x = 0; x < 100; x++)
{
    Serial.println(i);
}
```

Denne skriver ut tallene 0 – 99 til monitoren, ett tall for hver linje. Ønsker man å skrive ut tallene 0 – 100 kan man f.eks. skrive:

```
for(int x = 0; x <= 100; x++)
{
    Serial.println(i);
}
```

While-loopen – For å få programmet til å vente i en sløyfe til betingelsen ikke lenger er oppfylt

Det kan for eksempel være tilfelle når man venter på svar fra en sensor. While-loopen kan skrives slik:

```
while (<logisk betingelse>)
{
    // Her skrives koden som skal gjentas mens programmet venter
}
```



If-setning – For å kunne gjøre “veivalg” i programmet

Noen ganger ønsker vi å gjøre forskjellige ting på bakgrunn av ulike betingelser, da kan vi bruke *if-setninger*. Igjen tar vi et konkret eksempel:

```
if (i < 10)
{
    // Gjør dette dersom innholdet i variabelen i < 10
}
else if (i == 10)
{
    // Gjør dette dersom innholdet i variabelen i = 10
}
else
{
    // Gjør dette dersom noe annet er tilfelle i dette tilfellet i > 10
}
```

Avhengig av verdien til variabelen *i* utfører programmet ulike operasjoner. Parentesen etter *if()* skal inneholde en betingelse, denne kan være enkel, som her, eller ganske sammensatt. Man bruker da *logiske operatører* for å undersøke om noe er større enn (>), mindre enn (<) eller lik (==). Legg merke til det doble likhetstegnet. På tilsvarende måte kan en skrive større eller lik (>=) og mindre eller lik (<=).

Man kan lage flere betingelser med ulik respons ved å bruke en eller flere *else if()* med nye betingelser. Men alt som faller utenfor de definerte betingelse kan samles opp i en *else*.

Legg merke til at linjen, *if (i < 10)* ikke avsluttes med semikolon men etterfølges av { }. Dvs. at *if()* er egentlig en funksjon. Heller ikke klammeparentesene etterfølges med ; (semikolon). Slik er språket definert.

3.6 Bruk av I²C og biblioteker

I dette avsnittet skal se på hvordan vi kan bruke I²C buss til å overføre data fra sensorer utstyrt med denne typen kommunikasjonsgrensesnitt, som f.eks. trykkmåleren BMP180 eller BMP280 og displayet SSD1306 eller TM1637. Til dette bruker vi som oftest spesiallagede biblioteker som må installeres dersom de ikke alt finnes blant bibliotekene som følger med Arduino editoren (IDE).

Den såkalte I²C bussen består at to kommunikasjonslinjer som gjør det mulig å sende data etter hverandre (på serieform) mellom ulike kretser.

Standardbiblioteket for I²C følger med Arduino programvaren, men krever at man inkluderer “header”-filen: “wire.h” øverst i koden ved å skrive:

```
include <wire.h>
```

Dette medfører at programmet får tilgang til en del funksjoner knyttet til kommunikasjon via I²C-kommunikasjonslinjene.



For mange spesiallagde kretser finnes det spesielle pakker med funksjoner som gjør dem lettere å bruke. Slike biblioteker må gjerne hentes ned fra leverandøren eller andre og installeres i program-editoren.

NB! For Arduino UNO brukes A4 (SDA) og A5 (SCL) for å kommunisere med I2C-bussen. For ESP32 brukes GPIO21 (SDA) og GPIO22 (SCL).

I neste avsnitt skal vi se hvordan vi kan laste ned og installere biblioteker.

3.6.1 Installasjon av pakkede biblioteker generelt

Som eksempel bruker vi biblioteket som trengs for å lese data fra trykksensoren BMP180 som også inneholder en temperatursensor.

1. Last ned biblioteket i pakket form (*.zip). Denne pakken inneholder header-filer (*.h), biblioteksfunksjoner (*.cpp) og gjerne eksempler på bruk av sensoren. Er man heldig inneholder den også en beskrivelse av biblioteksfunksjonene. Man finner gjerne biblioteket på nett ved søke etter den aktuelle kretsen etterfulgt av ordet *library* om adressen ikke er kjent.

Fra og med Arduino IDE versjon 1.5 kan man fra menylinjen velge:

Sketch/Include library/Manage Libraries

og skrive inn den aktuelle sensoren i søkefeltet. Man får da beskjed om biblioteket er installert eller forslag til hvor man kan hente det fra.

2. **Last ned biblioteket fra ekstern kilde:**

Dersom biblioteksfilen for komponenten ikke finnes, kan man hente biblioteksfilene til BMP180 fra ulike steder, f.eks. fra SparkFun's hjemmeside:

<https://learn.sparkfun.com/tutorials/bmp180-barometric-pressure-sensor-hookup-/installing-the-arduino-library>

For BMP280 gjelder følgende adresse:

https://github.com/adafruit/Adafruit_BMP280_Library

Vi har også lagt dem ut under den blå hefteserien

3. **Installer biblioteket:**

Dernest installeres biblioteket i Arduino-programvaren ved å velge:

*Sketch/Include Library/Add *.ZIP Library* for så å velge den nedlastede zip-pakkede filen.

Biblioteksfilene blir dermed lagt under *Sketch/include library* og eksempler under: *File/Eksempler*

4. **Inkluder headerfilen (*.h) i programmet**

Det aktuelle biblioteket inkluderes i programmet ved å velge:

Sketch/Include Library/<aktuelle biblioteket>

Dermed er header-filen på plass i programmet.



3.6.2 Bibliotek for bruk av I²C

Også I²C bussens bibliotek må lenkes inn i koden når den skal brukes. Bussen fungerer ved at en “master” styrer kommunikasjonen. Vanligvis er dette Arduino mikrokontrollerkortet som automatisk blir definert som master ved kommandoen *Wire.begin()*; kommandoen i setup-funksjonen.

Biblioteket – Wire.h:

Biblioteket inneholder noen funksjoner som vi skal forsøke å beskrive her:

“Header file”:

```
#include <Wire.h>
```

“Header” kommandoen knytter I²C-biblioteket til programmet slik at kompilatoren vet hvilket bibliotek funksjonene skal hentes fra.

```
Wire.begin();
```

Denne funksjonen definerer Arduino mikrokontrolleren som master for kommunikasjonen, dvs. at all kommunikasjon med sensorer og andre periferikretser som er koblet til I²C-bussen styres av Arduino’en. Siden Arduino’en er masteren, er det ikke nødvendig å definere en adresse (*Wire.begin(<adresse>)*;) for denne. Ev. kan adressen være et tall mellom 0 – 127 som identifiserer masteren på bussen.

```
Wire.beginTransmission(<adresse>);
```

Med funksjonen *Wire.beginTransmission(<adresse>)*; kan man starte en dataoverføring til en enhet på bussen med den spesifiserte adressen.

Vi skal senere se hvordan vi anvender dette i praksis.

For tilkobling til I²C hos ESP32 ta gjerne en titt på følgende nettside: <https://randomnerdtutorials.com/esp32-i2c-communication-arduino-ide/#1>



4 Oppgavesamling

I denne delen skal vi ta for oss de enkelte delprosjektene og bygge opp et lite system bit for bit før vi til slutt å sette det hele sammen til det endelige produktet.

4.1 Kursets prosjektoppgave – Oppdrag

Som nevnt tar kurset sikte på å bygge opp kompetansen gradvis gjennom å arbeide med små maskin- og programvare moduler. Det endelige produktet (oppdraget) kan beskrives slik:

Oppdraget: *Man ser for seg et rom med adgangskontroll. Adgangskontrollen skjer ved bruk av RFID og personlige adgangskort. Ved ankomst til døra inn til rommet avleses ID-kortet og adgang gis dersom vedkommende er akkreditert for rommet ved at låsen går opp. I forbindelse med korona utbruddet våren 2020 så settes det begrensninger til hvor mange som kan være i et rom samtidig. Det skal derfor lages en optisk port ved døra som teller antallet som til en hver tid befinner seg i rommet. Dette medfører at den optiske slusa må registrere både at noen går inn og at noen går ut. Ved inngangen er det plassert et display som til en hver tid viser antallet som befinner seg i rommet. Det stilles også krav til å redusere forbruket av elektrisk energi slik at når første man passerer inn gjennom slusa slås lyset i rommet på og når sistemann forlater rommet skal alt lys slås av og ventilasjonsanlegget skrues ned på minimum.*

Dette produktet kan betraktes som et system som består av mange deler, men som kan brytes ned i enkeltdeler som er håndterbare.

Utfordring: Utfordre elevene til å dele opp systemet i mindre delprosjekter eller deloppgaver og be dem lage en plan for rekkefølgen de vil bruke for å løse de ulike oppgavene. Ser de noen spesielle utfordringer knyttet til delprosjektene, ev. når de skal sette dem sammen til et system?

Vi har valgt å starte med å lage en enkel og ganske “dum” optisk portal for å registrere passeringer. For så å avslutte med å modifisere portalen slik at den skiller mellom passeringer som går ut og inn. Vi har valgt å gjøre det slik for å få en passende progresjon i vanskelighetsgraden til prosjektet. Det samme gjelder bruk av RFID adgangskort som er lagt helt til slutt.

4.1.1 Oppdeling i delprosjekter

Under har vi foreslått en oppdeling i delprosjekter og en rekkefølge og en antydninger om ev. utfordringer. Dette er en tabell elevene bør lage selv og som kan variere fra elev til elev. La elevene gjerne arbeide i par og oppmuntre til diskusjon.

#	Beskrivelse av delprosjekt.	Utfordringer
0	Oppbygging av målejigg.	
1a	Koble opp den optiske portalen.	
1b	Opprett en optisk deteksjon ved passering av den optiske porten. Undersøk hvor pålitelig portalen er.	Undersøk om den reagerer raskt nok. Ev. hva kan gjøres for å øke reaksjonsevnen?



#	Beskrivelse av delprosjekt.	Utfordringer
1c	Sett terskelverdien slik at den optiske porten blir mest mulig pålitelig.	
1d	Tell antall passeringer og vis tallet i monitoren. Vis resultatet bare når det skjer en passering.	Hvordan vise et resultat bare når det skjer en endring, ikke ellers?
2a	Koble opp displayet!	
2b	Opprett forbindelse med displayet og finn ut hvordan man skrive tall til displayet.	
2c	Sørg for at displayet teller fra 0 – 10 med 1 sek mellomrom, stopp programutførelsen når du kommer til 10.	Hvordan stoppe programutførelsen etter 10 passeringer? Ev. legge skrijving til displayet i en egen funksjon.
3a	Kombiner programmene 1d og 2c og lag et program som teller opp med én for hver passering.	
4a	Koble opp servoen.	Er det nødvendig å installere et eget bibliotek for servoer?
4b	Lage eget lite program for å styre servoen som ønsket fra 0 → 90° og tilbake. Bommen skal være igjen i 5 sek og åpen i 3,5 sek.	Pass på at servoen starter og stopper i riktig posisjon i forhold til jiggjen.
5a	Kombiner program 3a med servoen 4b slik at bommen går opp hver gang noen passerer den optiske porten samtidig som antallet passeringer telles og vises i displayet.	
6a	Koble opp releet!	
6b	Styr releet på og av fra programmet.	
6c	Koble releet til en lampe eller en lysdiode og bekreft at releet virkelig slår inn.	I denne sammenhengen er det kanskje tilstrekkelig å registrere at releet slår inn i riktig situasjon.
7a	Monter en optisk port nr. 2!	
7b	Sjekk at den fungerer som forventet!	
7c	Finn en måte å skille personer som går inn fra de som går ut.	Det kan være utfordrende å finne en god og pålitelig løsning på denne oppgaven
7d	Tell opp og ned en variabel og vis den i monitoren!	
8a	Kombiner den optiske porten med servoen og bommen slik at den går opp når noen er på vei inn eller ut, og lukker når de har passert.	



#	Beskrivelse av delprosjekt.	Utfordringer
9a	Kombiner den optiske porten med bommen og telleren som viser antallet i rommet til en hver tid på displayet.	
10a	Kombiner den optiske porten med bommen og telleren med displayet og releet som slår på lyset når noen er i rommet og ellers har lyset slukket.	
11a	Koble opp RFID-leseren	Pass på at de riktige terminalene kobles til.
11b	Det skal lages en programkode som leser av RFID-kortleseren og viser ID-koden i monitoren. Denne koden skal senere legges inn som en kode som har adgang slik at bommen åpnes når kortet nærmer seg RFID-leseren.	
11c	Sørg for å lese av kort som legges inn til RFID-leseren og vis kortnummeret i monitoren.	
11d	Numrene som er godkjent legges inn i programvaren. Sammenlign det leste kortnummeret med en liste over godkjente nummer.	Lagring av en liste av akkrediterte nummer og sammenlign med alle
12a	Kombiner RFID med åpning og lukking av bommen. Når et akkreditert kort nærmer seg RFID-leseren så åpner bommen. Sørg for at bommen åpnes når et kort blir godkjent. Det holder at bommen går ned etter 5 sekunder.	
13a	Lag en kode som sammenstille alle deloppgavene og løser det totale oppdraget. Det skal ikke være nødvendig å bruke RFID når man går ut av rommet, kun den innvendige optiske portalen.	Det kan være lurt å inkludere en og en funksjon og teste for hver utvidelse.

4.2 Programmetts grunnstruktur

Vi vil i stadig større grad overlate programmeringen til dere, men utgangspunktet er standard rammeverk for Arduino og for ESP32 som vist i listingen under.

```
// Inkludering av biblioteker
// Deklarasjon av globale variable
void setup()
{
    // Kode som kun trengs å kjøres en gang i starten av programmet
    // plasseres her
}
```



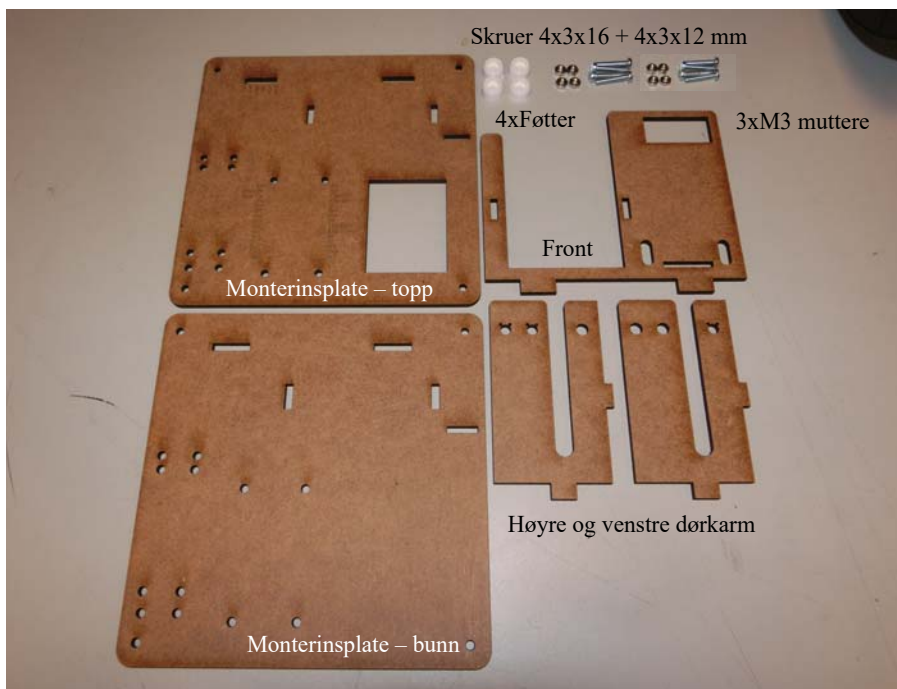
```
void loop()  
{  
    // Legg hovedprogrammet her, det som skal kjøres gjentatte ganger  
}  
  
// Ev. egendefinerte funksjoner
```

4.3 Oppbygging av målejigg

Vi har valgt å lage en målejigg som forestiller en inngangsdør til et forsamlingslokale og oppdraget som skal løses er knyttet til adgangskontroll. Målejigg gjør det mulig å montere de ulike sensorene slik at vi lett kan teste ut de delene av systemet. I dette underkapittelet skal vi se hvordan målejiggen kan bygges opp. Den er skåret på laserkutter i 3 mm MDF og kan lett mangfoldiggjøres. En forminskert utgave av malen er vist i vedlegg C.1, side 138.

4.3.1 Mekanisk byggeveiledning

Vi har valgt å bygge opp grunnplata av to deler. En årsak er at det er lettere å forholde seg til en materialtykkelse enn to, men kanskje det viktigste er at koblingsbrettet skal få god støtte på undersiden ved at det ligger an mot den underste monteringsplata. Uten støtte kan man lett skyve ut koblingsskinnene som bare holdes på plass av en dobbeltsidig tape. I tillegg monterer vi på fire 3D-printede bein. Dette er strengt tatt ikke nødvendig, men ser litt mer proft ut. Figuren under viser delene.



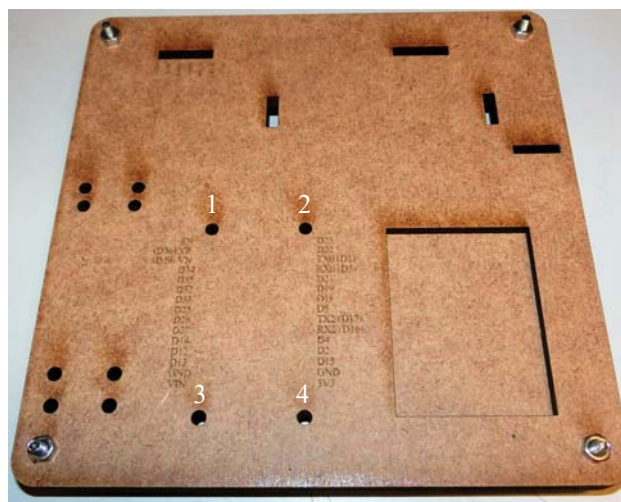


Målejiggen består av følgende deler:

- 1 monteringsplate – bunn 3 mm MDF
- 1 monteringsplate – topp 3 mm MDF
- 1 frontplate 3mm MDF
- 1 venstre dørkarm 3 mm MDF
- 1 høyre dørkarm 3 mm MDF
- 4 x 3D-printede bein
- 4 x 3 x 12 mm skruer for å holde mikrokontrollerkortet ESP32 på plass
- 4 x 3 x 16 mm skruer for montering av de fire beina

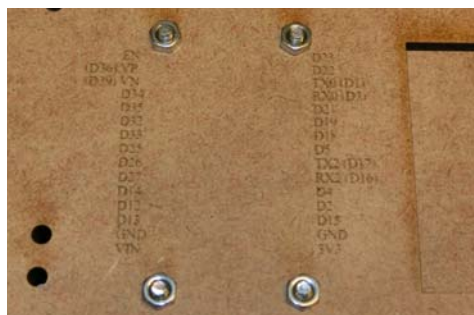
Montering

Legg de to monteringsplatene oppå hverandre slik at hullene stemmer overens. Plata med det rektangulære hullet og den graverte teksten legges øverst. Det rektangulære hullet skal holde koblingsbrettet. Bruk de fire 16 mm skruene til å sette sammen monteringsplata. Tre et bein inn på hver av skruene. Mutterne kan gjerne være på toppen som vist på figuren under.



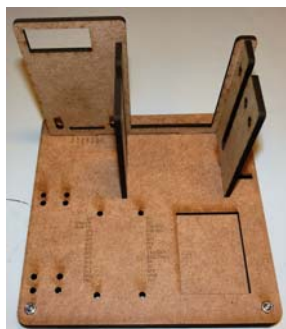
Fot montert på undersiden av plata

De fire 12 mm skruene skrues inn i de fire hullene merket med 1 – 4 på bildet over. Skruene skrues nedenfra og opp med mutteren på toppen som vist på figuren til høyre. Det er meningen at skruene skal stikke ca. 3 – 4 mm opp over mutterne slik at mikrokontrolleren kan tres ned på skruene og ligge an mot mutterne.

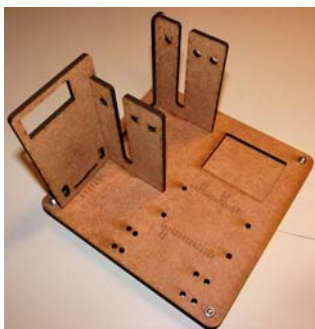




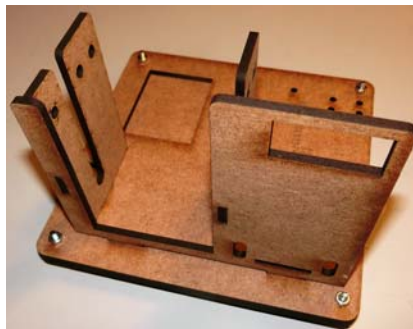
Derneast monteres høyre og venstre dørkarm til frontplata før alle tre trykkes ned i hullene i monteringsplata.



Sett bakfra



Sett fra siden



Sett forfra

4.3.2 Elektrisk byggeveiledning

Elektroniske komponenter

Etter som oppdraget skrider fram vil det bli behov for å montere følgende elektroniske komponenter i målejiggen:

1. ESP32 – mikrokontroller
Mini koblingsbrett
Optisk måleportal med hvit LED og LDR (del 1)
2. Display TM1637 med 4 x 7-segmentsiffer
3. Servo SG90 med bom
4. 1 polet rele
5. Optisk måleportal med hvit LED og LDR (del 2)
6. RFID RC522

Tabell over disponerte porter

Vi vil i dette avsnittet forsøke å vise hele oppkoblingen noe som er viktig siden enkelte porter må reserveres. Under er vist en tabell over disponering av porter til ulike bruk:

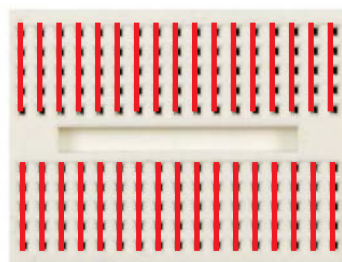
GPIO nr.:	Pinne	Bruk	GPIO nr.:	Port nr.:	Bruk
GPIO 1	TX0		GPIO 21	D21	RFID (SDA)
GPIO 2	RX0		GPIO 22	D22	RFID (RESET)
GPIO 3	D3		GPIO 23	D23	RFID (MOSI)
GPIO4	D4		GPIO 25	D25	
GPIO5	D5	LED2 (Portal) (OUTPUT)	GPIO 26	D26	
GPIO 12	D12		GPIO 27	D27	
GPIO 13	D13	Servo (Bom) (OUTPUT)	GPIO 32	D32	Display CLK (OUTPUT)



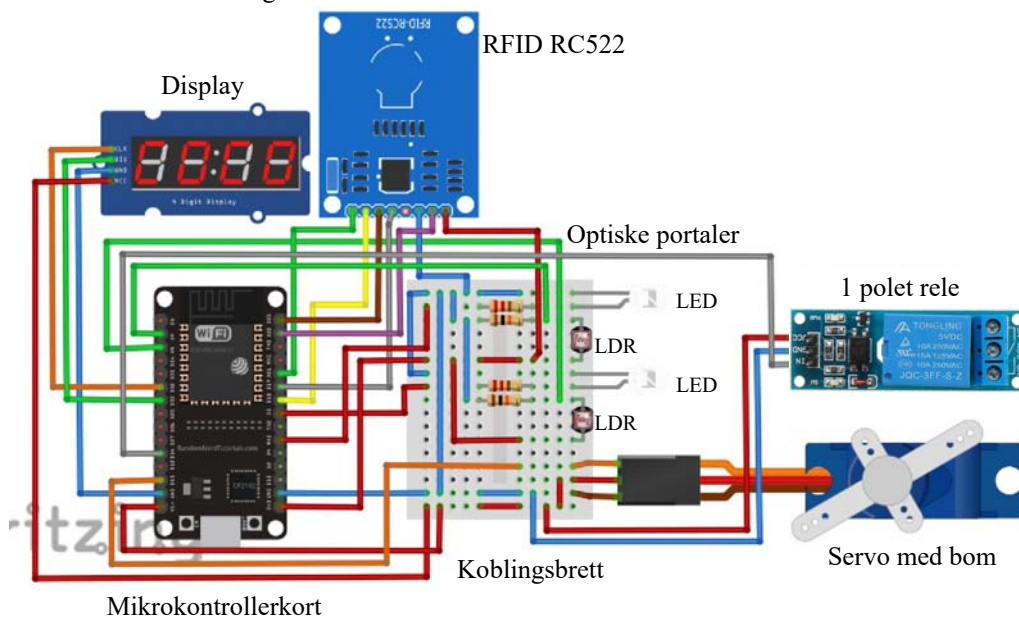
GPIO nr.:	Pinne	Bruk	GPIO nr.:	Port nr.:	Bruk
GPIO14	D14	Rele (Lys) (OUTPUT)	GPIO 33	D33	Display DIO (OUTPUT)
GPIO 15	D15		GPIO 34	D34	LDR2 (Analog) (Portal) (INPUT)
GPIO 16	RX2	LED1 (Portal) (OUTPUT)	GPIO 35	D35	
GPIO 17	TX2		GPIO 26	VN	LDR1 (Analog) (Portal) (INPUT)
GPIO 18	D18	RFID (SCK)	GPIO 39	VP	
GPIO 19	D19	RFID (MISO)			

Forslag til koblingsskjema

Vi bygger opp deler av kretsen på et mini koblingsbrett. Dette er litt smått, men slik kretsen er bygget opp så skal det være rikelig med plass. Til høyre på bildet til høyre er vist med streker hvilke hull som er elektrisk forbundet ved hjelp av metallskinner under hullene. Dette er det viktig å være klar over når man skal bygge opp kretsen.



Figuren under viser det komplette koblingsskjemaet over kretsen. Vi vil gradvis bygge opp hele kretsen etter som vi trenger de ulike delene.





4.4 Oppdrag 1 – Oppbygging av en enkel optisk portal

Det første oppdraget som skal løses er:

Oppdrag 1: Lag et lite program som registrerer at noen passerer den optiske porten, likegyldig hvilken vei

Den enkle optiske portalen skal registrere at det går personer gjennom døråpningen, men foreløpig ikke hvilken vei de går. Det kommer vi tilbake til senere.

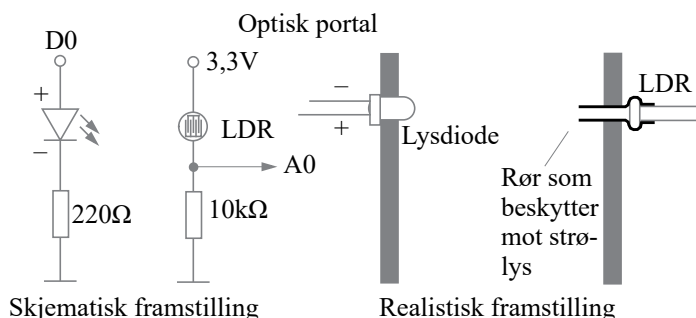
La oss først se hvordan en optisk portal kan virke.

4.4.1 Elektronisk virkemåte for en enkel optisk portal

Det finnes flere mulige løsninger for å lage en optisk portal. Vi har valgt å bruke en lysdiode (LED) og en lysfølsom resistor, en LDR (Light Dependent Resistor).

Lys fra lysdioden treffer den lysfølsomme motstanden (LDR), som når denne belyses, har *lav resistans*.

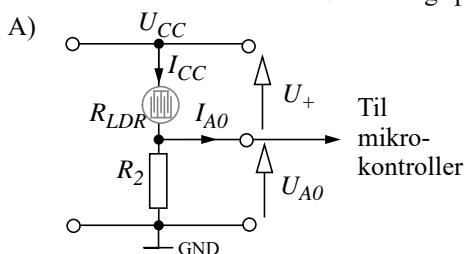
Når vi bryter lysstrålene kommer LDR'en i skyggen og den får *høy resistans*. For konvertere endringen i resistans til en målbar spenning bruker vi en *spenningsdeler*.



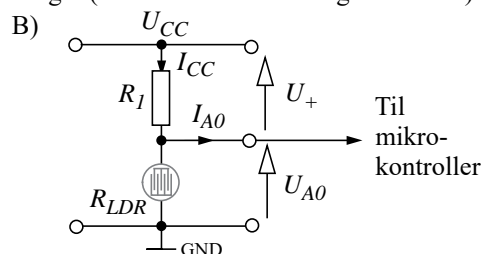
4.4.2 Spenningsdeleren

En spenningsdeler er nyttig for å konvertere en variasjon i resistive sensorer, f.eks. i en LDR, til en spenningsvariasjon som vår mikrokontroller kan registrere på en av sine analoge innganger. Den som ønsker en utdypende forståelse av spenningsdeleren, se avsnitt 5.1.

Ofte holder det at vi forstår hvordan spenningsdeleren prinsipielt fungerer siden vi ikke trenger å kjenne de eksakte verdiene for strømmer og spenninger (se innrammet tekst i figuren under).



Økende lysstyrke → Økende U_{A0}



Økende lysstyrke → Redusert U_{A0}

Vi tar utgangspunkt i **tegning A** på figuren over.

Vi antar at strømmen I_{A0} på inngangen av den analoge porten $A0$ er tilnærmet lik null. Vi antar videre at *lysstyrken øker*. Det betyr at motstandsverdien til R_{LDR} blir mindre. Når R_{LDR} blir mindre, vil strømmen I_{CC} øke. Siden den samme strømmen går gjennom både R_{LDR} og R_2 forteller Ohms lov oss at spenningen U_{A0} øker ($U_{A0} = I_{CC} R_2$). Dvs. at *økende lysstyrke gir økende spenning inn på mikrokontrolleren*.

Tilsvarende kan vi argumentere for oppkoblingen i **tegning B**:

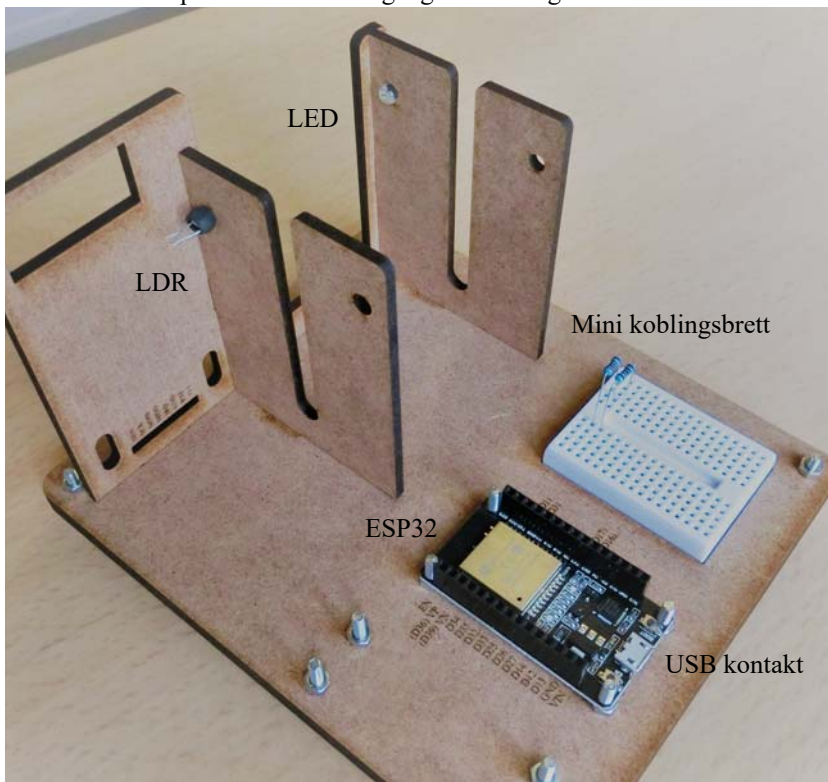
Dersom vi nå antar at *lysstyrken øker* og resistansen til R_{LDR} blir mindre, så vil strømmen I_{CC} øke som sist. Siden I_{CC} også går gjennom R_1 , forteller Ohms lov oss at spenningen U_+ også vil øke. Siden $U_{A0} = U_{CC} - U_+$, så vil U_{A0} avta. Dvs. at *økende lysstyrke gir minkende spenning inn på mikrokontrolleren*.

Her er det opp til oss å velge hva vi ønsker. Vi har valgt løsning A.

La oss koble opp *Den enkle optiske portalen*:

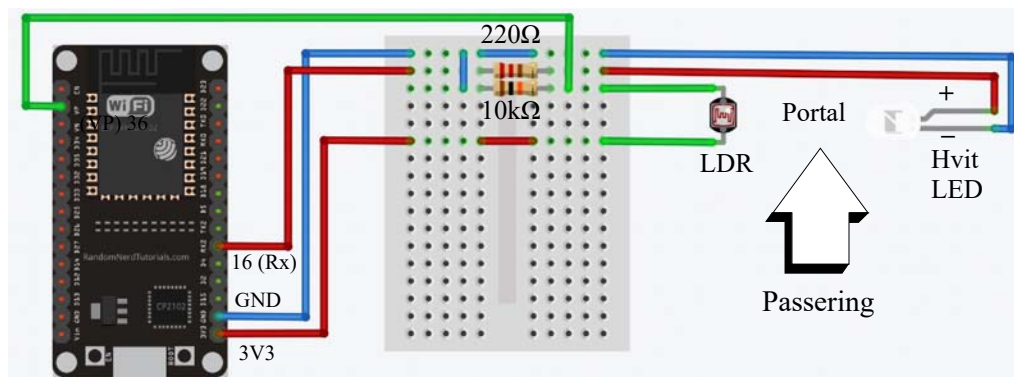
4.4.3 Oppkobling

Bildet under viser monteringen av ESP32, koblingsbrett med $1 \times 220\Omega$ og $1 \times 10k\Omega$, dessuten ar en LDR og en LED montert på hver side av inngangen. Ledningene er ikke montert ennå.





Figuren under viser et forenklet skjema av oppkoblingen. Vi bruker en seriemotstand på 220Ω for å begrense strømmen i lysdioden (langt bein anode (+) og kort bein katode (-)). Vi velger $10k\Omega$ som motstand i serie med LDR. En LDR kan betraktes som en motstand uten polaritet (retningen er likegyldig).



4.4.4 Programmering

Oppdrag 1: Lag et lite program som registrerer at noen passerer den optiske porten, likegyldig hvilken vei

Vi beskriver først den *sekvensielle rekkefølgen* av hva vi ønsker å gjøre:

- Slå på lysdioden
- Lese av den lysfølsomme motstanden
- Sette en terskelverdi for den avleste verdien for godkjent passering
- Telle opp en variabel (teller) ved hver passering
- Skrive variabelen til monitoren

Åpne en ny skisse

Velg *File* → *New* og hent opp en ny skisse, lagre skissen under navnet *OptiskPortal-1*

Slå på lysdioden

Vi velger å bruke GPIO-port nr. 16 for å tenne lysdioden. Vi definerer denne som en utgang, og slår på dioden:

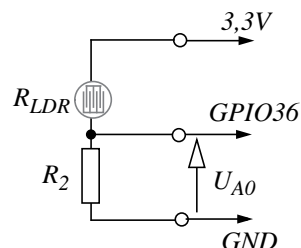
```
// Optisk portal-1
int pinLED = 16;
void setup()
{
  pinMode(pinLED, OUTPUT);
}
```




```
void loop()
{
    digitalWrite(pinLED, HIGH);
}
```

Les av lysfølsom motstand

Vi har koblet opp en spenningsdeler med den lysfølsomme sensoren mot +3,3 V og seriemotstanden til jord. Vi velger å bruke A1–0 (GPIO36) som inngang. Dessuten ønsker vi å skrive den avleste verdien fra LDR'en til monitoren. For testing kan det være lurt å skrive ut verdien en gang hvert sekund.



Legg inn følgende kommandoer på rett sted i programmet:

```
int pinLDR = 36; // Deklarerer pinnennr. for avlesning av LDR
int verdiLDR; // Deklarerer variabel som holder avlest verdi fra LDR

Serial.begin(9600); // Initierer seriell kommunikasjon til monitoren

verdiLDR = analogRead(pinLDR); // Leser av spenning på pinLDR
Serial.println(verdiLDR); // Skriver ut avlest verdi til monitor
delay(1000); // Venter 1000 millisekunder
```

Dersom kommandoene er lagt inn på rett sted så burde vi i monitoren se en rekke tall som angir den målte lysstyrken ved LDR'en.

Vil den avleste verdien minke eller øke dersom fingeren kommer mellom lysdioden og sensoren?

Sett fingeren mellom og se hvordan verdien endrer seg.

Legg inn terskelverdi og øk tellerverdi

Vi skal nå legge inn en terskelverdi slik at vi kan detektere at det skjer en passering. Til det skal vi bruke en *if()*-setning. Når en passering har skjedd så skal vi øke en teller med 1. For å hindre at noen “sniker” seg gjennom uten å bli registrert, må vi sørge for at vi punktprøver sensoren tilstrekkelig ofte. Skriv så ut telleverdien til monitoren.

Legg inn følgende kommandoer på rett sted i programmet:

```
int terskelLDR = 500;
int antall = 0;

if(verdiLDR < terskelLDR)
{
    antall = antall + 1;
}
```

Registrer og diskuter hva som skjer når dere lar fingeren passere portalen!

Dere vil sannsynligvis registrere to ting:

1. Telleren starter på 1
2. Det telles flere ganger dersom fingeren holdes i ro i portalen



Hva skyldes det?

1. *At telleren starter på en skyldes sannsynligvis at dere tenner lysdioden rett før dere måler. Dvs. at den rekker ikke å bli slått på ordentlig før målingen starter*
2. *At telleren teller flere passeringen når fingeren er i ro, er ikke så rart, da programmet fortsetter å gå selv om fingeren står i ro. Dermed registreres en ny passering neste gang programmet har gått en runde.*

Hvordan kan vi bøte på dette?

1. *Man kan sørge for at lysdioden har vært tent en tid før programmet går inn i loopen. Dette kan gjøres ved å legge tenningen inn i setup()-funksjonen og legge inn en liten forsinkelse*
2. *For å hindre mange tellinger når fingeren befinner seg i portalen må en sørge for at programmet stopper og venter til fingeren er passert.*

Det siste kan vi gjøre ved å legge inn en *while()*-løkke inne i *if()*-funksjonen:

```
while(verdiLDR < terskelLDR)
{
    verdiLDR = analogRead(pinLDR);
}
```

Ei *while*-løkke vil fortsette å gå i løkken så lenge betingelsen er oppfylt. Dvs. at programmet vil befinne seg i løkka så lenge den avleste verdien på LDR'en er lavere enn terskelen. Så lenge dette er tilfelle vil en utføre det som er mellom klammeparentesene {} hvor den fortsetter å lese av verdien på LDR'en og kommer ikke videre før fingeren er borte.

Dermed har vi laget et program som teller passeringer og vi er ferdige med første oppgave.

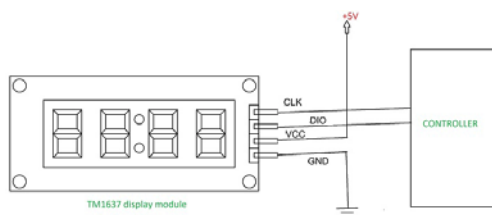
4.5 Oppdrag 2 – Skriv til displayet

Vi ønsker nå å skrive ut telleverdien til displayet, men først må montere og koble det opp.

Oppdrag 2: *Lag et lite program som skriver ut tall til displayet TM1637*

4.5.1 Oppkobling

I denne oppgaven skal vi bruke et lite praktisk display med fire siffer, TM1637. Displayet er enkelt å bruke og trenger bare 4 tilkoblinger til mikrokontrolleren:



To linjer tilfører spenning og to tilfører informasjon:

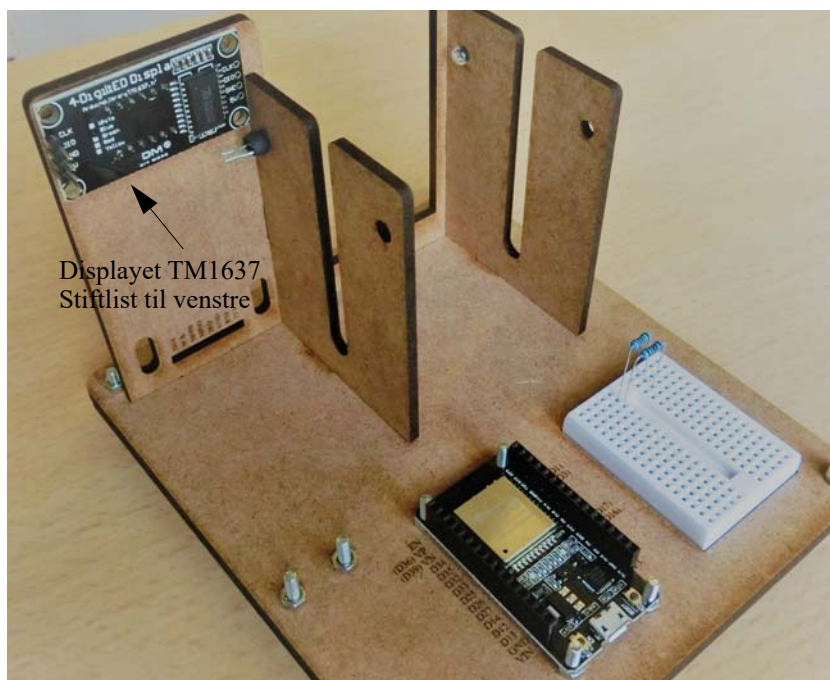
5V – Displayet skal ha spenning på 5V dette får man på VIN (5V) på ESP32-kortet (→ VIN)

GND – Gir jordforbindelse (–) (→ GND)

DIO – Serielle data inn (→ GPIO 33)

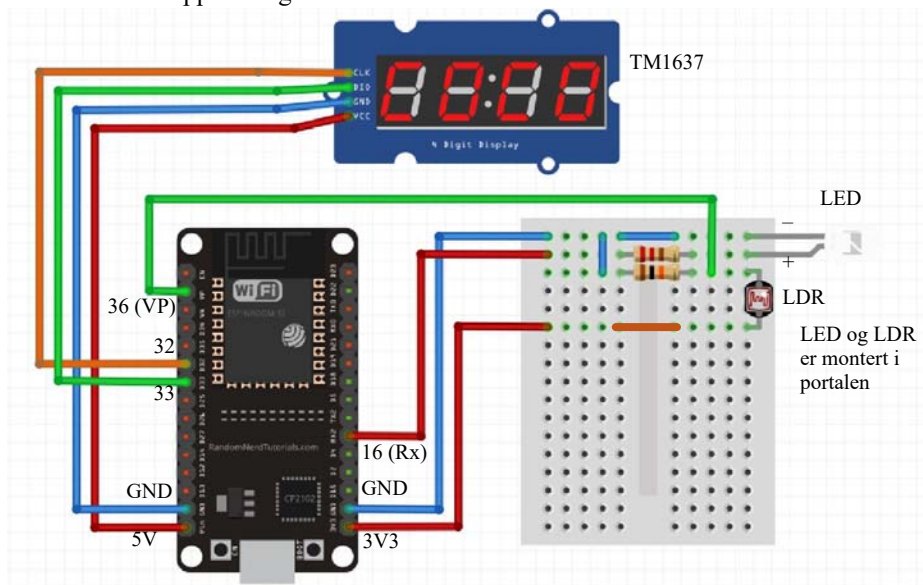
CLK – Klokkesignal (→ GPIO 32)

Bildet under viser monteringen av Displayet TM1637. Ledningene er utelatt.





Figuren under viser oppkoblingen

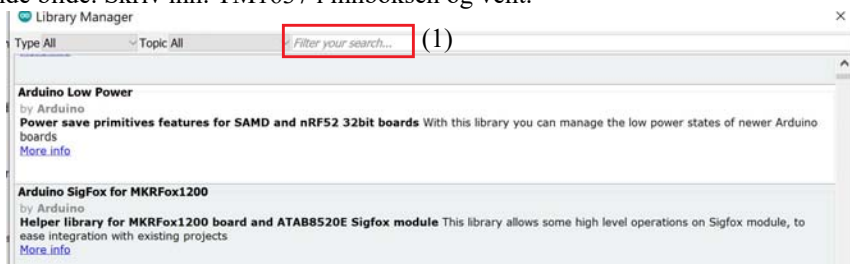


4.5.2 Programmering

For å bruke displayet TM1637 så trenger vi å installere biblioteket som hører til dette displayet.

Installasjon av bibliotek:

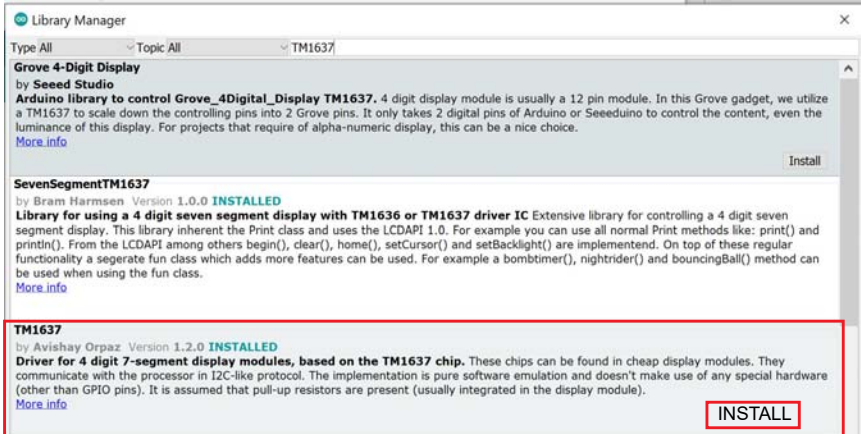
1. Velg *Sketch* menylinjen. Velg så *Include library* → *Library manage*. Du vil da få opp følgende bilde. Skriv inn: TM1637 i innboksen og vent.





2. Etter hvert vil det dukke opp flere alternative biblioteker for TM1637. Vår erfaring er at den som er rammet inn er den som har fungert best ved bruk sammen med ESP32.

(2)



(3)

3. Velg INSTALL nederst i høyre hjørne. Om den ikke markert, hold musa over den så vil den dukke opp. Etter kort tid er biblioteket installert og klart til å brukes.

Skriv et testprogram¹¹:

Det kan være lurt å skrive et test-program for å se at displayet fungerer som tiltenkt. Som test lager vi en teller som teller opp fra 1 til 10, for deretter å stoppe: Vi viser her programmet for så å forklare hver enkelt kommando:

```
/*
 * I denne øvelsen skal vi teste displayet
 * og lære hvordan vi skriver ut tall til
 * Nils Kr. Rossing 03.07.20 og Kåre-Benjamin H. Rørvik 15.07.20
 */

#include <Arduino.h>
#include <TM1637Display.h>

const byte PIN_CLK = 32;    // Definer CLK pin og velg port D32
const byte PIN_DIO = 33;    // Definer DIO pin og velg port D33

TM1637Display display(PIN_CLK, PIN_DIO);
// Definer instansen display av typen TM1637Display

int antall = 0; // Definer en variabel antall og sett den lik 0
```

11. Koden kan hentes ned fra <https://www.ntnu.no/skolelab/bla-hefteserie> under fanen Yrkesfaglærerkurs – ESP32 - Grunnkurs programmering



```
void setup()
{
    display.setBrightness(7); // Setter lysstyrken. Fra 0 (min) til 7 (max).
}

void loop()
{
    display.clear();           // Tøm displayet for data
    display.showNumberDec(antall); // Vis tall
    delay(1000);               // Vent i 1000 millisek. (1 sek.)
    antall = antall + 1;        // Øk telleren med 1
    while(antall == 11){}      // Stopp og vent her for alltid
}
```

Tips til forståelse:

- `//` – betyr at det som kommer bak på linjen er kommentarer og blir ikke inkludert i programmet
- `/*` – betyr at alt som kommer mellom `/*` og `*/` betraktes som kommentarer selv om det går over flere linjer
- `#include <TM1637Display.h>`
– kommandoen som inkluderer biblioteket slik at vi kan bruke funksjonene for å kommunisere med displayet
- `TM1637Display display(PIN_CLK, PIN_DIO);`
– Kommandoen setter et navn på displayet, nemlig *display*, og forteller hvilke datapinner dette displayet er koblet til. Dersom vi f.eks. hadde hatt to displayer så kunne vi ha kalt dem for *display-1* og *display-2* og koblet dem til forskjellige pinner.
- `display.setBrightness(7);`
– Kommandoen setter lysstyrken til displayet, fra 1 (min.) til 7 (maks.)
- `display.clear();`
– Kommandoen fjerner alt som står på displayet og setter det “blankt”
- `display.showNumberDec(antall);`
– Kommandoen skriver innholdet av variabelen *antall* ut på displayet som desimaltall fra 0 – 9999
- `while(antall == 11){}`
– Kommandoen stopper programmet når det har talt opp til 11. Dvs. når variabelen er blitt 11. Legg merke til at sammenligning skrives slik: `==`

For mer informasjon om kommandoer ved bruk av displayet TM1637 se: <https://github.com/avis-horp/TM1637/blob/master/TM1637Display.h> . Se også avsnitt 5.5, side 93.

Eksperimenter med programmet

Forsøk følgende:

- Endre koden slik at programmet teller fra 0 – 100 på 10 sek. og stopper der.
Ta gjerne tida.



4.6 Oppdrag 3 – Tell antall passeringer

Oppdrag 3: Lag et program som registrerer passeringer og viser antallet på displayet

Vi skal nå kombinere de to programmene som vi har laget slik at displayet viser antall passeringer som vi gjør med fingeren.

Det enkleste er at dere åpner de to programmene dere har laget i tillegg til en ny skisse. Lagre filen under navnet *DisplayAntallPasseringer-3*. Så kopierer dere fra de to tidligere programmene over i det nye.

4.6.1 Oppkobling

Dersom dere har gjort de to foregående oppgavene så skal all oppkobling være gjort.

4.6.2 Programmering

Siden dere har alt som trengs av kommandoer i de to foregående programmene, så vil vi her bare gi noen råd det kan være nyttig å ha med seg når dere kombinerer dem.

1. Unngå å deklarere samme variabelen to ganger.
Deklarasjonene gjøres før `setup()`-funksjonen
2. Rekkefølgen av kommandoene plassert i `setup()`-funksjonen er ikke så viktig.
3. Pass på at det som er plassert i `setup()`-funksjonen fortsatt settes der, og at det som står i `loop()`-funksjonen plasseres der.
4. Tenk gjennom rekkefølgen av kommandoene i `loop()`-funksjonen
5. Skriv gjerne ut variablene til monitoren for å sjekke f.eks. avlest verdi på lysfølsom motstand. Sjekk at terskelen har en fornuftig verdi.

Test programmet

Testing av programmet er viktig:

Undersøk følgende:

- Sjekk om dere får dobbeltregistrering av passeringer
- Sjekk om dere mister registrering av en passering

Har dere forslag til forbedringer?

4.7 Oppdrag 4 – Montering og styring av servo

I dette avsnittet skal vi se nærmere på bruk av servoer

Oppdrag 4: Monter og programmer en bom som er plassert i døråpningen. Bommen skal åpnes hvert 10 sekund, og lukkes etter å ha stått helt åpen i 3,5 sekund

Men la oss først se litt på hva en servo er (dere som er godt kjent med servoer kan gå til avsnitt 4.7.2, side 57).



4.7.1 Servoer

Det finnes flere typer servomotorer, det er de som kun kan dreie en vinkel i området fra 0 – 180°, f.eks. SG90 som vi skal bruke her. Og de som oppfører seg som en vanlig motor og går hele runden rundt. Disse kalles *360° servoer* eller *kontinuerlige* (f.eks. FS90R). Det spesielle med servoer er at dreievinkelen eller hastigheten kan styres ganske nøyaktig ved hjelp av lengden av en puls.

0 – 180°



SG90

360° – Kontinuerlig

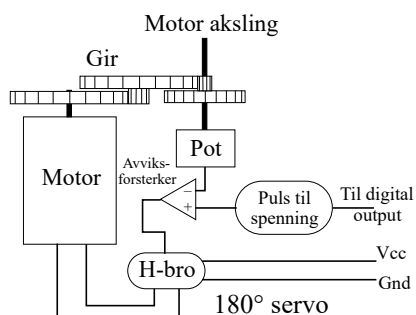


FS90R

Virkemåte

En 180° servo er en motor hvor man ved hjelp av en kommando kan dreie akslingen en bestemt vinkel mellom 0 – 180°. Dette skjer ved hjelp av en intern tilbakekobling som vist i figuren til høyre. Servoen styres av en puls på styreinngangen. SG90 er en slik 180° servo som er laget for spenninger fra 4,8 – 6V, men fungerer også for 3V, da med mindre dreiemoment.

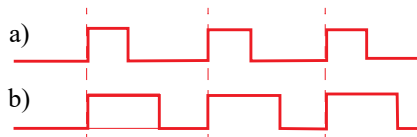
En *kontinuerlig roterende servo* kan dreie 360° og vil fungere som en vanlig motor. En slik vil *ikke* ha tilbakekobling som vist på figuren til høyre.

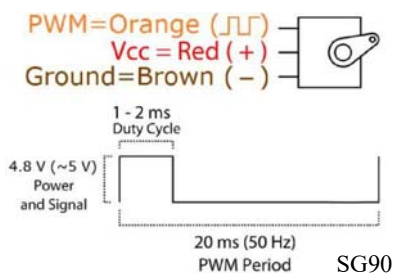


Motorene har normalt tre ledninger. I tillegg til spenning, Vcc (typisk 5 V (ev. 3,3 V)), og jord (GND), så tilføres et *styresignal* som hentes fra en av de digitale portene hos mikrokontrolleren. Det bør være en port som kan tilby PWM – *Pulsbreddemodulasjon*. Hos ESP32 kan alle porter tilby pulsbreddemodulasjon, dog ikke samtidig.

Pulsbreddemodulasjon

Pulsbreddemodulasjon (PWM – Puls Width Modulation) er en meget nyttig egenskap som utnyttes i mange ulike sammenhenger. En intern timer genererer et pulstog med en gitt frekvens, hvor pulslengden kan programmeres. Dette kan være en god måte å regulere lysstyrken til en lysdiode. Med en kort pulslengde i forhold til periodetiden, får vi svakt lys. Er pulslengden lang i forhold til periodetiden øker lysstyrken (se figuren over til høyre). Selv om det pulsbreddemodulerte signalet brukes på en annen måte hos en servo, så er utgangspunktet det samme, PWM.





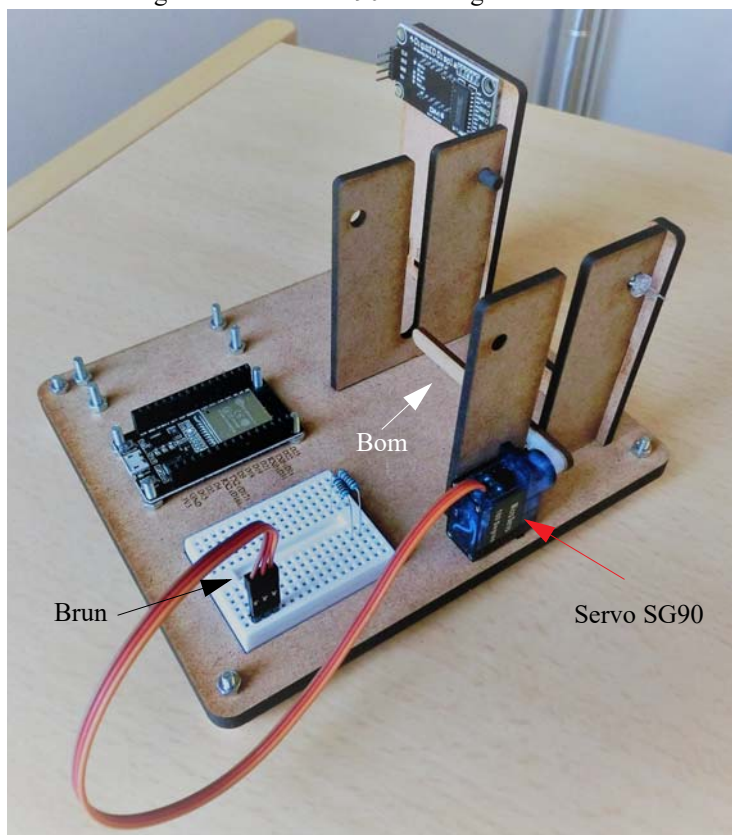
SG90: 180° motorene styres altså av et pulstog¹². Figuren til venstre viser en typisk situasjon for SG90. Den forlanger en puls på mellom 1 – 2 ms, som gjentar seg med en periode på 20ms (50Hz).

En pulslengde på 1,5 ms vil posisjonere servoen i posisjon 0°. En puls på 1,0 ms vil dreie den -90° mot høyre, mens en puls på 2,0 ms vil dreie den 90° mot venstre. Ved å endre på pulslengden så endres posisjonen. Fargene på figuren over angir fargene på de tre ledningene til servoen.

For mer informasjon om servoer se avsnitt 5.6, side 96.

4.7.2 Oppkobling

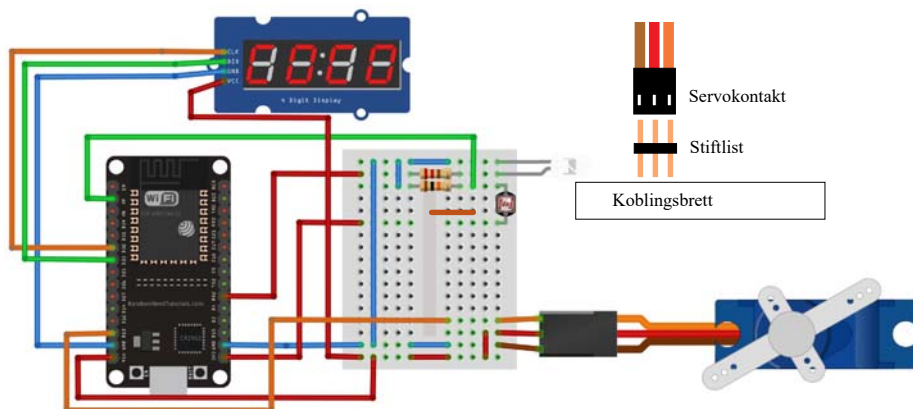
Bildet under viser monteringen av servoen SG90. Ledningene er utelatt.



12.http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf



Siden vi har tilgang til 5V på vår ESP32 så bruker vi den som spenningsforsyning og port D13 for styring som vist på figuren under. Siden servomotoren benytter en hylsekontakt med tre ledninger bruker vi en stiftlist med tre stifter slik at vi får muligheten til å stikke den ned i koblingsbrettet. For å få god kobling forskyver vi stiftene slik at de stikker like langt ut på begge sider av plastholderen.



4.7.3 Programmering

Opprett en ny programskisse og kall den *ServoTest-4*.

Oppdrag 4: Skriv et lite program som setter servoen i 0 og 90°. Juster gjerne posisjonverdiene slik at bommen står helt vertikalt og helt horisontalt

Også til denne trengs et eget bibliotek som kan hentes fra: <https://github.com/RoboticsBrno/ServoESP32> eller velg *Sketch* → *Include library* → *Library manage* og skriv *servoESP32* i søkefeltet. Da vil følgende komme opp:



Trykk på **INSTALL** nederst i høyre hjørne og biblioteket installeres.

Biblioteket er særdeles enkelt å bruke:

1. Inkluder biblioteket i fila ved hjelp av følgende:

```
#include <Servo.h>
```

Plasseres i starten av fila. Merk at standard servo-bibliotek for ordinære Arduino-kretser gir feilmelding. Det er derfor viktig å installere *ServoESP32*
2. Lag et servoobjekt med ønsket navn:

```
Servo myservo;
```

Vi har valgt å kalle det *myservo*. Settes før *setup()*-funksjonen



3. Gi beskjed om hvilken port som skal brukes for denne servoen:

```
myservo.attach(13);
```

Vi har valgt å bruke port D13. Denne kommandoen legges inn i setup()-funksjonen

4. Fortell så hvilket antall grader servoen skal stilles inn på 0 – 180

```
myservo.write(pos);
```

pos er et heltall fra 0 – 180. Denne kommandoen legges inn i loop()-funksjonen etter behov.

NB! Løsne servoen før dere er trygge på at armen ikke slår ned i gulvet. Ev. vent med å montere selve bommen til etter testingen. Dermed sikrer en seg at bommen monteres riktig på akslingen.

4.8 Oppdrag 5 – Kombiner den optiske portalen, telling av passeringer og servoen

Vi skal nå bygge ut funksjonen til programmet vårt ved å forsøke å sammenstille flere funksjoner:

Oppdrag 5: *Kombiner program 3a med servoen 4b slik at bommen går opp hver gang noen passerer den optiske porten samtidig som antallet passeringer telles og vises i displayet*

4.8.1 Oppkobling

Til dette oppdraget trengs ingen ny oppkobling dersom de tidligere er på plass

4.8.2 Programmering

Hent gjerne opp en ny skisse og lagre den under navnet: *DisplayAntallPasseringerServo-5*

Tips:

- Vi har valgt å definere to variabler for den justerte vinkelen for lukket og åpen bom
- Det anbefales å legge inn et lite delay på 100 ms etter at man har brukt kommandoen:

```
myservo.write(posAapen); // Åpne bommen, posApen angir pos i grader
delay(100);                // Følges av et kort delay
```

Dette er selvfølgelig ikke nødvendig dersom det likevel følger et annet langt delay.
- Det anbefales at man skriver programmet slik at bommen blir stående åpen så lenge en “person” befinner seg i den optiske portalen, dvs. skygger for lyset. For deretter å legge inn et ekstra delay etter at “personen” ikke skygger lengre.

Sjekk også at telleren fortsatt fungerer som den skal.

4.9 Oppdrag 6 – Styling av lys ved hjelp av rele

En av hensikten til portalen er å kunne styre et rele som tenner lysene i lokalet. Siden det her er snakk om styling av 230 V så må det benyttes et rele. Oppdraget denne gangen er relativt enkelt:

Oppdrag 6: *Slå på lyset i når det er personer i rommet og slå det av når det er tomt.*

La oss først se hvordan et rele fungerer.

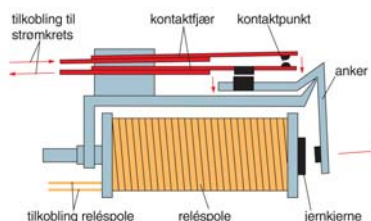


4.9.1 Releets funksjon

Det sier seg selv at en mikrokontroller ikke uten videre kan styre store strømmer eller spenninger siden de opererer med spenninger fra typisk 0 – 5,0 V. Vi trenger derfor en komponent som lar seg styre med lave spenninger og strømmer, men som selv kan styre store strømmer og spennin-ger. Vi skal se nærmere på releer som nettopp har denne egenskapen.

Releets funksjon

Figuren til høyre¹³ viser hvordan et tradisjonelt rele fun-gerer. Spolen til en elektromagnet er koblet til lavspenningsdelen til f.eks. en mikrokontroller. Når spo-len får strøm fra kontrolleren vil *ankeret* trekkes til og en bryter legges over og slutter en høyspenningskrets. Som vi ser er det en rent elektromagnetisk-mekanisk bryter.



Det som ikke er så uvanlig når det gjelder tilkobling av releer til f.eks. Arduino er at selv strømmen som kreves for å dra ankeret i posisjon er større enn det en Arduino eller ESP32 normalt klarer. En Arduino kan normalt lever fra 20 – 40 mA (det samme gjelder en ESP32, maks 40 mA), mens et rele kan-skje trenger en noe større strøm. I så fall er det vanlig å bruke en liten transistordriver foran releet. For mer informasjon om transistordrivere se avsnitt 5.7.2, side 100.

I vårt tilfelle vil vi bruke et rele med innebygget driver.

Det leveres en rekke ulike relekort med fra 1 til 8 releer som kan styres direkte fra en Arduino siden kortet inneholder drivere. Driverstrømmen er 0,6 mA og en spenning på 3.3V synes tilstrekkelig.

NB! Vær oppmerksom på at releet er aktivt lavt. Det betyr at releet slår inn når styresignalet legges til jord og releet slås av når styresignalet er høyt.

På figuren under er det vist noen eksempler. Disse er utstyrt med opto-kobler på inngangen slik at lav- og høyspenningsdelen er galvanisk skilt fra hverandre. Releene tåler 10A, 230V AC, og 10A, 30V DC.

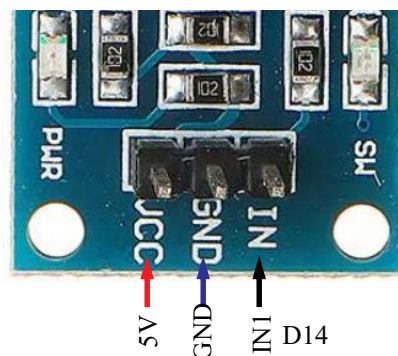


Det finnes en mengde slike varianter om man leter.

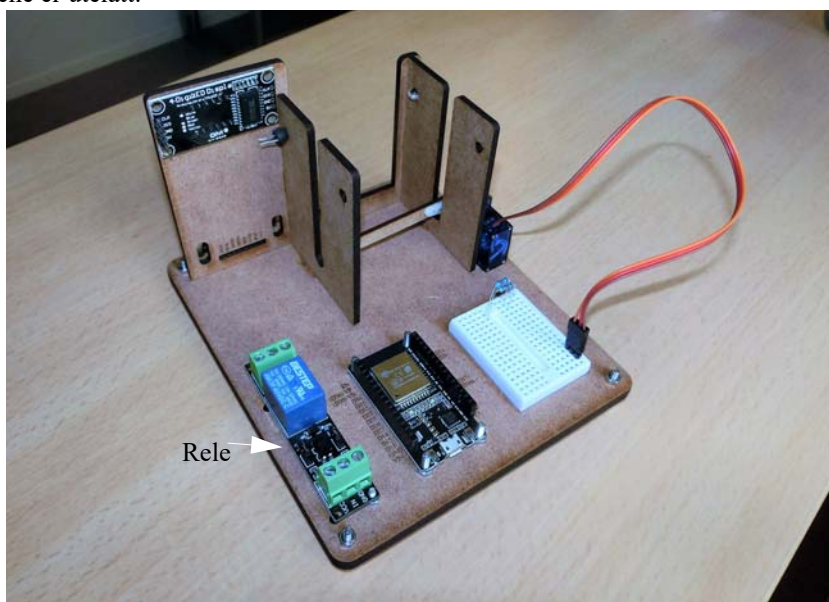
¹³.Store norske leksikon

4.9.2 Oppkobling

Oppkoblingen er særdeles enkel. Normalt trenger man kun å koble til jord og en av utgangene fra ESP32 eller Arduino. I vår eksempel skal kortet i tillegg ha 5V. Vi velger å koble IN1 til port D14 på ESP32. Vi har valgt et 5Vs rele for å kunne la det belaste 5V framfor 3.3V spenningen. Dermed unngår vi å belaste den interne spenningsregulatoren.



Bildet under viser monteringen av releet. Ledningene er utelatt.

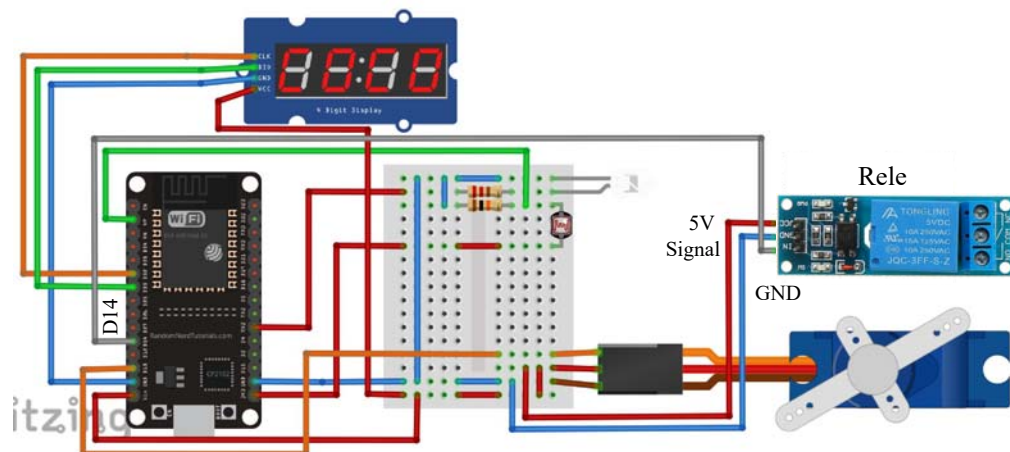


Det releet som til slutt ble valgt er litt annerledes enn det som er vist på bildet. Det er noe kortere og forsynt med en stiftlist for tilkobling av spenning og styresignal, noe som gjør det lett å koble opp. Se bilde under og bildet øverst på siden.





Figuren under viser oppkoblingen til og med oppkobling med rele.



4.9.3 Programmet

Oppdrag 6: *Slå på lyset i når det er personer i rommet og slå det av når det er tomt.*

Som nevnt er oppdraget denne gangen ganske enkelt, vi skal slå på lyset når det er personer i rommet og slå det av når det er tomt.

Tips:

- Bruk en `if()`-funksjon for å undersøke om rommet er tomt eller om det er noen der. Bruk antallet som betingelse.
- Styrepinnen til releet (IN1) må legges lav for at rele skal skifte tilstand fra passiv til aktiv (Under uttestingen har vi brukt et kretskort med fire releer derfor IN1 – 4).

Last opp og sjekk at programmet fungerer som tiltenkt.

4.10 Oppdrag 7 – Retningssensitiv optisk portal

Hittil har vi bare registrert passeringer og talt disse uansett om personene har gått inn eller ut. Der- som vi skal nå vårt endelige mål om at slusa både skal både kunne telle opp og ned avhengig av hvilken vei personene går, så må vi lage en mer intelligent sluse.

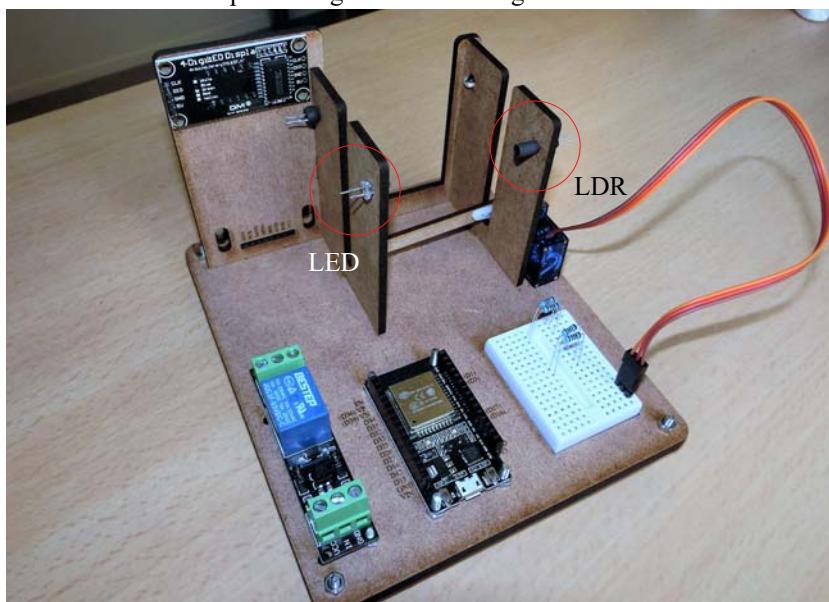
Oppdrag 7 *Det skal lages en optisk sluse som registrerer retningen til de som passerer. Når noen går inn skal antallet økes og når noen går ut skal antallet reduseres.*

For å klare dette så har vi laget to optiske portaler, en på hver side av bommen. Bommen skal åpne enten vi kommer innefra eller utenfra. Foreløpig tenker vi oss kun enkle situasjoner hvor en som går inn fra den ene eller den andre siden går helt gjennom uten å stoppe opp inne i slusa.

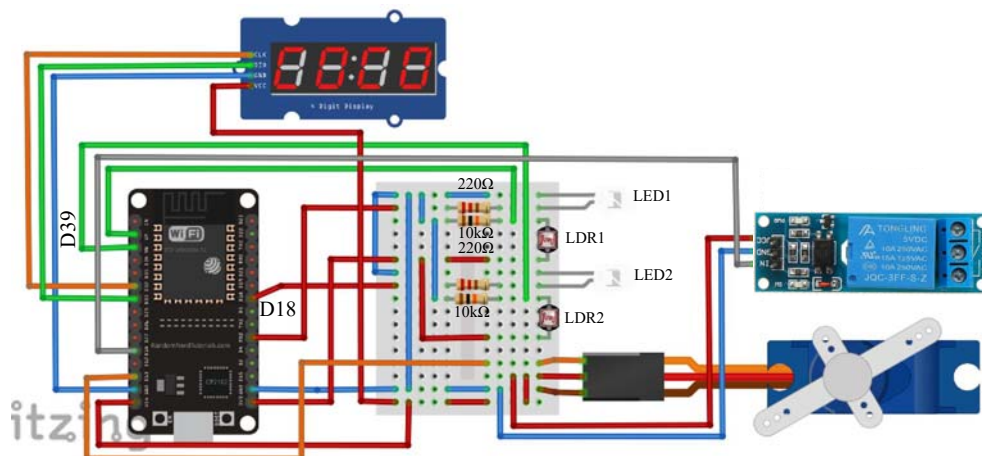
4.10.1 Oppkobling

Denne gangen må vi koble opp en ny optisk portal maken til den første på innsiden av bommen. Vi må huske på at personer enten kommer innenfra eller utenfra og bommen skal rekke å gå opp uansett fra hvilken kant personene kommer fra.

Bildet under viser monteringen av en ekstra optisk portal idet man har passert bommen. Ytterligere to motstander er montert på koblingsbrettet. Ledningene er utelatt.



Forslag til oppkobling er vist på figuren under.





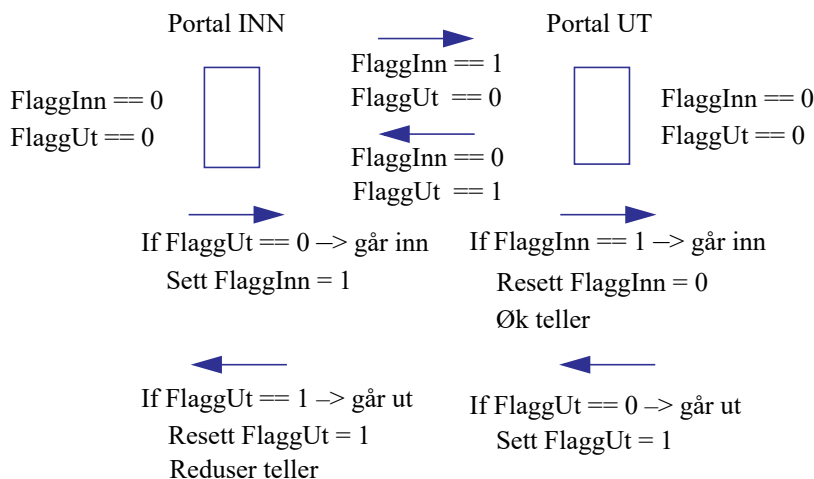
4.10.2 Programmet

Oppdrag 7 Det skal lages en optisk sluse som registrerer retningen til de som passerer. Når noen går inn skal antallet økes og når noen går ut skal antallet reduseres.

Åpne en ny skisse og kall den: *AvansertOptiskPortal-7*

Tips:

- Lag et program som *kun* registrerer passeringer i hver retning, ta gjerne utgangspunkt i det første programmet dere skrev *OptiskPortal-1*
- Skriv ut antallet til monitoren i første omgang. De øvrige funksjonene skal vi inkludere i senere oppgaver.
- Vi kan bruke variabler som *husker* om en tidligere portal er passert. Slike kaller vi gjerne *flagg*¹⁴.
- La oss lage en tegning for lettere å forstå tankegangen. Under har vi vist hvordan tilstanden til flaggene er etter som vi passerer ut og inn av portalen.



Det er en ting vi ikke har tenkt på her og det er at når vi går ut gjennom Portal INN så må vi ikke slå på FlaggInn . Likeså må vi ikke slå på Flagg UT når vi passerer PortalUT på vei inn, da blir det rot i systemet vårt.

Her gjelder det å holde “tunga i rett munn” skal man komme i mål. Det er imidlertid denne typen problematikk en ofte kommer opp i når man programmerer og som er det jeg vil kalle typisk “*algoritmisk tenkning*” og som en gjerne ønsker å trene på.

Tenk gjennom situasjonen og se om dere klarer å finne en løsning.

¹⁴. Å bruke betegnelsen flagg kommer sannsynligvis fra den amerikanske tradisjonen av å “heise” et lite flagg ved postkassen når det er lagt ny post i kassen. Det er for å indikere at en hending har inntruffet.



Tilstandsmaskin

Normalt vil en programmere en slik situasjon som en tilstandsmaskin. Man definerer da alle mulige tilstander en kan befinne seg i og alle hendinger som kan inntreffe. Dernest lager man et *tilstandsdiagram* som forbinder tilstandene med piler som indikerer hvilken ny tilstand man skal bevege seg til ut fra hvilken hending som inntreffer.

Tenk gjennom situasjonen og se om dere klarer å sette opp et tilstandsdiagram.

4.11 Oppdrag 8 – Suppler avansert optisk sluse med bom

Oppdrag 8 *Kombiner den optiske porten med servoen og bommen slik at den går opp når noen er på vei inn eller ut, og lukker når de har passert.*

4.11.1 Oppkobling

Det trengs ingen ny oppkobling for å løse dette oppdraget.

4.11.2 Programmet

For å løse dette oppdraget kombineres programmet knyttet til den avanserte optiske portalen med nødvendig kode fra programmet *ServoTest-4* slik at bommen går opp når noen går ut og tilsvarende når noen går inn. Ellers skal døren være stengt.

Tips:

- Det anbefales at man tar utgangspunkt i programmet *AvansertOptiskPortal-7* og inkluderer kode fra f.eks. *ServoTest-4*. Samtidig som man lagrer det nye programmet under et nytt navn. F.eks. *AvansertOptiskPortalServo-8*.
- Pass på at bommen går opp i “god tid” før den som skal ut kommer til bommen.

Test programmet i etterkant og se om det oppfyller alle betingelser. Merk dere ting som kunne ha fungert bedre.

4.12 Oppdrag 9 – Suppler avansert optisk sluse og bom med visning av antall

Oppdrag 9 *Kombiner den optiske porten, bommen og telleren med displayet, og vis antallet i rommet til en hver tid.*

4.12.1 Oppkobling

Det trengs ingen ny oppkobling for å løse dette oppdraget.



4.12.2 Programmet

For å løse dette oppdraget kombineres programmet knyttet til den avanserte optiske portalen og bommen, *AvansertOptiskPortalServo-8*, med nødvendig kode fra programmet *DisplayTeller-2* slik at telleren på utsiden av rommet oppdateres kontinuerlig med antallet som befinner seg i rommet.

Tips:

- Det foreslås at man lagrer det kombinerte programmet under navnet: *AvansertOptiskPortal-ServoDisplay-9*
- En må ta stilling til når det oppdaterte antallet skal vises. Når vedkommende er midt i slusa eller når vedkommende er vel inne eller ute.

Test programmet og vurder om det fungerer tilfredsstillende.

4.13 Oppdrag 10 – Suppler avansert optisk sluse, bom og visning av antall med et rele

Oppdrag 10 *Kombiner den optiske porten med bommen og telleren med displayet og releet som slår på lyset når noen er i rommet og ellers har lyset slukket.*

4.13.1 Oppkobling

Det trengs ingen ny oppkobling for å løse dette oppdraget.

4.13.2 Programmet

For å løse dette oppdraget kombineres programmet knyttet til den avanserte optiske portalen, bommen og visningen av antall på displayet, *AvansertOptiskPortalServoDisplay-9*, med nødvendig kode fra programmet *DisplayAntallPasseringerServoRele-6* slik at releet slår på lyset når når førstemann ankommer og siste mann har gått.

Tips:

- Sørg for å lagre det kombinerte programmet under et eget navn, f.eks. *AvansertOptiskPortal-ServoDisplayRele-10*

Test programmet slik at det fungerer etter hensikten

4.14 Oppdrag 11 – Program som leser RFID-kort og viser ID'en i monitoren

Oppdrag 11 *Det skal lages et program som leser av RFID-kortleseren og viser ID-koden i monitoren. Denne koden skal senere legges inn som en kode som har adgang slik at bommen åpnes når ID-kortet forevises leseren.*

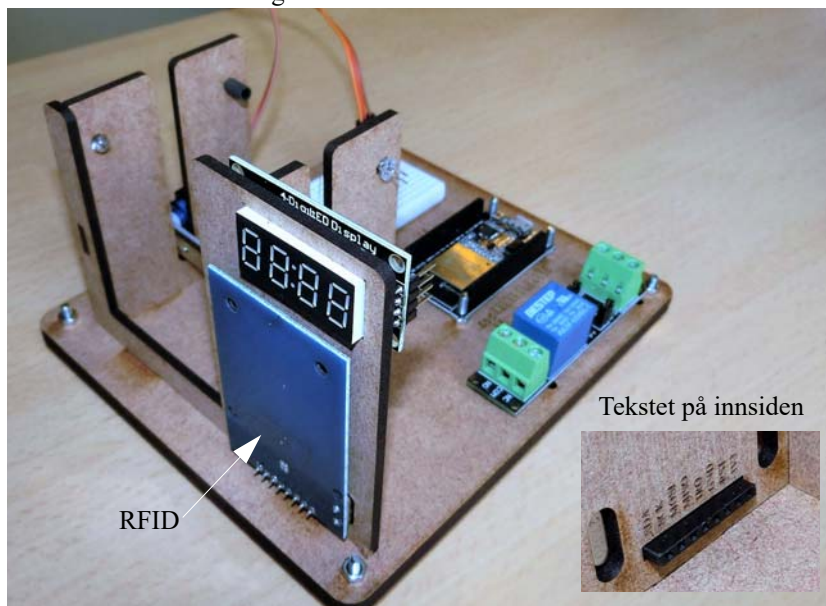
La oss først se litt på hvordan RFID-brikken og leseren fungerer.



La oss se hvordan vi kan koble opp kortleseren til vår ESP32

4.14.2 Oppkobling

Bildet under viser monteringen av RFID RC522. Legg merke til at kontaktpunktene er navngitt på veggen over kontakten. Ledningene er utelatt.

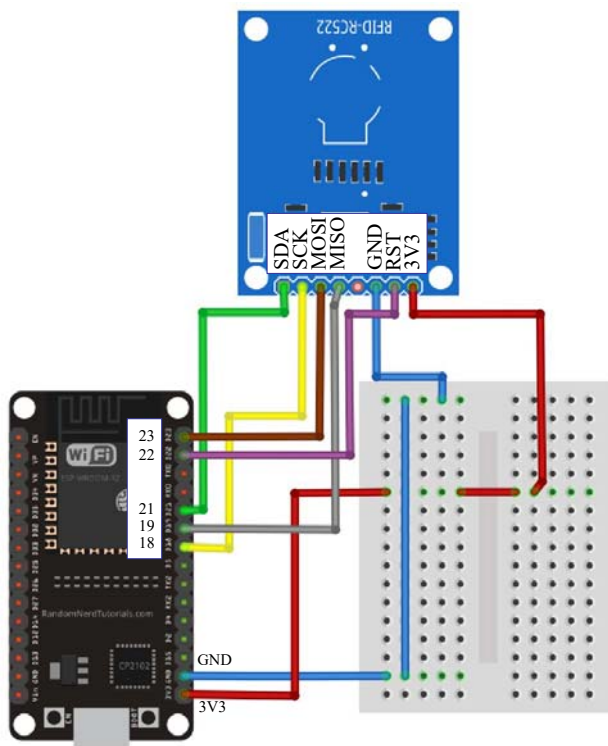


Kortet har en hylselist nederst som smettes inn i en spalt slik at det blir stående til høyre for den optiske slusen. For lettere å kunne koble til på baksiden av veggen viser vi her terminalene med betegnelser.

Figuren under viser oppkoblingen. Vi velger å koble signallinjene direkte fra ESP32, spenning og jord (GND) henter vi fra koblingsbrettet.



I figuren har vi valgt å fjerne alle andre oppkoblinger for å få mer oversikt over oppkoblingen.



4.14.3 Programmet

La oss repetere hva oppdraget går ut på:

Oppdrag 11 *Det skal lages et program som leser av RFID-kortleseren og viser ID-koden i monitoren. Denne koden skal senere legges inn som en kode som gir adgang slik at bommen åpnes når ID-kortet holdes bort til leserenheten.*

Får å kunne kommunisere med RFID-kortet må vi bruke to biblioteker:

- SPI.h (ligger som standard i IDE)
- MFRC522.h

SPI.h brukes for å kommunisere med RFID-leseren og finnes alt installert som standard.

MFRC522-biblioteket må vi installere om det ikke alt er gjort. Biblioteket kan blant annet hentes fra: <https://www.arduino-libraries.info/libraries/mfrc522> eller velg *Sketch* → *Include library* →



Library manage og skriv *MFRC522* i søkefeltet. Da vil følgende komme opp:



Trykk på **INSTALL** nederst i høyre hjørne og biblioteket installeres.

Eksempelprogram

Vi har valgt å vise dere et eksempelprogram som dere kan laste ned fra <https://www.ntnu.no/skolab/bla-hefteserie> under fanen: *Grunnkurs programmering ESP32 – dag 2: Lag en adgangskontroll*. Det leser kortinformasjon til nærliggende RFID-kort og presenterer ID-koden for de ulike kortene som holdes nært leseren i monitoren. Koden består av 4 byte som tolkes som fire tall. Disse kan så legges inn i programmet. Når kortet som er registrert i koden, gjenkjennes, kan en gi adgang til f.eks. et rom.

Eksempelkode:

```
/* Programmet leser informasjon fra RFID-briken og skriver ut til monitoren
 * Programmet er skrevet av Kåre-Benjamin Rørvik 08.07.20
 * Programmet er videreutviklet av Nils Kr. Rossing 17.07.20
 */

#include <SPI.h>
#include <MFRC522.h>

const int RST_PIN = 22; // Reset
const int SS_PIN = 21; // Slave

MFRC522 mfrc522(SS_PIN, RST_PIN); // Opprett instans mfrc522 av typen MFRC522

int ID0 = 0; // Erstatt 0 med avlest byte 0
int ID1 = 0; // Erstatt 0 med avlest byte 1
int ID2 = 0; // Erstatt 0 med avlest byte 2
int ID3 = 0; // Erstatt 0 med avlest byte 3
int RFIDflagg = 0; // Flagget som indikerer at nytt kort er lest

void setup() {
  Serial.begin(115200); // Starter seriell kommunikasjon med Baud rate
                        // på 115200 bit/sekund (husk å endre i terminalen).
  SPI.begin(); // Starter SPI - Initierer SPI-buss som kommuniserer mot RFID RC522
  mfrc522.PCD_Init(); // Starter RFID leseren.
  Serial.println("***Klar til å lese kort***");
}

void loop()
{
```



```
// Identifiser kort og les
if (mfr522.PICC_IsNewCardPresent() && mfr522.PICC_ReadCardSerial())
{
    // Skriv ut kortinformasjon
    Serial.print("Identifisert kort med ID: ");
    Serial.print(mfr522.uid.uidByte[0]); // Skriv ut ID byte 0
    Serial.print(" ");
    Serial.print(mfr522.uid.uidByte[1]); // Skriv ut ID byte 1
    Serial.print(" ");
    Serial.print(mfr522.uid.uidByte[2]); // Skriv ut ID byte 2
    Serial.print(" ");
    Serial.print(mfr522.uid.uidByte[3]); // Skriv ut ID byte 3
    Serial.println(" ");

    mfr522.PICC_HaltA(); // Stopp dersom kortet er uendret
    RFIDflagg = 1;      // Sett et flagg som indikerer at nytt kort er lest
}

// Sjekk at kortet er godkjent og ønsk velkommen inn
if((mfr522.uid.uidByte[0]==ID0) &&
    (mfr522.uid.uidByte[1]==ID1) &&
    (mfr522.uid.uidByte[2]==ID2) &&
    (mfr522.uid.uidByte[3]==ID3) && RFIDflagg == 1)
{
    Serial.println("Velkommen inn");
}
RFIDflagg = 0; // Reset flagg som forteller at kortet er sjekket
}
```

Forklaring av eksempel-programmet:

1. Inkluder bibliotekene i fila ved hjelp av følgende:
#include <SPI.h>
#include <MFRC522.h>
Plasseres i starten av fila
2. Derne må vi fortelle hvilke porter vi ønsker å bruke for å kommunisere med RFID-leseren.
Disse port-numrene defineres som heltallskonstanter:
const int RST_PIN = 22; // Reset
const int SS_PIN = 21; // Slave
Plasseres i starten av fila etter at bibliotekene er inkludert.
3. Så må gi kortleser et navn (deklarerer en instans mfr522 av typen MFRC522)
MFRC522 mfr522(SS_PIN, RST_PIN);
Deklarasjonen legges også i starten av fila før setup()-funksjonen.

For enkelhets skyld har vi i dette eksempel-programmet definert fire variabler som tar vare på de fire bytene i ID-koden:

```
int ID0 = 0; // Erstatt 0 med avlest byte 0
```



```
int ID1 = 0;      // Erstatt 0 med avlest byte 1
int ID2 = 0;      // Erstatt 0 med avlest byte 2
int ID3 = 0;      // Erstatt 0 med avlest byte 3
```

Disse settes i utgangspunktet til 0, helt til vi har funnet koden til det kortet vi ønsker skal gis tilgang.

4. Deretter initialiseres seriekommunikasjonslinjene (SPI) og RFID-kortet med følgende kommandoer:

```
SPI.begin(); // Starter SPI
mfrc522.PCD_Init(); // Starter RFID leseren.
```

Dette gjøres kun en gang og kan derfor legges i setup()-funksjonen

5. Vi ønsker å skrive ut informasjonen fra det leste RFID-kortet til monitoren. Erfaringer har vist at denne fungerer best med relativt høy datahastighet til PC'en og monitoren. Vi setter derfor denne til:

```
Serial.begin(115200);
```

Denne kommandoen legges også i setup()-funksjonen. Dersom vi skal se informasjonen i monitoren, må vi huske å endre hastigheten tilsvarende i monitoren.

6. Det første programmet må gjøre er å sjekke om det er noe kort som befinner seg i nærheten av kortleseren (mfrc522.PICC_IsNewCardPresent()) for deretter å lese innholdet korrekt (mfrc522.PICC_ReadCardSerial()). Dersom dette går bra så skriver vi ut de fire ID-bytene:

```
if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial())
{
    // Skriv ut kortinformasjon
    Serial.print("Identifisert kort med ID: ");
    Serial.print(mfrc522.uid.uidByte[0]); // Skriv ut ID byte 0
    Serial.print(" "); // Skriv ut mellomromt
    Serial.print(mfrc522.uid.uidByte[1]); // Skriv ut ID byte 1
    Serial.print(" "); // Skriv ut mellomromt
    Serial.print(mfrc522.uid.uidByte[2]); // Skriv ut ID byte 2
    Serial.print(" "); // Skriv ut mellomromt
    Serial.print(mfrc522.uid.uidByte[3]); // Skriv ut ID byte 3
    Serial.println(" "); // Skriv ut mellomromt

    mfrc522.PICC_HaltA(); // Stopp dersom kortet er uendret
    RFIDflagg = 1;        // Sett et flagg som indikerer at nytt kort er lest
}
```

Funksjonen sørger for at programmet stopper helt til kortet er fjernet fra kortleseren (mfrc522.PICC_HaltA();). Vi setter et flagg (RFIDflagg = 1;) for å indikere at ID-kortet er lest. Dette bruker vi i neste omgang for å fortelle at nå er et nytt kort lest og koden er klar til å sammenligne med ID-koden som har fått tilgang.

Den siste sekvensen sjekker om koden er godkjent slik at det kan gis adgang, her foreløpig symbolisert ved "Velkommen inn":

```
// Sjekk at kortet er godkjent og ønsk velkommen inn
```




```

if((mfr522.uid.uidByte[0]==ID0) &&
    (mfr522.uid.uidByte[1]==ID1) &&
    (mfr522.uid.uidByte[2]==ID2) &&
    (mfr522.uid.uidByte[3]==ID3) && RFIDflagg == 1)
{
    Serial.println("Velkommen inn");
}
RFIDflagg = 0; // Resett flagg som forteller at kortet er sjekket

```

I betingelsen til if()-setningen sammenligner vi hvert av de avlese bytene med de fire som har adgang: ID0, ID1, ID2 og ID3

Kjør eksempel-programmet

1. Første gang vi kjører eksempel-programmet skrives ID-koden ut som fire tall;

```
Identifisert kort med ID: 124 17 94 116
```

I dette tilfellet er koden 124, 17, 94, 116

Denne koden legges inn i starten av programmet der ID0, ID1, ID2 og ID3 er deklarert:

```

int ID0 = 124;      // Erstatt 0 med avlest byte 0
int ID1 = 17;       // Erstatt 0 med avlest byte 1
int ID2 = 94;       // Erstatt 0 med avlest byte 2
int ID3 = 116;      // Erstatt 0 med avlest byte 3

```

Her har vi erstattet 0'ene med de avleste kortdataene.

2. Laster vi opp programmet med disse dataene vil vi om vi holder kortet bort til RFID-leseren motta følgende beskjed:

```

Identifisert kort med ID: 124 17 94 116
Velkommen inn

```

Vi er nå klare til å videreutvikle dette programmet:

4.15 Program som gir tilgang for akkrediterte og åpner bommen

Oppdrag 12: *Kombiner RFID med åpning og lukking av bommen. Når et akkreditert kort nærmer seg RFID-leseren så åpner bommen. Sørg for at bommen åpnes når kortet godkjennes og lukkes etter 5 sekunder.*

I denne oppgaven skal vi kombinere avlesning av kortet, sjekking av at kortet er akkreditert og åpne bommen om dette er tilfelle. Dersom det ikke er tilfellet skal ikke bommen åpnes.

4.15.1 Oppkobling

Det kreves ingen ekstra oppkobling for å løse dette oppdraget.



4.15.2 Programmet

Sekvensen i dette programmet kan være omtrent som følger:

1. Les av kortet
2. Sjekk at ID-kode er akkreditert
3. Hvis ja, åpne bommen og lukk bommen etter 5 sekunder
4. Hvis nei, hold bommen lukket

Lagre programmet under navnet *RFIDTestBom-12*

4.15.3 Tilleggsoppgaver

Dette er jo et temmelig primitivt program for å gi adgang ved at programmet kun gir adgang til en person og at ID-dataene må legges inn i programmet. Her er noen oppgaver som elevene kan bryne seg på:

1. **Tilgang for mange**

Lag et program der det er mulig å legge inn et vilkårlig antall personideer som alle sjekkes ved akkreditering. Bommen åpnes når ID-koden finnes og holdes stengt når ID'en ikke er registrert

2. **Automatisert registrering av nye akkrediterte**

Lag et program hvor ID-data fra et kort som holdes opp til leseren lagres i en "database" dersom registreringen skjer idet man trykker ned en knapp. Ved senere vil vedkommende slippe inn ved framvisning av ID-kort. I første omgang kan det være tilstrekkelig at informasjonen "huskes" så lenge mikrokontrolleren har spenning, og "glemmes" ved strømbrudd.

3. **Automatisk registrering og varig lagring av nye akkrediterte**

Dette programmet fungerer som omtalt under punktet foran, men data lagres permanent i EEPROM i mikrokontrolleren slik at dataene huskes etter strømbrudd.

4.16 Oppdrag 13 – Endelig sammenstilling av hele systemet

I denne siste delen av prosjektet skal vi kombinere alle elementene slik at vi får oppfylt vår systemspesifikasjon:

Det komplette oppdraget: *Man ser for seg et rom med adgangskontroll. Adgangskontrollen skjer ved bruk av RFID og personlige adgangskort. Ved ankomst til døra inn til rommet registreres ID-kortet og adgang gis dersom vedkommende er akkreditert for rommet ved at låsen (bommen) går opp. I forbindelse med korona-utbruddet våren 2020 så settes det begrensninger til hvor mange som kan være i et rom samtidig. Det skal derfor lages en optisk port ved døra som teller antallet som til en hver tid befinner seg i rommet. Dette medfører at den optiske slusa må registrere både at noen går inn og at noen går ut. Ved inngangen er det plassert et display som til en hver tid viser antallet som befinner seg i rommet. Det stilles også*



krav til å redusere forbruket av elektrisk energi slik at når første man passerer inn gjennom slusa slås lyset i rommet på og når. Likeså skal lyset i rommet slås av og ventilasjonsanlegget til rommet skrus ned på minimum (vi nøyer oss med å slå inn et høyspenningsrele).

Deloppdraget kan formuleres slik:

Oppdrag 13 *Kombiner RFID med avansert optisk portal, bommen som går opp og ned og tenning av lys. Antallet i rommet skal til en hver tid være oppdatert. En foreslår at bommen går opp så snart RFID-kortet er godkjent og er åpen til at den optiske slusa er passert. Når man forlater rommet er det tilstrekkelig å gå inn i den innerste portalen for at bommen skal gå opp.*

4.16.1 Oppkobling

Nå skal hele systemet være oppkoblet, det skal derfor ikke være nødvendig med noen ytterligere kobling.

4.16.2 Programmet

Det burde nå kun stå igjen å inkludere RFID-leseren i det øvrige programmet og håpe at det ikke oppstår noen problemer i kombinasjonen. Opprett som vanlig en ny fil som du kan kalle *AdgangsKontrollTellingDisplay-13*.

Tips:

- For at det skal være lettere å få oversikt over strukturen i programmet anbefaler vi å plassere hver enkelt del av programmet i *funksjoner*. Se beskrivelsen under for hvordan man bygger opp funksjoner.
- Den optiske slusa skal fungere som tidligere mht. telling av de som går inn og ut. Men en godkjenning av ID-kortet skal være nok til at bommen går opp og venter på passering. Den kan så gå ned når vedkommende har passerte den innerste optiske portalen.
- Test programmet grundig og finn hvilke svakheter det har og ev. forbedringer.
- Ser dere noen anvendelser ut over den som er skissert i oppdraget?

Foreslått struktur ved bruk av funksjoner:

- Funksjon som leser ID-kort og godkjenner akkreditering
- Funksjon som teller passeringer gjennom slusa
- Funksjon som oppdaterer display

Vurder som det trengs å lages en egen funksjon for:

- Styring av bommen
- Styring av releet



Lag egne funksjoner

Funksjoner er en særdeles nyttig metode for å bygge opp strukturerte programmer. De fleste kommandoene som vi så langt har brukt er egentlig funksjoner som gjerne er definert i biblioteker. Dette gjør det enkelt å programmere og gir god oversikt.

Vi kan selv definere funksjoner med navn som vi kan velge slik at de gir mening. Vi kan i hovedsak tenke oss to typer funksjoner:

1. Funksjoner som bruker globale variabler

Disse bruker verdier som er overført via variabler som alle funksjoner i programmet har tilgang til. Disse funksjonene brukes hovedsakelig for å få en oversiktlig struktur, men også for å komprimere programmet.

2. Funksjoner som bruker lokale variabler

Når vi bruker disse overfører vi variabelverdiene idet vi kaller funksjonen. Ofte returnerer funksjonen også en verdi som et resultatet av beregninger funksjonen utfører med variabelverdiene vi overfører.

Et eksempel på en slik funksjon kan være:

```
double sinus_til_vinkelen  
sinus_til_vinkelen = sin(3.14); // Overfører ønsket vinkel i funksjonskallet
```

Funksjonen `sin(<argument>)` er definert i et bibliotek. Argumentet er vinkelen (i radianer) som vi ønsker å finne sinus til. Funksjonen beregner sinus og returnerer den til variabelen `sinus_til_vinkelen` som vi har deklart som en double (stort desimal tall)

Hvordan definere en egen funksjon?

Følgende regler må overholdes:

- Navnet må være sammenhengende og begynne med en bokstav (ikke tall eller tegn). Tror det er lurt å unngå norske bokstaver. Det må heller ikke komme i konflikt med eksisterende funksjoner med samme navn
- Funksjonen må defineres utenfor `setup()` og `loop()` som forøvrig begge er funksjoner. Det er ikke uvanlig å definere dem til slutt, men kan også gjøres i starten av programmet.

Vi ser først på en *Funksjon som bruker globale variabler*, dvs. en funksjon er uten argumenter i kallet og uten noen returverdi.

Eksempel: Funksjon som bruker globale variabler

Deklarering av funksjonen:

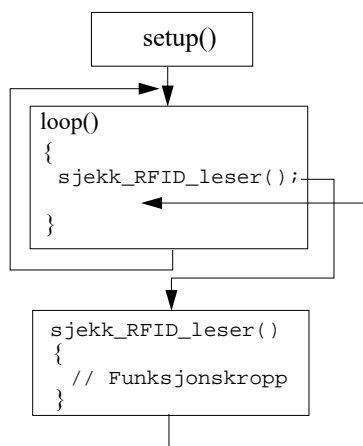
```
void sjekk_RFID_leser()  
{  
    // Her skriver vi koden som utfører sjekk av RFID, se eksempel-program  
}
```



Programmet har ingen argumenter (<tomt mellom parentesene>) og returnerer ingen ting `void`. (`void` betyr “tom”). I dette tilfellet må all informasjonen som funksjonen skal bruke overføres via globale variable som alle funksjonene er tilgang til.

Når vi har deklarert funksjonen så må den kalles på et passende tidspunkt fra hovedprogrammet, i `setup()` eller `loop()`. I nevnte eksempel vil kallet se slik ut:

```
loop()
{
    // Kode før kallet
    sjekk_RFID_leser(); // Kall av funksjonen
    // Kode etter kallet
}
```



Merk at vi må ha med parentesene og avslutter med “;”.

Når programmet kommer til funksjonskallet så hopper det til funksjonen, utfører den og hopper tilbake til hovedprogrammet på linja etter funksjonskallet som vist på figuren over.

Eksempel: Funksjon som bruker lokale variabler

Deklarering av funksjonen:

```
void skrivTilDisplay(int ant)
{
    // Her skriver vi koden som skriver til displayet
}
```

Her definerer vi en heltallsvariabel i argumentet til funksjonen. Det betyr at når vi kaller funksjonen så forventes en heltallsvariabel i argumentet når vi kaller funksjonen, i dette tilfellet antallet vi ønsker å skrive til displayet.

Argumentet i kallet må ha samme type (`int`) som

```
loop()
{
    int antall;
    // Kode før kallet
    skrivTilDisplay(antall); // Kall av funksjonen som skriver til displayet
    // Kode etter kallet
}
```

Vi har valgt ikke å returnere noen verdi i eksempelet vårt. En mulig retur kunne ha vært om skriveingen var vellykket (`retur = 1` (sann)) om den var vellykket og `retur 0` (usann) om den mislyktes).

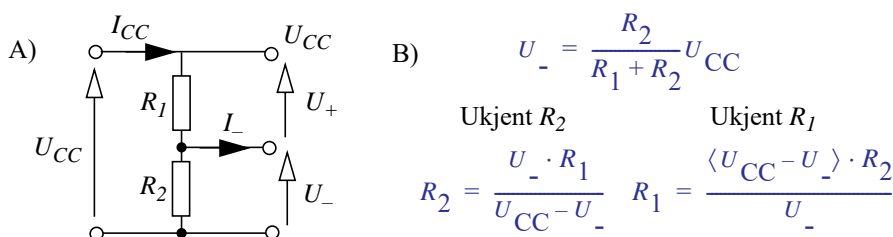


5 Noen aktuelle sensorer og aktuatorer

I dette kapitlet skal vi ta for oss noen av sensorene og aktuatorene som vi bruker i denne oppgaven. Men aller først skal vi se nærmere på spenningsdeleren som er en særdeles nyttig komponent når vi tar i bruk resistive sensorer slik som f.eks. lysfølsomme motstander som en LDR.

5.1 Spenningsdeleren

En spenningsdeler er nyttig for å etablere spenningsnivåer som normalt ikke er tilgjengelig fra strømforsyningen, men også for å konvertere en *variasjon i en motstandsverdi*, f.eks. i en LDR, til en *spenningsvariasjon* som mikrokontrollere kan registrere på en av sine analoge innganger.



Dersom vi har en spenning U_{CC} (f.eks. batterispenningen), så kan vi ved hjelp av to motstander ta ut en mindre del av denne spenningen. Tegning A i figuren over viser en enkel spenningsdeler. Spenningen U_- vil være en neddeling av spenningen U_{CC} , bestemt av motstandsverdiene til R_1 og R_2 .

Vi ønsker å finne sammenhengen mellom U_{CC} og U_- som funksjon av R_1 og R_2 .

Vi antar at $I_- = 0$, dvs. spenningsdeleren er ubelastet, hvilket er en god tilnærming dersom denne terminalen skal kobles til en av inngangene hos en mikrokontroller. Vi kan da sette opp en sammenheng mellom strømmer og spenninger i kretsen:

$$U_{CC} = (R_1 + R_2) I_{CC} \quad (5.1)$$

$$I_{CC} = \frac{U_{CC}}{(R_1 + R_2)} \quad (5.2)$$

Siden $I_- = 0$, vet vi at hele I_{CC} går gjennom R_2 . Vi kan da bruke Ohms-lov på R_2 og finner da et uttrykk for U_- som er spenningen av R_2 :

$$U_- = R_2 \cdot I_{CC} = \frac{R_2}{R_1 + R_2} U_{CC} \quad (5.3)$$

Tilsvarende kan vi finne U_+ som er spenningen over R_1 :



$$U_+ = R_1 \cdot I_{CC} = \frac{R_1}{R_1 + R_2} U_{CC} \quad (5.4)$$

Vi vet også at:

$$U_{CC} = U_- + U_+ \quad (5.5)$$

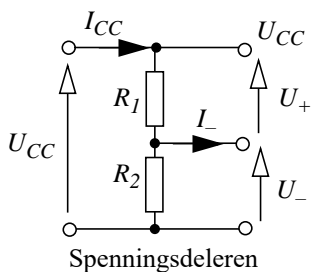
En vanligere situasjon er at vi kjenner spenningen U_- , men ønsker å bestemme verdien til en av motstandene. Dersom vi kjenner R_1 og U_- , men ønsker å bestemme R_2 , kan vi bruke følgende uttrykk:

$$R_2 = \frac{U_- \cdot R_1}{U_{CC} - U_-} \quad (5.6)$$

Dersom vi kjenner R_2 og U_- , men ønsker å bestemme R_1 , kan vi bruke følgende uttrykk:

$$R_1 = \frac{(U_{CC} - U_-) \cdot R_2}{U_-} \quad (5.7)$$

På tilsvarende måte kan vi bestemme resistansene når vi kjenner U_+ .



La oss se hvordan vi kan få en intuitiv forståelse av hvordan en spenningsdeler fungerer.

Fra Ohms lov vet vi at:

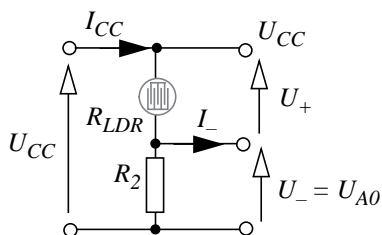
$$U_+ = R_1 \cdot I_{CC} \quad (5.8)$$

$$U_- = R_2 \cdot I_{CC} \quad (5.9)$$

Siden strømmen er den samme i begge motstandene, så kan vi dividere de to ligningene med hverandre og skrive:

$$\frac{U_+}{U_-} = \frac{R_1}{R_2} \quad (5.10)$$

Vi ser altså at forholdet mellom spenningene er forholdet mellom resistansene til de to motstandene.



Spenningsdeleren med LDR

I figuren til venstre har vi byttet ut R_1 med en resistiv sensor, i dette tilfellet en lysfølsom motstand

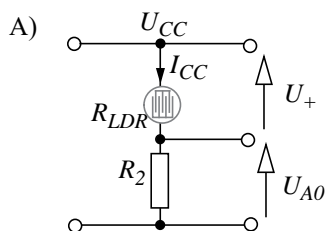
Vi tenker oss at U_- kobles til en analog inngang på mikrokontrolleren vi kaller den U_{A0} siden vi kobler den til den analoge inngangen $A0$.

Fra ligning 5.3 vet vi at forholdet mellom spenningen U_{A0} og motstandsverdien til LDR'en, R_{LDR} kan uttrykkes slik:

$$U_- = \frac{R_2}{R_{LDR} + R_2} U_{CC} \quad (5.11)$$

som vi ser så er det ikke nødvendigvis en lineær sammenheng. Men fra ligning 5.3 vet vi at vi kan regne oss tilbake å finne R_{LDR} med følgende omregning:

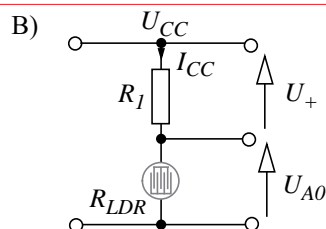
$$R_{LDR} = \frac{\langle U_{CC} - U_{A0} \rangle \cdot R_2}{U_{A0}} \quad (5.12)$$



Økende lysstyrke \rightarrow Økende U_{A0}

Ofte holder det at vi forstår rent kvalitativt hvordan spenningsdeleren fungerer da vi gjerne bruker LDR i en detektor.

Vi antar at *lysstyrken øker*. Det betyr at motstandsverdien til R_{LDR} blir mindre. Når R_{LDR} blir mindre, vil strømmen I_{CC} øke. Siden den samme strømmen går gjennom R_2 forteller Ohms lov oss at spenningen U_{A0} øker ($U_{A0} = I_{CC} R_2$).



Økende lysstyrke \rightarrow Redusert U_{A0}

Vi kan imidlertid tenke oss en alternativ plassering av LDR'en ved at den erstatter R_2 istedet for R_1 .

Dersom vi nå antar at *lysstyrken øker* og motstandsverdien til R_{LDR} blir mindre. Så vil strømmen I_{CC} øke som sist. Siden I_{CC} også går gjennom R_1 , forteller Ohms lov oss at spenningen U_+ også vil øke. Siden $U_{A0} = U_{CC} - U_+$ så vil U_{A0} avta.

Spenningsdeleren med LDR

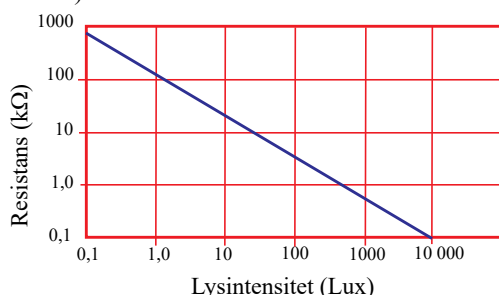
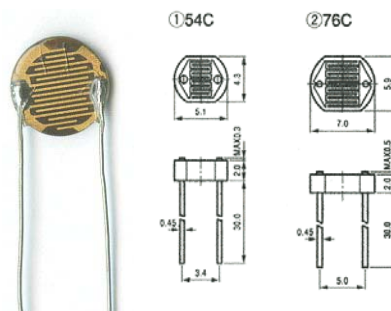
5.2 Lysfølsom sensor – LDR

Deteksjon av lys kan gjøres på mange ulike måter. I dette avsnittet skal vi se hvordan vi kan bruke LDR (Light Dependent Resistor) og fototransistorer som lysfølsomme komponenter.

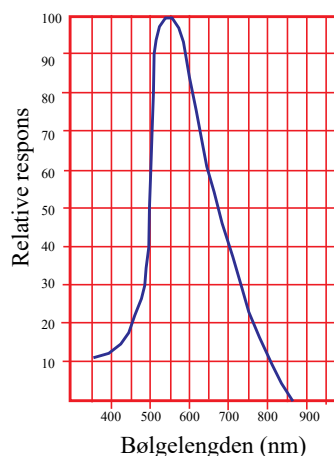
5.2.1 Fotomotstand (LDR - Light Dependent Resistor)

Grunnen til at vi velger å omtale fotomotstander er at den gjennom tidene har vært en gjenganger i mange elektronikkprosjekter og brukes i mange sammenhenger som f.eks. for automatisk tenning av gatebelysning, som enkle lysmålere, som automatisk tenning av frontlyktene på bil, innbruddsalarm ved at en lysstråle brytes o.l.

Fotomotstander har tradisjonelt vært laget av Cadmium-Sulfid (CdS) belagt med fingerelektroder som vist på figuren til høyre. I mørket vil stoffet CdS være omtrent isolerende og kan gi en motstand på over 1 M Ω . Belyses stoffet, kan resistansen i fotomotstanden falle til under 1 k Ω . Årsaken er at fotoner (lys) med tilstrekkelig energi, eksisterer elektroner fra valensbåndet til ledningsbåndet, hvor de kan bevege seg fritt og bidra til ladningstransporten. Effekten er imidlertid ikke like framtreddene for alle frekvenser. Til høyre på figuren under ser vi at materialet er spesielt følsomt for lys i det synlige området av spekteret nær 540 nm (nanometer, 10^{-9} m). Vi ser også (til venstre på figuren) at det er en omtrent lineær sammenheng mellom lysstyrken målt i lux og resistansen (begge skalaer er logaritmiske).



Resistans som funksjon av lysintensitet (venstre), følsomhet som funksjon av bølgelengde.



Lysfølsomme motstander er imidlertid relativt langsomme. En endring i lysstyrken på noen μ sek, kan gi en responstid på opp til 100 msek. hos fotomotstanden, men som i mange tilfeller er mer enn godt nok. Bruker man fotomotstander bør man også være klar over følgende [9]:

- Resistansen for fotomotstander kan variere mye fra eksemplar til eksemplar. Dette gjelder spesielt resistansen ved mørke.
- Resistansen hos en LDR vil også være temperaturavhengig
- En LDR kan også brenne opp ved for store strømmer. Dette er noe en bør være oppmerksom på dersom den skal brukes i sterkt sollys. Under slike forhold vil motstanden fall og strømme



øke. I slike tilfeller må man sørge for en tilstrekkelig høy seriemotstand slik at strømstyrken og dermed også effekttapet holdes under verdier som kan være skadelige for LDR'en.

- Dersom sensoren går i metning ved sterkt sollys, forsøk å redusere serieresistansen (men pass på effekttapet). Ønsker man større følsomhet i det mørke området øker man serieresistansen

For å konvertere endring i resistans til spenning, kan vi bruke en enkel spenningsdeler (se figuren under). Her trengs normalt ingen målebro eller forsterker for å registrere endring i resistans siden endringen er så stor.

Lysintensitet måles i lux. 1 lux er 1 lumen pr. m^2 .

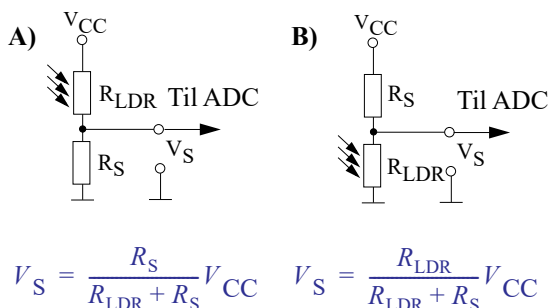
Dette tilsvarer:

- Fullt sollys 11 000 lux (eller ca. 1000 W/m^2)
- Sollyset en tidlig morgen 6 000 lux
- Belysningen i et TV-studio 1 000 lux
- Et godt opplyst kontor 400 lux
- Lyset fra en fullmåne 1 lux

Oppkobling mot ADC

Siden grensnittet til kontrolleren krever en spenning, kobles LDR-motstanden i serie med en motstand som vist i figuren til høyre. Velg verdien på seriemotstanden lik den typiske resistansen til fotomotstanden (LDR) i det aktuelle lysintensitetsområdet der fotomotstanden skal brukes.

Ønsker man maksimal spenningsvariasjon fra dypt mørke til sterkt lys kan seriemotstanden beregnes ut fra følgende ligning [9]:



$$R_S = \sqrt{R_{LDR(light)} \times R_{LDR(dark)}} \quad (5.13)$$

hvor $R_{LDR(light)}$ er resistansen i sterkt lys og $R_{LDR(dark)}$ er resistansen i mørke.

Spenningsnivået V_S beregnes fra formlene som antydnet på figuren. Legg merke til at oppkoblingen på tegning A gir økende spenning V_S med økende lysstyrke, mens oppkoblingen i tegning B gir fallende spenning med økende lysstyrke. For en utdypende diskusjon av spenningsdeleren se avsnitt 5.1.

Kalibrering:

Utfordringen blir å finne en omregningsformel fra lysstyrke til spenning:

1. Mål spenning som funksjon av lysstyrke (krever lysmåler)

2. Bruk regresjon for å finne et best tilpasset funksjonsuttrykk
3. Legg omregningsformelen inn i prosessoren

5.3 Trådløs identifikasjon – RFID¹⁶

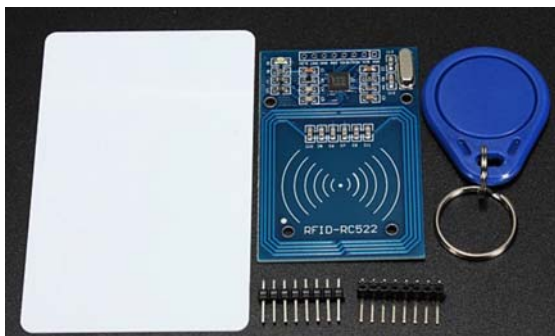
Vi definerer *Radio Frequency Identification* (RFID) som en sensor, og velger å beskrive MFRC522¹⁷ som har relativt kort rekkevidde (0–5 cm). Settet vi har benyttet består av et sender-mottaker-enhet (lese-enhet) og en ID-kort som er merket med et unikt nummer som kan avleses av lese-enheten.

ID-kortet inneholder en antenne som er avstemt til samme frekvens som antennen på ID-kortet og en liten microchip som har 1kbit lager for å holde informasjon. Siden ID-kortet ikke har noen energikilde, må energien som skal til for å drive microchipen overføres fra lese-enheten via antennene og til chipen. Denne tilføres dermed tilstrekkelig energi til å svare lese-enheten med sin unike identifikasjonskode. ID-kort og lese-enhet av den typen omtalt her kan både avlese ID-kortets kode og skrive ny informasjon til kortet.

Lese-enheten opererer på frekvensen 13,56MHz og kan kobles til Arduino'en via en I²C- eller SPI-buss. Den forsynes med en spenning på 3,3V. Kretsen kan overføre datarater på 424kbit/s på I²C-bussen og opp til 10Mbit/s på SPI-bussen. ISO14443A omtaler protokollen og beskriver datarammestrukturen, feildeteksjon og krypteringsmetode. Til tross for at forsyningsspenningen er 3,3 V, tåler lese-enheten 5V spenning på terminalene og kan derfor kobles direkte til en Arduino med 5V forsyningsspenning.

Organisering av dataene på ID-kortet:

La oss ganske kort se hvordan dataene er organisert på ID-kortet. Som vi tidligere har nevnt har ID-kortet 1kbit dataminne hvor vi kan lagre data som senere kan leses. Disse dataene er organisert i 16 *sektorer*, hver med fire *blokker* og hver blokk inneholder 16 *byte*, hvilket betyr at det totalt kan lagres $16 \times 4 \times 16 \text{ byte} = 1024 \text{ byte}$. Figuren til høyre viser de siste sektorene og blokkene.



COM6

Send

Firmware Version: 0x92 = v2.0

Scan PICC to see UID, SAK, type, and data blocks...

Card UID: 20 C3 93 5E

Card SAK: 08

PICC type: MIFARE 1Ks

Sector	Block	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	AccessBits
15	63	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	62	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	61	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	60	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
14	59	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	58	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	57	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	56	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
13	55	00	00	00	00	00	00	FF	07	80	69	FF	FF	FF	FF	FF	FF	[0 0 1]
	54	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	53	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]
	52	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	00	[0 0 0]

16. <https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/>

17. https://www.kultogbillig.no/index.php?_route_=RFID-RC522-RF-IC-Card-Sensor-Modul&search=RFID

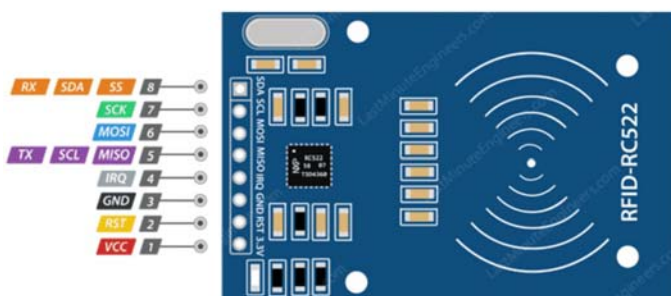


I sektor 0, blokk 0 finner vi en unik identifikasjonskode for det aktuelle ID-kortet som består av i alt 8 byte som vist på figuren til høyre. Denne koden går under betegnelsen *Manufacturer block* eller *Manufacturer data* og inneholder IC manufacturer data og en *Unik identifier* som kan brukes til å identifisere ID-kortet. En skal være forsiktige med å skrive over denne koden da det kan låse kortet slik at man senere ikke får tilgang.

Sector	Block	Data (Hex)	Data (ASCII)
0	0	20 C3 93 SE 2E 80 04 00	[0 0 0]
0	1	00 00 00 00 00 00 00 00	[0 0 0]
0	2	00 00 00 00 00 00 00 00	[0 0 0]
0	3	00 00 00 00 00 00 00 00	[0 0 0]
0	4	00 00 00 00 00 00 00 00	[0 0 0]
0	5	00 00 00 00 00 00 00 00	[0 0 0]
0	6	00 00 00 00 00 00 00 00	[0 0 0]
0	7	00 00 00 00 00 00 00 00	[0 0 0]
0	8	00 00 00 00 00 00 00 00	[0 0 0]

Dersom noen ønsker å lese ut innholdet i kortet finnes oppkoblingen og programvare på nettsiden til *Last Minute Engineers*: <https://lastminuteengineers.com/how-rfid-works-rc522-arduino-tutorial/> Her finner man også mye interessant stoff om RFID og MFRC522.

Figuren til høyre viser pinninngen til kortet MFRC522. Vi legger merke til I²C-bussen på pinne 5 (SCL) og 8 (SDA). Videre tilbyr kretsen et interruptsignal (IRQ) som vil gi et signal til Arduino'en dersom det kommer et ID-kort i nærheten av leseenheten. Dermed slipper Arduino'en å sjekke status for leseenheten, men kan avbryte sine gjøremål bare når det foreligger et interrupt.



Spenningsforsyningen (Vcc) skal være mellom 2,5 – 3,3 V. 5V vi sannsynligvis ødelegge kretsen. Strømtrekket er normalt på mellom 13 – 26 mA.

Reset (RST) vil, når denne pinnene er lav (GND), slå kretsen helt av, hvilket kan være fornuftig dersom man ønsker å spare strøm. Den vi i denne tilstanden ikke reagere på ID-kort. Ide spenninngen kommer på igjen så vil kortet resettes.

Dersom man ønsker større lese og skrive hastighet kan det være lurt å benytte SPI bussen som anvender fire linjer. Disse fire linjene kombineres imidlertid med de to andre databussene:

MISO / SCL / Tx pinnen fungerer som *Master-In-Slave-Out* (MISO) når SPI-bussen er aktiv, og som *serieklokke* (SCL) når vi benytter I²C-bussen og som *seriedata* utgang (Tx) når vi benytter UART-bussen.

MOSI (Master Out Slave In) fungerer som *datainnngang* når vi benytter SPI-bussen.

SCK (Serial Clock) fungerer som *serieklokkeinnngang* fra SPI-bussens master, f.eks. Arduinoe'en.

SS / SDA / Rx pinnen fungerer som inngangssignal når kretsen benytter SPI-bussen, eller som seriedata (SDA) når I²C-bussen benyttes, og fungerer som seriedata inn (Rx) når UART-bussen benyttes. Denne pinnen er også vanligvis markert med et kvadrat for å indikere at den kan brukes som referanse for de øvrige pinnene.

Oppkobling:

SPI-bussen er tilknyttet forskjellige porter hos de ulike Arduino-kortene. Tabellen under gir en oversikt over pinningen på tre vanlig brukte Arduino-kort:

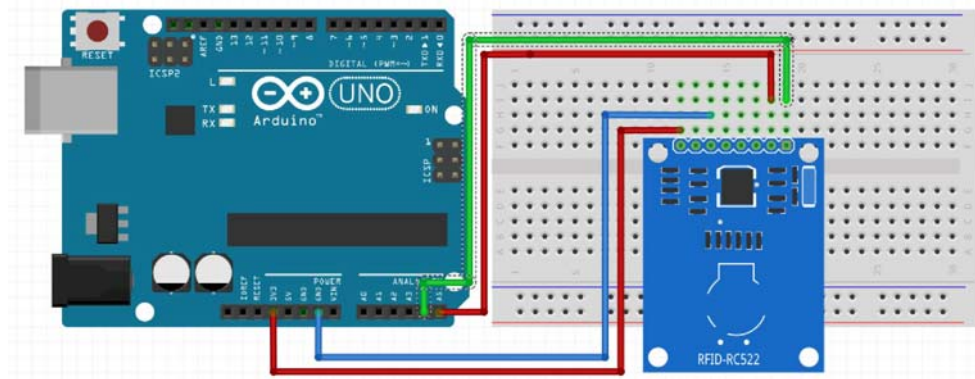
Arduino	MOSI	MISO	SCK	SS
UNO	11	12	13	10
NANO	11	12	13	10
MEGA	51	50	52	53

Figuren til høyre viser hvordan man kan koble opp MFRC522 til Arduino UNO hvor kommunikasjonen skjer ved hjelp av SPI-bussen.

Tilsvarene kan vi koble opp leseenheten ved bruk av I²C-bussen. Også i dette tilfellet er bussen tilknyttet ulike pinner på de forskjellige Arduino-kortene:

Arduino	SDA	SCL
UNO	A4	A5
NANO	A4	A5
MEGA	D20	D21

Figuren under viser hvordan RFID-RC522 kan kobles opp til Arduino UNO via I²C-bussen.





Programmering:

- **Installasjon av biblioteket:** Vi skal se hvordan vi kan kommunisere med kortet ved hjelp av SPI-bussen. For å få til dette på en enkel måte laster vi ned biblioteket: MFRC522.h fra f.eks.: <https://www.arduino-libraries.info/libraries/mfrc522> Biblioteket installeres på vanlig måte.

- **Legg inn nødvendige biblioteker i programmet:** Vi legger inn bibliotekene:

```
#include <SPI.h>
#include <MFRC522.h>
```

helt først i programfila

- **Deklarasjon:** Derneft deklarerer variabler som refererer til noen av portene i Arduinoen:

```
#define RST_PIN 5    // Configurable, see typical pin layout above
#define SS_1_PIN 10 // Configurable, take a unused pin, only HIGH/LOW required,
                    // different to SS 2

#define NR_OF_READERS 2

MFRC522 mfrc522[NR_OF_READERS]; // Create MFRC522 instance.
```

Vi deklarerer også instansen mfrc522[] av klassen MFRC522. Disse deklarasjonene legges inn før setup()-funksjonen.

- **Initialisering:** Så skal vi initialisere RFID kortet og SPI-bussen som gjøres i setup()-funksjonen

```
SPI.begin(); // Init SPI bus
mfrc522[0].PCD_Init(SS_1_PIN, RST_PIN); // Init the MFRC522 card
```

Vi legger merke til at her angir vi hvilke pinner vi bruker til *SS* og *RST*. I tillegg til å initialisere RFID-kortet må vi initialisere SPI-bussen.

- **Avlesning av ID-kort** og returner ID-nummer ved hjelp av følgende funksjon som kalles fra loop()-funksjonen med kallet: `ID = LesAvRFID();` Selve avlesningen skjer fra funksjonen:

```
int LesAvRFID()
{
    int R;
    ID = 1000;
    // Sjekk RFID-brikke
    if (mfrc522[0].PICC_IsNewCardPresent() && mfrc522[0].PICC_ReadCardSerial())
    {
        ID = mfrc522[0].uid.uidByte[1];
        mfrc522[0].PICC_HaltA(); // Halt PICC
    }
    return ID;
}
```

5.4 Lysdioder

Lysdioder finnes i svært mange varianter, både med hensyn til størrelse, utforming, farge og lysstyrke. I tillegg inneholder noen lysdioder flere enkeltdioder med ulike farger som kan styres individuelt.

5.4.1 Ordinære lysdioder¹⁸

Vi skal se litt nærmere på ordinære lysdioder som det finnes mange varianter av.

En lysdiode omdanner en elektrisk strøm til fotoner med ulik bølgelengde, farge. Fargen bestemmes av materialvalgene i halvlederen. Som navnet sier så er det en diode og en diode leder strøm den ene veien, hvilket også er tilfelle for en lysdiode (LED – Light Emitting Diode). Det betyr at for at lysdioden skal lyse så må den kobles rett vei i forhold til pluss og minus.

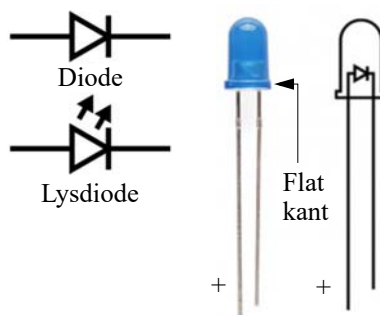
Polaritet: Lysdioder med bein har vanligvis et langt og et kort bein, hvor det lengste beinet normalt er den positive polen. Kanna (huset) til dioden har også gjerne en flat kant ved det negative beinet (kort). Dette kan det være nyttig å vite om dersom beina er klippet like lange etter bruk.

Effektforbruk og virkningsgrad: Virkningsgraden for en lyskilde kan måles i lm/W (lysstyrke i lumen pr. tilført elektrisk effekt i Watt). Normalt trekker en lysdiode langt mindre effekt enn en glødelampe ved samme lysintensitet. Den er derfor en mer effektiv lyskilde enn glødelampen og lysstoffrøret. Dette er selvfølgelig også grunnen til at LED i stadig større grad overtar for tradisjonelle lyskilder.

Strømstyrt: Det er en direkte sammenheng mellom strømtrekk og lysintensitet. Det betyr at ved å kontrollere strømstyrken kan vi regulere lysintensiteten. En lyssterk lysdiode vil naturlig nok også trekke mer strøm og tappe batteriet raskere. Det er også viktig at vi begrenser strømmen slik at den ikke brenner opp. For å regulere strømmen kan vi bruk en regulerbar strømkilde, men for laveffekts lysdioder brukes kun en motstand.

Små lysdioder trekker fra 10 – 50 mA, mens sterke lysdioder kan trekke fra 350 mA til flere Ampere. Strømmen i små lysdioder begrenses vanligvis med en motstand på fra 150 – 330 Ω . Mens kraftige lysdioder styres med konstante eller variable strømkilder. Vi skal komme tilbake til dette under behandling av lysdioder som lyskilder.

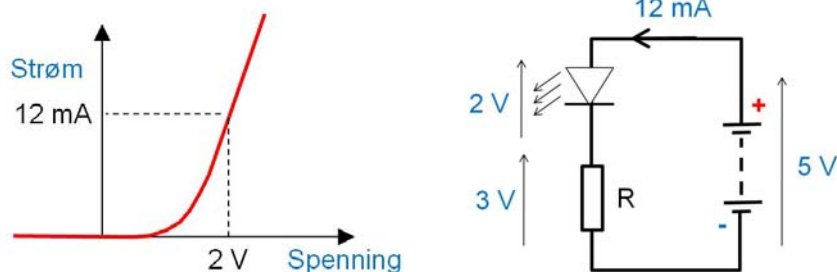
Beregning av strøm: La oss se på et enkelt regnestykke for å beregne seriemotstanden til en lysdiode brukt som indikator (ikke lyskilde). For denne typen lysdioder er det stort slingsringsmonn mht. valg av seriemotstand.



18. https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds?_ga=2.37374368.1867605279.1589209561-856163918.1476131557



Ofte kan en få oppgitt *foroverspenningen* over dioden ved en gitt strøm. Disse verdiene stammer fra diodekarakteristikken som viser sammenhengen mellom foroverspenning og strømmen i dioden. I vårt tilfelle er spenningen over dioden 2V når det går en strøm på 12 mA (til venstre på figuren under). Vi har også valgt en spenningskilde lik 5V som er typisk for Arduino.



Vi bruker Kirchhoffs spenningslov og ser at en strøm på 12 mA gir en spenning på 3V over motstanden. Når vi kjenner ønsket strøm og spenning for en motstand, kan vi beregne nødvendig motstandsverdi etter Ohms lov:

$$R = U / I = 3V / 12mA = 250\Omega \quad (5.14)$$

I dette tilfellet kan man sannsynligvis oppnå gode resultater med motstandsverdier fra 150 – 330 Ω .

Datablader: La oss ganske kort se på et eksempel på datablad for en lysdiode omtrent som den vi har omtalt foran.

Tabellen under viser noen interessante parametere, deriblant maksimalverdier.

ITEMS	Symbol	Absolute Maximum Rating	Unit
Forward Current	I_F	20	mA
Peak Forward Current	I_{FP}	30	mA
Suggestion Using Current	I_{su}	16-18	mA
Reverse Voltage ($V_R=5V$)	I_R	10	μA
Power Dissipation	P_D	105	mW
Operation Temperature	T_{OPR}	-40 ~ 85	$^{\circ}C$
Storage Temperature	T_{STG}	-40 ~ 100	$^{\circ}C$
Lead Soldering Temperature	T_{SOL}	Max. 260 $^{\circ}C$ for 3 Sec. Max. (3mm from the base of the epoxy bulb)	

Forward Current angir den maksimale strømstyrken man kan la gå i lysdioden over lengre tid.

Peak Forward Current er absolutt maksimal strøm som kan tillates i korte perioder (sekunder)

Suggestion Using Current er anbefalt område for strømmen.

Power Disipation er maksimalt avgitt effekt fra dioden som beregnes ut fra strømmen i dioden x spenningen over dioden. I vårt eksempel $12\text{ mA} \times 2V = 24\text{ mW}$.

Operation Temperature er anbefalt temperaturområde under vanlig drift. Blir dioden varmere enn 100 $^{\circ}C$ over noe tid, kan den lett gå i stykker.



Følgende tabell viser typiske verdier brukt under testing og kan være gode retningslinjer for valg av f.eks. foroverspenning:

ITEMS	Symbol	Test condition	Min.	Typ.	Max.	Unit
Forward Voltage	V_f	$I_f=20\text{mA}$	1.8	---	2.2	V
Wavelength (nm) or TC(k)	$\Delta \lambda$	$I_f=20\text{mA}$	620	---	625	nm
*Luminous intensity	I_v	$I_f=20\text{mA}$	150	---	200	mcd

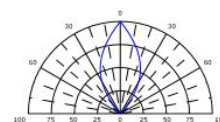
Forward Voltage angir typisk spenning over dioden når det går 20 mA. Dette er en viktig parameter når man skal beregne seriemotstanden.

Wavelength angir utstrålt bølgelengde i nanometer ved en strømstyrke lik 20 mA. 620 nm ligger i det røde området.

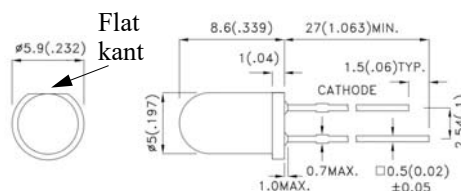
Luminous intensity angir utstrålt lysintensitet i millicandela ved 20 mA. 200 mcd er ingen kraftig lysdiode, men grei som indikatorlys.

For de fleste lysdioder angis også med hvilken åpningsvinkel lyset stråler ut og angis i intensitet i prosent av maksimalverdien, og som funksjon av vinkelen til side for senterlinjen 0° . Vi legger merke til at lysintensiteten er falt til under 50% sett 30° til siden for senterlinjen. Noen dioder stråler smalt andre bredt. Diagrammet over til høyre viser en relativt smal utstråling. Utstrålingsvinkelen er gjerne bestemt av hvordan toppen av plasthuset er utformet. En smal utstrålingsvinkel oppnås ved å lage toppen som en linse.

Viewing Angle Drawing



Utforming: Lysdioder leveres i mange forskjellige utforminger, men de vanligste er runde med en tverrsnittsdiameter på huset på 3, 5 eller 10 mm. 5 mm er utvilsomt det vanligste (se figuren til høyre). Legg spesielt merke til den flate kanten av det runde huset som angir den siden katoden (–) er plassert på. Men det finnes også lysdioder med andre former f.eks. rektangulære og overflatemonterte.

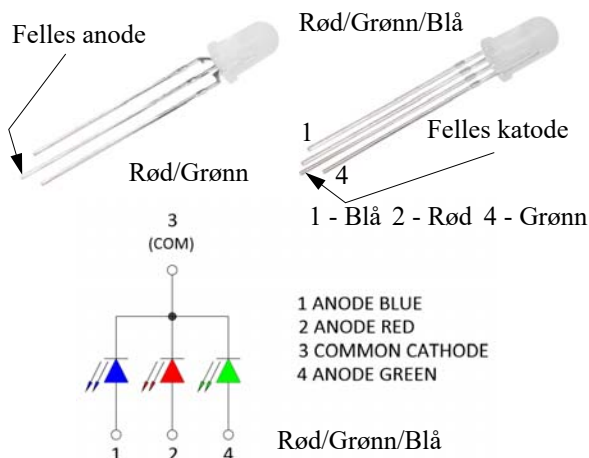




5.4.2 Flerfargede lysdioder – RGB-dioder

Flerfargede lysdioder kan også være utformet på ulikt vis. Men mange er av typen med 5mm runde hus, hvor det er montert to eller tre lysdioder inne i det samme huset. Disse kan styres individuelt ved hjelp av separate bein, gjerne med en felles katode eller anode. Det siste er verdt å merke seg da man kan bli ganske hjelpeløs dersom man forveksler disse to. Det vanligste er at de har felles katode. Vanligvis finnes de i kombinasjonene Grønn/Gul, Grønn/Rød og Rød/Grønn/Blå (RGB).

De er ofte utformet med diffust hus siden man ønsker å kunne blande farger. Erfaringer har imidlertid vist at blandingsfarger kan være vanskelig å få fram og blir ikke tydelig med mindre man betrakter dioden fra en spesiell kant. Et diffust hus vil imidlertid hjelpe til at fargene blandes bedre enn for et blankt hus.



5.4.3 Sykliske RGB-dioder

Sykliske RGB LED har som vanlige RGB LED tre lysdioder montert i samme hus. Men istedet for at de styres eksternt ved hjelp av ulike pinner, så har de innebygget en integrert styringskrets som veksler mellom å sette spenning på de ulike diodene, gjene også flere slik at man får fargeblanding (Flashing LED). Figuren til høyre viser hvordan styringskretsen er plassert inne i huset sammen med diodene¹⁹. De finnes i to typer: FAST og SLOW. I tillegg finnes de med 3, 5 og 10 mm diameter.



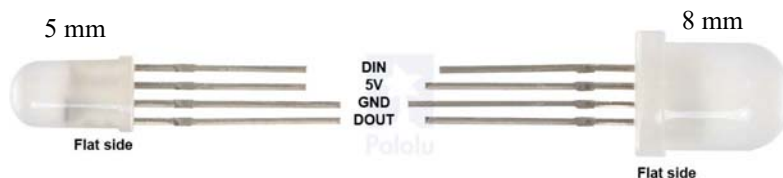
19. https://learn.sparkfun.com/tutorials/light-emitting-diodes-leds?_ga=2.37374368.1867605279.1589209561-856163918.1476131557



Typisk foroverspenning på disse diodene som selges av Sparkfun²⁰ er 2V og fargene skifter med noen få sekunders mellomrom.

5.4.4 Digitalt styrte RGB-dioder ved WS2811

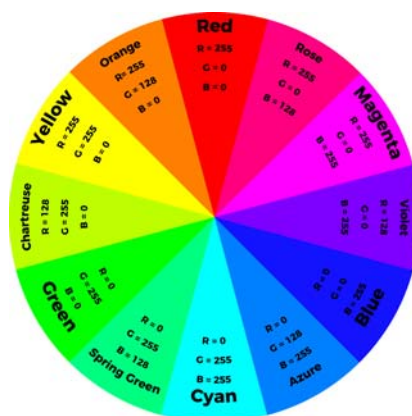
Dette er lysdioder satt sammen av en rød, en grønn og en blå LED med hver sin programmerbare strømkilde i tillegg til det som trengs for å overføre data til dioden.



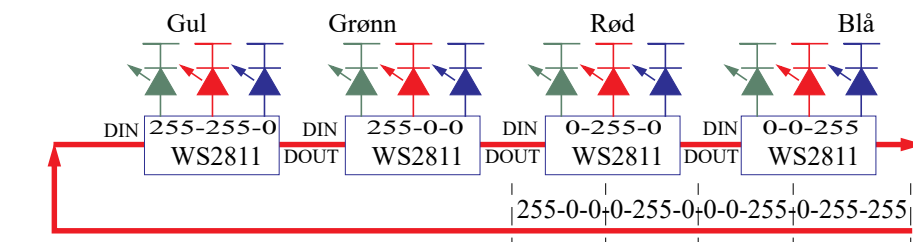
Figuren viser en variant av denne typen diode som i tillegg til GND og 5V har en DIN (data inn) og DOUT (data ut) pinne slik at flere slike lysdioder kan seriekobles og adresseres individuelt.

Intensiteten til hver av de tre lysdiodene styres av en verdi fra 0 – 255, hvor 0 gir slukket diode og 255 gir en diode som lyser med maksimal lysstyrke. Ved å variere lysstyrken til hver av de tre diodene kan man blande seg fram til de fleste fargene i paletten. I teorien kan man blande seg fram til $256 \times 256 \times 256 = 16\,777\,216$ farger inkludert at alle diodene er avslått, om man kan kalle det en farge.

Fargehjulet²¹ på figuren til høyre gir et inntrykk av hvordan man kan få fram varianter ved å blande fargene fra de tre lysdiodene. *Det man imidlertid må huske på er at rekkefølgen er annerledes enn den tradisjonelle RGB. Benytter man WS2811 er rekkefølgen lik GRB.* Dvs. at man først oppgir verdien av grønn, så rød og tilslutt blå.



Adresseringen er ganske enkel ved at man legger alle fargekodene i en lang rekke og så sender den lange serien med tall inn i rekken av lysdioder. Når alle tallene er kommet inn så ligger de på plass og kan tenne hver sin lysdiode. Dette er forsøkt illustrert i figuren under.



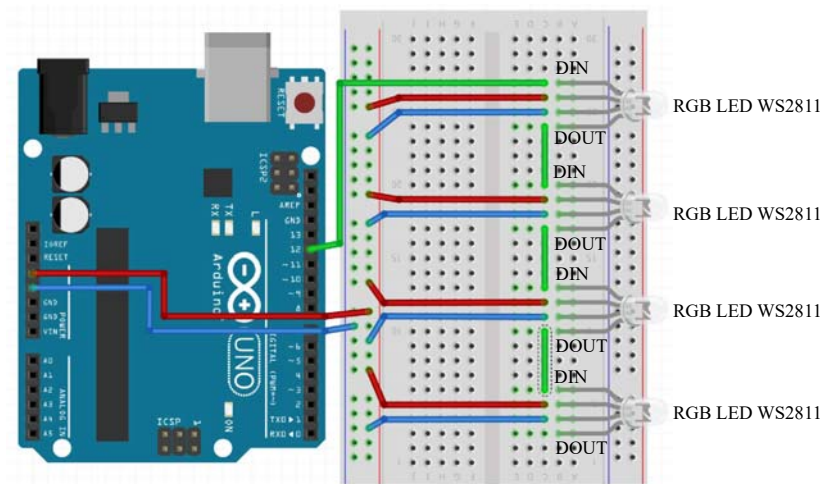
20. <https://www.sparkfun.com/products/11452>

21. <https://learn.sparkfun.com/tutorials/lilypad-rgb-led-hookup-guide/all>



Oppkobling:

Etter som diodene kan seriekobles og har en DIN og en DOUT er det lett å koble dem opp:



Slik kan vi koble mange programmerbare LED i serie.

Programmering:

- **Last ned biblioteket:** *PololuLedStrip.h* fra: <https://github.com/pololu/pololu-led-strip-arduino> og installer dette på vanlig måte i Arduino IDE.
- **Inkluder biblioteket** i programmet ved å skrive:

```
#include <PololuLedStrip.h>
```

først i programmet
- **Dernest deklarerer** en LED-stripe med f.eks. navnet: `ledStrip` samtidig som man angir hvilken port som skal mate ut dataene til LEDstripen, i vårt tilfalle ar vi valgt port 12:

```
PololuLedStrip<12> ledStrip;
```

Denne deklarasjonen gjøres tidlig i programmet før `setup()`-funksjonen
- **Så defineres et buffer** `colors[]` som skal holde alle fargekodene i riktig rekkefølge. Dette bufferet skal ha en lengde som er tilstrekkelig lang til å holde alle fargekodene, tre pr. diode og være av typen (klassen) `rgb_color`:

```
// Lag et buffer for å holde fargene (3 byte pr. farge)
#define LED_COUNT 60
rgb_color colors[LED_COUNT];
```

Også denne plasseres før `setup()`-funksjonen. Bufferlengden er definert på 60 byte. Som vi ser så vil lange LED-striper kreve mye RAM i kontrolleren.
- **Så legges fargekodene inn i bufferet** `colors[]` en etter en i henhold til ønsket farge på de enkelte RGB-diodene. I vår tilfelle har vi lagt inn fargene gul, grønn, rød og blå i de fire diodene som vi har koblet opp:

```
colors[0] = rgb_color(255, 0, 0); // Grønn
```



```
colors[1] = rgb_color(0, 255, 0); // Rød  
colors[2] = rgb_color(0, 0, 255); // Blå  
colors[3] = rgb_color(255, 255, 0); // Gul-blanding av rød og grønn.
```

Denne koden legges inn i loop()-funksjonen ev. i setup()-funksjonen dersom oppsettet gjelder så lenge programmet går eller som en startverdi. Her kan man eksperimentere med ulike verdier for å gjenskape de ønskede fargenyansene.

For ytterligere diskusjon og tips til programmeringen se: <https://github.com/pololu/pololu-led-strip-arduino>

5.5 Numeriske 7 segment display – TM1637

5.5.1 Kort omtale:

Dette er et numeriske display med fire 7-segment siffer som egner seg godt til små prosjekter da det er billig og lett å programmere via en serie-buss. Displayet er montert på et kretskort med kretsen TM1637 som gjør at det kun trenger 2 tilkoblinger til mikrokontrolleren i tillegg til spenning og jord:

5V – Displayet trenger en spenning på 3,3 – 5V

GND – Jordforbindelse (–)

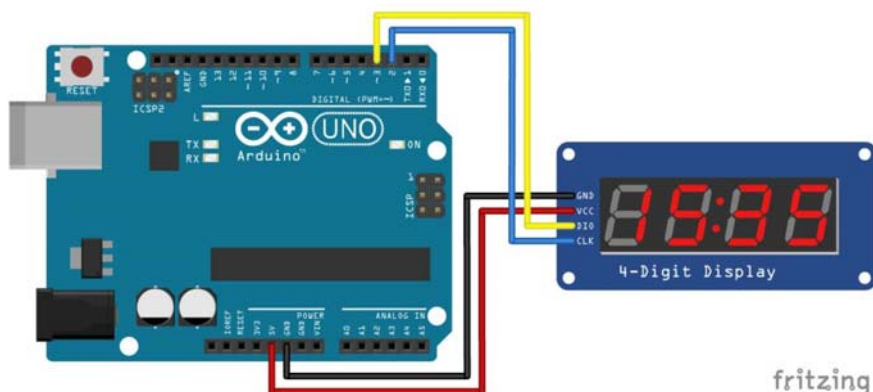
DIO – Serielle data inn

CLK – Klokkesignal

Displayet kan leveres i ulike farger, hvit, blå, grønn, rød og gul. Det har et strømtrekk på typisk 80 mA ved 5V og lysstyrken kan settes til 8 nivåer.

Displayet leveres både med desimalpunkt for hvert siffer og med kolon mellom andre og tredje siffer for bruk som klokkedisplay.

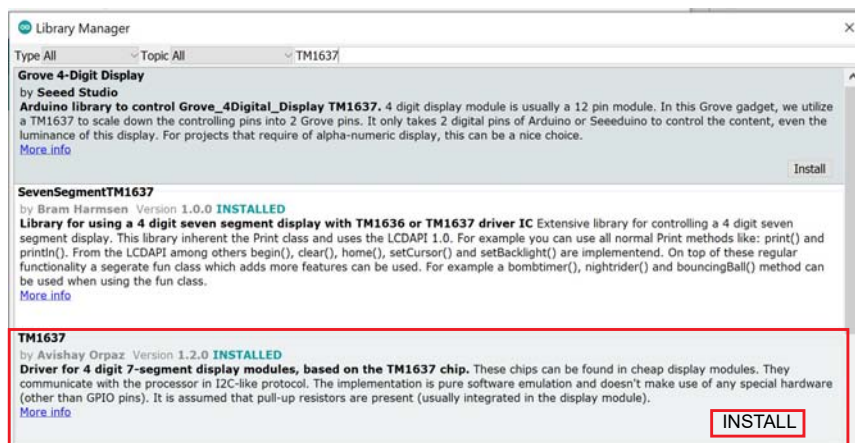
Figuren under viser hvordan displayet kan kobles opp mot Arduino UNO²².





5.5.2 Programmering

1. For å bruke displayet TM1637 så må vi installere biblioteket som hører til dette displayet. Velg *Sketch* menylinjen. Velg så *Include library* → *Library manage*. Skriv inn: TM1637 i innboksen.
2. Etter hvert vil det dukke opp flere alternative biblioteker for TM1637. Flere kan sikkert velges, men den innrammede varianten (Avishay Orpaz) synes å fungere godt også med ESP32.



Etter kort tid er biblioteket installert og klart til å brukes.

3. Inkluder biblioteket

Biblioteket inkluderes med følgende kommando i starten av programmet:

```
#include <TM1637Display.h>
```

Denne kommandoen gir tilgang til bibliotekets mange funksjoner

4. Deklarer instanser

Deklarasjonen gir vårt display navnet *display* av typen *TM1637Display*, samtidig så forteller deklarasjonen hvilke pinner klokke (CLK) og datainngangen (DIO) skal kobles til (2, 3):

```
#define CLK 2
```

```
#define DIO 3
```

```
TM1637Display display = TM1637Display(CLK, DIO);
```

Deklarasjonen gjøres tidlig i programmet før *setup()*-funksjonen.

Dersom man ønsker å koble opp flere displayer så kan disse tilknyttes andre porter på mikro-kontrolleren. Eksempelen under viser hvordan vi kan bruke tre displayer med ulike navn og forskjellig tilkoblingspunkter for klokke- og data-inngang:

```
TM1637Display display_1 = TM1637Display(2, 3);
```

```
TM1637Display display_2 = TM1637Display(4, 5);
```

```
TM1637Display display_3 = TM1637Display(6, 7);
```

22.Figuren er hentet fra <https://www.makerguides.com/tm1637-arduino-tutorial/>



5. Tøm displayet

Dersom vi ønsker å slå av samtlige segmenter slik at displayet går i “svart” kan vi bruke kommandoen:

```
display.clear();
```

Hvor *display* er navnet vi har gitt displayet. Denne kan brukes når som helst, men det kan jo være greit og “tømme” displayet i *setup()*-funksjonen før bruk.

6. Sett lysstyrke

Biblioteket gir mulighet til å sette lysstyrken til displayene i 8 trinn fra 0 (slukket) til 7 (maksimal lysstyrke). I eksempelet under er lysstyrken satt til maks.:

```
display.setBrightness(7);
```

Kommandoen kan brukes hvor som helst, men det er ikke unormalt å sette lysstyrken i *setup()*-funksjonen. Den fullstendige funksjonen kan skrives slik:

```
void display.setBrightness(uint8_t brightness, bool on = true);
```

Det vil si at parameter nr. to kan slå på (*true*) eller av displayet (*false*).

7. Skriv tall

Vi kan skrive heltall, f.eks. innholdet i variabelen *i*, til displayet med følgende kommando:

```
display.showNumberDec(i);
```

Denne vil kunne skrive tall, positive tall opp til 9999, og negative tall opp til -999. Kommandoen vil vise fortegnet for negative tall på displayet.

Kommandoen har flere argumenter og kan fullt utskrevet uttrykkes som:

```
display.showNumberDec(i, true, 4, 0);
```

Hvor *i* er tallet som ønskes vist, *true* angir at ledende 0’er skal vises, *false* at de ikke skal vises, og *0* at mest signifikante siffer er lengst til venstre, 3 betyr at mest signifikante siffer er lengst til høyre.

Det finnes også en utvidet kommando som gir mulighet til å sette desimalpunktet der man ønsker:

```
display.showNumberDecEx(i, 0b00100000, true, 4, 0);
```

Hvor *i* er tallet som ønskes vist, *0b0010000* angir desimalpunkt etter 3. siffer fra venstre, *true* angir ledende 0 skal vises, *false* at de ikke skal vises og *0* at mest signifikante siffer er lengst til venstre, 3 betyr at mest signifikante siffer er lengst til høyre,

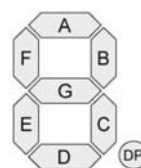
8. Skriv til segmenter

Man kan også skrive til de enkelte segmentene til hvert av sifrene hos displayet. En måte å gjøre dette på er å definere et array av fire 8-bits elementer (byte), hvor hvert element henviser til hvert av de fire sifrene. I eksempelet under er arrayet kalt *data*:

```
const uint8_t data[] = {0xFF, 0xFF, 0xFF, 0xFF};
```

Arrayet er i denne sammenhengen definert som en konstant bestående av fire byte uten fortegn *uint8_t*. Hver bit i hver byte representerer ett av de 8 segmentene inkludert desimalpunktet i hvert av de fire sifrene. Konstanten *data[]* vil slå på alle segmentene (0xff). Tilsvarende vil følgende eksempel:

```
const uint8_t data[] = {0x00, 0x00, 0x00, 0x00};
```





```
slå av segmentene og tilsvare kommandoen display.clear() ;  
For å sende dette arrayet til displayet brukes kommandoen:  
display.setSegments(data) ;
```

For lettere å vite hvilket segment man skriver til inneholder biblioteket mulighet til å slå på de ulike segmentene ved å bruke konstantene: SEG_A, SEG_B, SEG_C, SEG_D, SEG_E, SEG_F, SEG_G. Dette egner seg godt dersom man f.eks. vil skrive en tekst på displayet:

```
const uint8_t done[] = {  
  SEG_B | SEG_C | SEG_D | SEG_E | SEG_G,           // d  
  SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,    // O  
  SEG_C | SEG_E | SEG_G,                             // n  
  SEG_A | SEG_D | SEG_E | SEG_F | SEG_G             // E  
};
```

Her defineres ordet dOnE.

Dette skrives da til displayet med følgende kommando:

```
display.setSegments(done) ;
```

5.6 Servomotorer

Det finnes flere typer servomotorer. Her skal vi ta for oss 180° og 360° motorer. 180° graders motorer kan dreie en arm til en gitt vinkel fra 0 – 180° (f.eks. FS90). En 360° motor oppfører seg som en vanlig motor (FS90R). Spesielt med motorene er at dreievinkelen eller hastigheten kan styres ganske nøyaktig ved hjelp av lengden av en puls. Her vil vi behandle begge typende under ett.

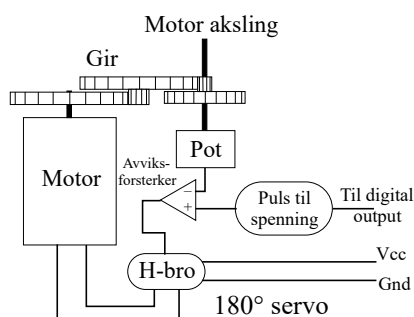


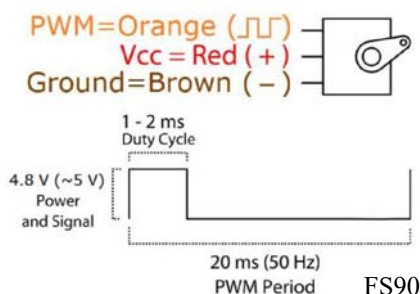
5.6.1 Virkemåte

En 180° servo er en motor hvor en har full kontroll på vinkelen akslingen skal dreie. Normalt fra 0° til 180°. Dette skjer ved en intern tilbakekobling som vist i figuren til høyre. Servoen styres av en puls på styreinngangen. FS90 er en slik 180° servo.

En *kontinuerlig roterende servo* kan dreie 360° og vil fungere som en vanlig motor. En slik vil ikke ha tilbakekobling som vist på figuren til høyre.

Motorene tilføres normalt tre ledninger. I tillegg til spenning, Vcc, typisk 5V, og jord (GND) så tilføres et styresignal som hentes fra en av de digitale portene hos mikrokontrolleren. Det må være en port som kan tilby PWM – Pulsbreddemodulasjon.





FS90: 180° motorene styres ved hjelp av et pulstog²³. Figuren til venstre viser en typisk situasjon for FS90. Den forlanger en puls på mellom 1 – 2 ms, som gjentar seg med en periode på 20ms (50Hz).

En pulslengde på 1,5 ms vil posisjonere servoen i posisjon 0°. En puls på 1,0 ms vil dreie den –90° mot høyre, mens en puls på 2,0 ms vil dreie den 90° mot venstre. Ved å endre på pulslengden så endres posisjonen. Fargene på figuren over angir fargene på de tre ledningene til servoen.

FS90R: En 360° motor styres på tilsvarende måte av en lignende puls. Dette gjelder f.eks. FS90R der en pulslengde på 1,5 ms $\pm 0,05$ ms vil sørge for at servoen står i ro. En puls på 0,9 ms vil sørge for at motoren får full fart mot høyre, mens en puls på 2,1 ms vil gi full fart mot venstre. Ved å endre på pulslengden så endres farten.

En kan imidlertid erfare at motoren ikke står stille ved en pulslengde på 1,5 ms. I så fall finnes det en trimmeskrue på undersiden av servoen som kan justeres slik at den står stille ved den angitt pulslengden på 1,5 ms. Denne motoren har normalt en vindu hvor den skal være i ro på $\pm 0,045$ ms²⁴.



5.6.2 Viktige parametere for servoer

Av viktig parametere kan vi nevne følgende. Vi bruker fortsatt FS90 som eksempel på en 180° servomotor og FS90R som eksempel på en 360° servomotor:

- **Driftspenning og strøm:** Denne angir typisk spenningsområde for servoene. For både FS90 og FS90R er dette 4,8 – 6V, de egner seg derfor godt for bruk med Arduino. Servoene har selvfølgelig et strømtrekk som varierer fra når de er i ro (typ. 5 – 6mA) til de er i aktiv bevegelse (typ. 100–120mA). Dersom motorene blokkeres vil strømmen øke betydelig typisk 700 – 800mA for FS90 og 550 – 650 for FS90R, hvilken lett kan være til skade om det får stå på en stund.
- **Dimensjoner:** Størrelse og vekt er viktige dersom servoen skal brukes i fly. Dette kan være tilfelle dersom servoen skal dreie side- eller haleror. Både FS90 og FS90R har en vekt 9g og dimensjoner (23,2 x 12,0 x 22,0 mm).

23. http://www.ee.ic.ac.uk/pcheung/teaching/DE1_EE/stores/sg90_datasheet.pdf

24. <https://www.pololu.com/product/2820>



- **Styrke:** Styrken oppgis som dreiemomentet (kraft x arm) når motoren hindres i å rotere, dette kalles “stall torque”. For både FS90 og FS90R er det oppgitt til å ha en toppverdi på 1,3 – 1,5 kg cm når motoren hindres i å rotere (“staller”). Dreiemomentet er avhengig av hvilken spenning motoren tilføres. Det opplyses også om hvilket materiale giret er laget av da dette har betydning for hvilken påkjenning det tåler. Det må antas at nylon, som i dette tilfellet, er svakere enn metall.
- **Hastighet:** For en 180° servo vil hastigheten angis ved hvor lang tid det tar å dreie den 60°. For FS90 er det ca. 0,1 – 0,12 sek. avhengig av spenningen. Jo høyere spenning, jo raskere. For en 360° servo angis hastigheten som antall omdreining pr minutt (rpm). For FS90R er denne på 110 – 130 rpm avhengig av spenningen.
- **Tilleggsutstyr:** Normalt følger det med ulike armer og festeskiver med servoen for ulike bruk. Disse tres ned på akslingen som gjerne er riflet. Settet som er vist på figuren over følger gjerne alle servomotorer av denne typen.



5.6.3 Programmering av servoer

Arduino har egne bibliotek som gir oss et sett av funksjoner som gjør det lett å programmere servomotorer. Det er viktig å merke seg at det kan være forskjellige biblioteker avhengig av om vi bruker ESP32 eller f.eks. Arduino UNO.

- **Hent bibliotek:** Biblioteket for styring av servoer er standard for Arduino og trenger normalt ikke å installeres. Dersom vi bruker ESP32 må vi installere et eget bibliotek som kan hentes her: <https://github.com/RoboticsBrno/ServoESP32>
- **Inkludering av biblioteket:**
Biblioteket inkluderes ved å skrive

```
#include <Servo.h>
```


øverst i fila.
- **Deklarasjon av servoene:**
Derneft må vi deklare servoer eller servoene med navn og en type. Dersom vi ønsker å bruke flere servoer gir vi dem forskjellige navn. Det kan f.eks. være tilfelle dersom vi skal laga en robot og trenger en servo for hvert av de to hjulene. I alt kan man definere 8 servoer for en en Arduino UNO:

```
Servo myleftservo; // Deklarerer objektet myleftservo av typen servo  
Servo myrightservo; // Deklarerer objektet myrightservo av typen servo  
Servo myarmservo; // Deklarerer objektet myarmservo av typen servo
```


Deklarasjonene gjøres før setup()-funksjonen



- **Tilkobling og frakobling**

Dernest må vi fortelle Arduino'en hvilken digital utgang <pin> som skal kobles til styreinngangen på servoen.

```
myleftservo.attach(9); // Knytt venstre servo til pinne 9
myrightservo.attach(10); // Knytt høyre servo til pinne 10
myarmservo.attach(11); // Knytt servoen styrer armen til pinne 11
```

Disse tilordningene gjøres i setup-funksjonen

Tilsvarende kan en frakoble servoene om det er behov for det med følgende kommandoer:

```
myleftservo.detach(); // Koble venstre servo fra pinne 9
myrightservo.detach(); // Koble høyre servo fra pinne 10
myarmservo.detach(); // Koble høyre servo fra pinne 11
```

Det kreves ikke noe pinnenummer ved frakobling.

- **Styring av servo med “write()”:**

Servoen styres med kommandoen – “write”

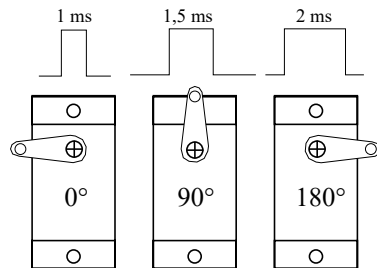
```
myarmservo.write(60);
```

Kommandoen skrives normalt i loop()-funksjonen.

Hos en *180° servo* vil vinkel angi dreiningsvinkelen til servoen i grader (0° – 180°).

Hos en *kontinuerlig roterende servo* vil vinkelen angi motorens hastighet og retning. Her vil 0° angi full hastighet i den ene retningen, 90° være i ro, mens 180° angir full hastighet i motsatt retning.

Kommandoene plasseres i loop()-funksjonen etter behov



Egentlig styres servoen av pulser som antydnet på figuren over.

- **Styring av servo med “writeMicroseconds()”:**

Denne kommandoen fungerer omtrent som “write”, men angir direkte lengden av pulsen i mikrosekunder. Følgende er vanlige verdier for en *180° servo*:

1000µs dreies til posisjon 0°

1500µs dreies til posisjon 90°

2000µs dreies til posisjon 180°

Følgende er vanlige verdier for en *kontinuerlig roterende servo*:

1000µs full fart *mot* urviseren

1500µs i ro

2000µs full fart *med* urviseren

Det er viktig å unngå at servoen står å stanger mot et ytterpunkt, da dette trekker mye strøm, samtidig som det kan skade komponenten.

- **Monitoreringskommandoen – “read”**

Kommandoen “read” returnerer siste “write”-verdi sendt til servoen.

```
vinkel = myarmservo.read();
```

Kommandoen plasseres i loop()-funksjonen etter behov.

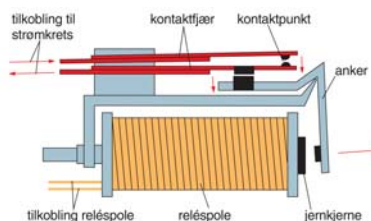


5.7 Releer

Det sier seg selv at en mikrokontroller ikke uten videre kan styre en store strømmer eller spenninger siden de operer med spenninger fra typisk 3,0 – 5,0 V. Vi trenger derfor en komponent som lar seg styre med lave spenninger og strømmer, men som selv kan styre store strømmer og spenninger. Vi skal se nærmere på reler som nettopp har denne egenskapen.

5.7.1 Releets funksjon

Figuren høyre²⁵ viser hvordan et tradisjonelt rele fungerer. Spolen til en elektromagnet er koblet til lavspenningsdelen til f.eks. en mikrokontroller. Når spolen får strøm fra kontrolleren vil *ankeret* trekkes til og en bryter legges over og slutter en høyspenningskrets. Som vi ser er det en rent elektromagnetisk-mekanisk bryter.



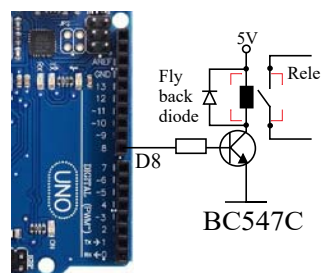
Det som ikke er så uvanlig når det gjelder tilkobling av reler til f.eks. Arduino er at selv strømmen som kreves for å dra ankeret i posisjon er større enn det en Arduino normalt klarer. En Arduino kan normalt lever fra 20 - 40 mA, mens et rele kanskje trenger en noe større strøm.

5.7.2 Transistordriveren

En transistor er en komponent som gjør det mulig å styre en større strøm med en liten strøm eller liten spenning. Derfor kan det være ideelt å bruke en transistor mellom en Arduino-port og et rele som vist på figuren til høyre.

“Fly back” dioden

Vi legger merke til dioden som er koblet over spolen. Den er montert slik at det normalt ikke går strøm i den. Dette er en viktig komponent skal beskytte transistoren når strømmen i spolen brytes. Når en strøm brytes brått i spole så vil dannes en motspenning som hindrer strømmen i å forsvinne brått. Denne motspenningen kan lett bli svært høy og ta knekken på transistoren. Denne dioden derimot vil kortslutte og dempe denne spenningen. Dioden er svært viktig. Uten denne dioden vil transistoren ganske snart ødelegges.



25.Store norske leksikon

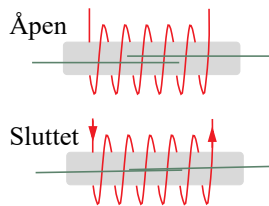
5.7.3 Reed releet

Det finnes mange typer releer. Et av de mest populære i sammenheng med små kretser med relativt små strømmer er reed-releer. Som navnet sier så består de av to tynne metallfjærer montert i et rør av glass og som normalt ikke berører hverandre (se bildet under). Disse metallfjærene er av et metall som lar seg magnetisere. Dersom vi nærmer oss disse metallfjærene med en magnet vil releet slå inn og slutte kretsen. De to fjærene blir magnetiske slik at de tiltrekker hverandre.



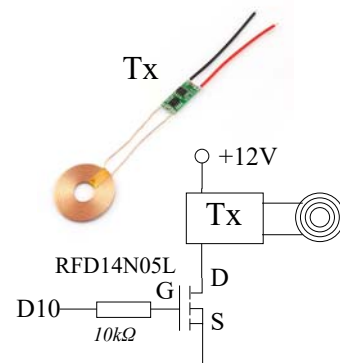
Reed reler egner seg derfor godt i en sammenheng hvor en elektrisk krets skal sluttes når en mekanisk arm nærmer seg et punkt.

En kan også vikle en spole rundt glassrøret og på den måten oppnå at metallfjærene slutter kretsen når det går en strøm i spolen. Dette siste er gjerne tilfellet for reed releer i salg. Glassrør med fjærer og spole er montert i et plasthus som det stikker ledninger ut av som vist på figuren til høyre. Også disse må gjerne ha en drivertransistor koblet mellom kontrollerkortet og spolen.



5.7.4 FET-transistorer for styring av store likestrømmer.

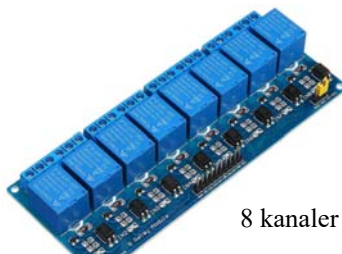
Dersom vi har behov for en relativt stor strøm, men begrenset spenning så kan vi f.eks. benytte FET-transistorer. Disse kan gjerne drives direkte fra en av de digitale portene til en Arduino siden de er spenningsstyrte og ikke krever noen strøm. Det er også enkelt å styre en slik FET-transistor selv komponenten den styrer trenger en høyere spenning en 5V. Figuren til høyre viser hvordan en FET-transistor slå av på en strøm på mange amper med spenninger på noen titals volt. I vårt eksempel slås det på en ladespole (Tx) for overføring av trådløs energi til en mobiltelefon.





5.7.5 Releer med innebygget driver

Det leveres en rekke ulike relekort med fra ett til 8 releer som kan styres direkte fra en Arduino siden kortet inneholder drivere. På figuren under er det vist et par eksempler. Disse er utstyrt med opto-kobler på inngangen slik at lav- og høyspenningsdelen og er galvanisk skilt fra hverandre. Relene tåler 10A, 230V AC, og 10A, 30V DC.



Det finnes en mengde slike varianter om man leter.



6 Referanser

- [1] Mer informasjon om Sparkfun Invention's kit:
<https://www.sparkfun.com/products/11227>
- [2] Mer informasjon om I²C-bussen:
<http://www.i2c-bus.org/>
- [3] Mer informasjon om Serikommunikasjon på SPI-bussen:
http://en.wikipedia.org/wiki/Serial_Peripheral_Interface_Bus
- [4] Koblingsskjema for Arduino UNO:
http://arduino.cc/en/uploads/Main/Arduino_Uno_Rev3-schematic.pdf.
- [5] For mer informasjon om Arduino UNO R3 layout:
<http://arduino.cc/en/Main/ArduinoBoardUno/>
- [6] For nedlasting av programeditoren IDE for Arduino:
<http://arduino.cc/hu/Main/Software>
- [7] For nedlasting av Referansemanualen for Arduino C++
<http://arduino.cc/en/Reference/HomePage>
- [8] Skolelaboratoriets blå hefteserie:
<https://www.ntnu.no/skolelab/bla-hefteserie>
- [9] Elliot Williams, *Make: AVR Programming – Learning to Write Software for Hardware*, Make Community, LLC; 1 edition (February 25, 2014)



Vedlegg A Komponentliste

Tabellen under inneholder noen av komponentene som er brukt og hvor de kan skaffes fra:

Typebetegnelse	Type komponent	Leverandør	Nettadresse
TM1637	Display 4x7 seg.	Banggood	https://www.banggood.com/0_36-Inch-4-Digit-LED-Display-Tube-7-segments-TM1637-30x14mm-Yellow-Decimal-Point-Module-p-1654852.html
RFID-RC522	RFID-leser	Kult og Billig	http://www.kultogbillig.no/RFID-RC522-RF-IC-Card-Sensor-Modul?keyword=RFID
LED Hvit	Lysdioder	Banggood	https://www.banggood.com/100PCS-5MM-20mA-Ultra-Bright-Transparent-Round-White-2Pin-LED-Diode-DIY-Light-p-1535246.html
Mini	Koblingsbrett	Banggood	https://www.banggood.com/Mini-Solderless-Prototype-Breadboard-170-Points-For-Arduino-Shield-p-74814.html
Jumpere	Diverse	Banggood	https://www.banggood.com/40pcs-20cm-Male-To-Female-Jumper-Cable-Dupont-Wire-For-Arduino-p-973822.html?rmmds=search&cur_warehouse=CN
ESP WROOM 32 30 pin m/hylselist	Mikrokontroller	Kult og Billig	http://www.kultogbillig.no/ESP32-Development-Board-WiFi-Bluetooth-Ultra-Low-Power-Consumption-Dual-Cores-ESP-32-ESP-32S-Board?keyword=ESP32
ESP WROOM 32 30 pin m/hylselist	Mikrokontroller	Banggood	https://www.banggood.com/Geekcreit-ESP32-WiFi-bluetooth-Development-Board-Ultra-Low-Power-Consumption-Dual-Cores-Unsoldered-p-1214159.html
USB-kabel	USB A → micro	Banggood	https://www.banggood.com/BlitzWolf-BW-MC12-Micro-USB-Charging-Data-Cable-1ft0_3m-For-Samsung-S7-S6-Xiaomi-Redmi-Note-5-p-1339804.html
SG90	Servo 180°	Banggood	https://www.banggood.com/6PCS-SG90-Mini-Analog-Gear-Micro-Servo-9g-For-RC-Airplane-Helicopter-p-1078614.html
Rele	Rele 3V enpolet Aktivt lav	Banggood	https://www.banggood.com/10pcs-1-Channel-3_3V-Low-Level-Trigger-Relay-Module-Optocoupler-Isolation-Terminal-p-1557154.html
Rele	Rele 5V enpolet Aktivt lav	Banggood	https://www.banggood.com/1-Channel-5V-Relay-Module-with-Optocoupler-Isolation-Relay-Board-p-1399429.html?rmmds=search&cur_warehouse=CN



Vedlegg B Programmer løsningsforslag

B.1 Oppdrag 1 – Enkel optisk portal

```
// Optisk portal-1
int pinLED = 16;
int pinLDR = 36;
int verdiLDR;
int terskelLDR = 500;
int antall = 0;

void setup()
{
    pinMode(pinLED, OUTPUT);

    Serial.begin(9600);

    digitalWrite(pinLED, HIGH);
    delay(500);
}

void loop()
{
    verdiLDR = analogRead(pinLDR);

    if(verdiLDR < terskelLDR)
    {
        Serial.println(verdiLDR);
        antall = antall + 1;

        while(verdiLDR < terskelLDR)
        {
            verdiLDR = analogRead(pinLDR);
        }
    }
    Serial.println(antall);
    delay(100);
}
```



```
}
```

B.2 Oppdrag 2 – Skriv til display

```
/*
 * I denne øvelsen skal vi teste displayet
 * og lære hvordan vi skriver ut tall til
 * Nils Kr. Rossing 03.07.20 og Kåre-Benjamin H. Rørvik 15.07.20
 */

#include <Arduino.h>
#include <TM1637Display.h>

const byte PIN_CLK = 32;    // Definer CLK pin og velg port D32
const byte PIN_DIO = 33;    // Definer DIO pin og velg port D33

TM1637Display display(PIN_CLK, PIN_DIO);
// Definer instansen display av typen TM1637Display

int antall = 0; // Definer en variabel antall og sett den lik 0

void setup()
{
    display.setBrightness(7); // Setter lysstyrken på displayet. Fra 0
    (min) til 7 (max).
}

void loop()
{
    display.clear();           // Tøm displayet for data
    display.showNumberDec(antall); // Vis tall
    delay(1000);               // Vent i 1000 millisek. (1 sek.)
    antall = antall + 1;        // Øk telleren med 1
    while(antall == 11){}      // Stopp og vent her for alltid
}
```



B.3 Oppdrag 3 – Tell antall passeringer og vis på display

Programmet er en sammenstilling av Optisk portal og skriv til display

```
// Tell og vis antall passeringer på displayet
// Nils Kr. Rossing 08.07.20
// Gjennomgått og testet av Kåre-Benjamin H. Rørvik 15.07.20

#include <Arduino.h>
#include <TM1637Display.h>

const byte PIN_CLK = 32;    // Definer CLK pin og velg port D32
const byte PIN_DIO = 33;    // Definer DIO pin og velg port D33

TM1637Display display(PIN_CLK, PIN_DIO);
// Definer instansen display av typen SevenSegmentTM1637

int pinLED = 16;
int pinLDR = 36;
int verdiLDR;
int terskelLDR = 2000;
int antall = 0;

void setup()
{
  Serial.begin(115200);
  display.setBrightness(7); // Setter lysstyrken på displayet.
                           // Fra 0 (min) til 7 (max).

  pinMode(pinLED, OUTPUT);
  digitalWrite(pinLED, HIGH);

  delay(500);
}

void loop()
{
```



```
verdiLDR = analogRead(pinLDR);

if(verdiLDR < terskelLDR)
{
    antall = antall + 1;
    Serial.print("Antall passeringer ");
    Serial.println(antall);
    while(verdiLDR < terskelLDR)
    {
        verdiLDR = analogRead(pinLDR);
    }
}
display.showNumberDec(antall);           // Vis tall

}
```

B.4 Oppdrag 4 – Sett opp servoen slik at den åpner og lukker bommen

Programmet tester ut servoen slik at den slår opp og ned. Programmet kan brukes til å finne de riktige verdiene for at bommen skal stå i riktig lukket og åpen funksjon.

Complete project details at <https://randomnerdtutorials.com>

Written by BARRAGAN and modified by Scott Fitzgerald

Modifisert av Nils Kr. Rossing 05.07.20,

Gjennomgått og testet av Kåre-Benjamin H. Rørvik 15.07.20

*****/

```
#include <Servo.h>
```

```
Servo myservo; // Lag et objekt med navnet myservo
```

```
// Hele 12 slike objekter kan lages ved hvert kort
```

```
int pos; // variabel som holde posisjonen i grader
```

```
void setup() {
```

```
    myservo.attach(13); // attaches the servo on pin 13 to the servo object
```

```
}
```



```
void loop()
{
    pos = 0;                // Her kan du justere posisjonen om du ønsker
                           // å få en penere possisjonering.
    myservo.write(pos); // Be servoen om å gå til posisjon pos
    delay(5000);           // Stå i posisjon 0 i 5 sek

    pos = 90;               // Her kan du justere posisjonen om
                           // du ønsker å få en penere posisjonering.
    myservo.write(pos); // Be servoen om å gå til posisjon pos
    delay(3500);           // Stå i posisjon 0 i 5 sek
}
```

B.5 Oppdrag 5 – Sammenstill enkel optisk portal, teller og åpning av bommen

Dette programmet teller opp antall passeringer i den optiske portalen og viser resultatet på displayet, dessuten åpner det bommen når noen skal inn og lukker den etter dem.

```
// Tell og vis antall passeringer på displayet samt åpne og lukke bommen
// Nils Kr. Rossing 05.07.20
// Kåre-Benjamin Hammervold Rørvik 16.7.2020
```

```
#include <TM1637Display.h>
#include <Servo.h>
```

```
const byte PIN_CLK = 32; // Definer CLK pin og velg port D32
const byte PIN_DIO = 33; // Definer DIO pin og velg port D33
```

```
TM1637Display display(PIN_CLK, PIN_DIO);
// Lag et objekt med navnet display av typen TM1637Display
Servo myservo; // Lag et objekt med navnet myservo av typen Servo
```

```
int pinLED = 16;
int pinLDR = 36;
int pinServo = 13;
int verdiLDR;
int terskelLDR = 2000;
int antall = 0;
```



```
int posLukket = 0;    // Angir posisjonen til servoen i lukket tilstand
int posAapen  = 90;   // Angir posisjon til servoen i åpen tilstand

void setup()
{
    Serial.begin(115200);

    pinMode(pinLED, OUTPUT);
    pinMode(pinServo, OUTPUT);
    pinMode(PIN_CLK, OUTPUT);
    pinMode(PIN_DIO, OUTPUT);
    digitalWrite(pinLED, HIGH);

    myservo.attach(pinServo); // Koble servoen til port D13

    display.setBrightness(7); // Setter lysstyrken på displayet.
                             // Fra 0 (min) til 7 (max).
    display.showNumberDec(antall); //Starter Displayet (får da 0).
    Serial.print("Antall passeringer ");
    Serial.println(antall);

    delay(500);
}

void loop()
{
    verdiLDR = analogRead(pinLDR);
    myservo.write(posLukket);           // Lukk bommen
    delay(100);

    if(verdiLDR < terskelLDR)
    {
        antall = antall + 1;
        myservo.write(posAapen);       // Åpne bommen
        delay(100);
        display.showNumberDec(antall); // Vis tall
    }
}
```



```
Serial.print("Antall passeringer ");
Serial.println(antall);
while(verdiLDR < terskelLDR)
{
    verdiLDR = analogRead(pinLDR);
    //delay(1000); // Dette kan brukes til å jevne ut små bevegelser
                // fra sensoren, og forholde døren åpen.
                // Fjern // før "delay" for å bruke denne linjen.
}
delay(3500);
myservo.write(posLukket);           // Lukk bommen
delay(100);
}
}
```

B.6 Oppdrag 6 – Styring av rele

Programmet teller antall passeringer, åpner bom ved passering og slår på lyset når antallet er større enn 0.

```
// Tell og vis antall passeringer på displayet samt åpne og lukke bommen
// I tillegg slås releet på når det er personer i rommet
// Nils Kr. Rossing 05.07.20
// Kåre-Benjamin Hammervold Rørvik 16.7.2020
```

```
#include <TM1637Display.h>
#include <Servo.h>
```

```
const byte PIN_CLK = 32;    // Definer CLK pin og velg port D32
const byte PIN_DIO = 33;    // Definer DIO pin og velg port D33
```

```
TM1637Display display(PIN_CLK, PIN_DIO
    // Lag et objekt med navnet display av typen TM1637Display
Servo myservo; // Lag et objekt med navnet myservo av typen Servo
```

```
int pinLED = 16;
int pinLDR = 36;
int pinRele = 14;
int pinServo = 13;
```



```
int verdiLDR;
int terskelLDR = 2000;
int antall = 0;
int posLukket = 0; // Angir posisjonen til servoen i lukket tilstand
int posAapen = 90; // Angir posisjon til servoen i åpen tilstand

void setup()
{
    Serial.begin(115200);

    pinMode(pinLED, OUTPUT);
    pinMode(pinRele, OUTPUT);
    pinMode(pinServo, OUTPUT);
    pinMode(PIN_CLK, OUTPUT);
    pinMode(PIN_DIO, OUTPUT);
    digitalWrite(pinLED, HIGH);

    myservo.attach(pinServo); // Koble servoen til port D13

    display.setBrightness(7); // Setter lysstyrken på displayet.
                             // Fra 0 (min) til 7 (max).
    display.showNumberDec(antall); //Starter Displayet (får da 0).
    Serial.print("Antall passeringer ");
    Serial.println(antall);

    delay(500);
}

void loop()
{
    verdiLDR = analogRead(pinLDR);
    myservo.write(posLukket); // Lukk bommen
    delay(100);

    if(verdiLDR < terskelLDR)
    {
```




```
    antall = antall + 1;
    myservo.write(posAapen);           // Åpne bommen
    delay(100);
    display.showNumberDec(antall);      // Vis tall
    while(verdiLDR < terskelLDR)
    {
        verdiLDR = analogRead(pinLDR);
        //delay(1000); //Dette kan brukes til å jevne ut små bevegelser
        // fra sensoren, og forholde døren åpen.
        // Fjern // før "delay" for å bruke denne linjen.
    }
    delay(3500);
    myservo.write(posLukket);           // Lukk bommen
    delay(100);
}

// Styring av rele. Relet er aktivt høy
// (men kan også være lav for andre typer Relé...). Strømtrekk 0,6 mA
if(antall == 0)
{
    digitalWrite(pinRele, LOW);
    Serial.print("Reléet er AV og ");
    Serial.print("antall passeringer er: ");
    Serial.println(antall);
}
else
{
    digitalWrite(pinRele, HIGH);
    Serial.print("Reléet er PÅ og ");
    Serial.print("antall passeringer er: ");
    Serial.println(antall);
}
}
```

B.7 Oppdrag 7 – Avansert optisk port, registrering av inngang og utgang

Programmet tester bare telling av de som kommer og de som går.

```
// Tell antallet som går inn og ut
```



```
// Nils Kr. Rossing 05.07.20
// Kåre-Benjamin Hammervold Rørvik 16.7.2020

int pinLED1 = 16;
int pinLDR1 = 36;
int verdiLDR1;
int terskelLDR1 = 1500;
int flaggSluse1 = 0;
int pinLED2 = 18;
int pinLDR2 = 39;
int verdiLDR2;
int terskelLDR2 = 1500;
int flaggSluse2 = 0;

int antall = 0;

void setup()
{
    Serial.begin(115200);

    pinMode(pinLED1, OUTPUT);
    pinMode(pinLED2, OUTPUT);
    digitalWrite(pinLED1, HIGH);
    digitalWrite(pinLED2, HIGH);

    delay(500);
}

void loop()
{
    verdiLDR1 = analogRead(pinLDR1);
    verdiLDR2 = analogRead(pinLDR2);

    // Registrerer innganger

    if((verdiLDR1 < terskelLDR1) && (flaggSluse1 == 0) && (flaggSluse2 == 0))
    {
        while(verdiLDR1 < terskelLDR1)
        {
            verdiLDR1 = analogRead(pinLDR1);
            delay(100);
        }
    }
}
```



```
}
flaggSluse1 = 1;
delay(100);
}

if((verdiLDR2 < terskelLDR2) && (flaggSluse1 == 1) && (flaggSluse2 == 0))
{
    while(verdiLDR2 < terskelLDR2)
    {
        verdiLDR2 = analogRead(pinLDR2);
        delay(100);
    }
    flaggSluse1 = 0;
    antall = antall + 1;
    delay(100);
}

// registrerer utganger

if((verdiLDR2 < terskelLDR2) && (flaggSluse2 == 0) && (flaggSluse1 == 0))
{
    while(verdiLDR2 < terskelLDR2)
    {
        verdiLDR2 = analogRead(pinLDR2);
        delay(100);
    }
    flaggSluse2 = 1;
    delay(100);
}

if((verdiLDR1 < terskelLDR1) && (flaggSluse2 == 1) && (flaggSluse1 == 0))
{
    while(verdiLDR1 < terskelLDR1)
    {
        verdiLDR1 = analogRead(pinLDR1);
        delay(100);
    }
    flaggSluse2 = 0;
    antall = antall - 1;
    delay(100);
}
```



```
// Skriv ut antallet til monitor
Serial.print("Det er foreløpig: ");
Serial.print(antall);
Serial.println(" inne i rommet.");
}
```

B.8 Oppdrag 8 – Avansert optisk portal med servo bom

Programmet er likt det som er presentert i oppdrag 7 bare at bommen går opp og ned når personer går inn og ut av rommet.

```
// Tell antallet som går inn og ut og åpne og lukke bom
// Nils Kr. Rossing 06.07.20
// Kåre-Benjamin H. Rørvik 16.07.20

#include <Servo.h>

Servo myservo; // Lag et objekt med navnet myservo av typen Servo

int pinLED1 = 16;
int pinLDR1 = 36;
int verdiLDR1;
int terskelLDR1 = 1500;
int flaggSluse1 = 0;
int pinLED2 = 18;
int pinLDR2 = 39;
int verdiLDR2;
int terskelLDR2 = 1500;
int flaggSluse2 = 0;

int pinServo = 13;
int posLukket = 0; // Angir posisjonen til servoen i lukket tilstand
int posApen = 90; // Angir posisjon til servoen i åpen tilstand
int servoFlagg = 0;

int antall = 0;

void setup()
{
    Serial.begin(115200);

    pinMode(pinLED1, OUTPUT);
    pinMode(pinLED2, OUTPUT);
```



```
pinMode(pinServo, OUTPUT);
digitalWrite(pinLED1, HIGH);
digitalWrite(pinLED2, HIGH);

myservo.attach(pinServo);      // Koble servoen til port D13
myservo.write(posLukket);      // Lukk bommen
delay(100);

delay(500);
}

void loop()
{
  verdiLDR1 = analogRead(pinLDR1);
  verdiLDR2 = analogRead(pinLDR2);

  // Registrerer innganger

  if((verdiLDR1 < terskelLDR1) && (flaggSluse1 == 0) && (flaggSluse2 == 0))
  {
    while(verdiLDR1 < terskelLDR1)
    {
      verdiLDR1 = analogRead(pinLDR1);
      delay(100);
    }
    flaggSluse1 = 1;
    delay(100);
  }

  if((verdiLDR2 < terskelLDR2) && (flaggSluse1 == 1) && (flaggSluse2 == 0))
  {
    while(verdiLDR2 < terskelLDR2)
    {
      verdiLDR2 = analogRead(pinLDR2);
      delay(100);
    }
    flaggSluse1 = 0;
    antall = antall + 1;
    delay(100);
  }
}
```



```
// registrerer utganger

if((verdiLDR2 < terskelLDR2) && (flaggSluse2 == 0) && (flaggSluse1 == 0))
{
    while(verdiLDR2 < terskelLDR2)
    {
        verdiLDR2 = analogRead(pinLDR2);
        delay(100);
    }
    flaggSluse2 = 1;
    delay(100);
}

if((verdiLDR1 < terskelLDR1) && (flaggSluse2 == 1) && (flaggSluse1 == 0))
{
    while(verdiLDR1 < terskelLDR1)
    {
        verdiLDR1 = analogRead(pinLDR1);
        delay(100);
    }
    flaggSluse2 = 0;
    antall = antall - 1;
    delay(100);
}

//Styr servoen
if(flaggSluse2 == 1 || flaggSluse1 == 1)
{
    myservo.write(posAapen);           // Åpne bommen
    servoFlagg = 1;
    delay(100);
}
else if(flaggSluse2 == 0 && flaggSluse1 == 0)
{
    myservo.write(posLukket);           // Lukket bommen
    servoFlagg = 0;
    delay(100);
}

Serial.print("Det er foreløpig: ");
Serial.print(antall);
```



```
Serial.print(" inne i rommet ");

if (servoFlagg == 1)
{
    Serial.println("og porten er åpen!");
}

if (servoFlagg == 0)
{
    Serial.println("og porten er stengt!");
}

//Kommandoer som kan brukes for feilsøking
//Serial.print("FlaggSluse1: ");
//Serial.print(flaggSluse1);
//Serial.print(", FlaggSluse2: ");
//Serial.print(flaggSluse2);
}
```

B.9 Oppdrag 9 – Avansert optisk portal med bom og display

Programmer kombinerer den avanserte optiske portalen med en bom som går opp og ned når personer kommer og går i tillegg til at antallet i rommet til en hver tid vises på displayet.

```
// Tell antallet som går inn og ut og åpne og lukke bom
// Nils Kr. Rossing 06.07.20
// Kåre-Benjamin Hammervold Rørvik 16.7.2020

#include <TM1637Display.h>
#include <Servo.h>

Servo myservo; // Lag et objekt med navnet myservo av typen Servo

const byte PIN_CLK = 32; // Definer CLK pin og velg port D32
const byte PIN_DIO = 33; // Definer DIO pin og velg port D33

TM1637Display display(PIN_CLK, PIN_DIO); // Lag et objekt med navnet display av
typen TM1637Display

int pinLED1 = 16;
int pinLDR1 = 36;
int verdiLDR1;
```



```
int terskelLDR1 = 1500;
int flaggSlusel = 0;
int pinLED2 = 18;
int pinLDR2 = 39;
int verdiLDR2;
int terskelLDR2 = 1500;
int flaggSluse2 = 0;

int pinServo = 13;
int posLukket = 0;           // Angir posisjonen til servoen i lukket tilstand
int posAapen = 90;          // Angir posisjon til servoen i åpen tilstand
int servoFlagg = 0;

int antall = 0;

void setup()
{
    Serial.begin(115200);

    pinMode(pinLED1, OUTPUT);
    pinMode(pinLED2, OUTPUT);
    pinMode(pinServo, OUTPUT);
    digitalWrite(pinLED1, HIGH);
    digitalWrite(pinLED2, HIGH);

    myservo.attach(pinServo);      // Koble servoen til port D13
    myservo.write(posLukket);      // Lukk bommen
    delay(100);

    display.setBrightness(7);      // Setter lysstyrken på displayet.
                                   // Fra 0 (min) til 7 (max).
    display.showNumberDec(antall); //Starter Displayet (får da 0).

    delay(500);
}

void loop()
{
    verdiLDR1 = analogRead(pinLDR1);
    verdiLDR2 = analogRead(pinLDR2);

    // Registrerer innganger
```




```
if((verdiLDR1 < terskelLDR1) && (flaggSluse1 == 0) && (flaggSluse2 == 0))
{
    while(verdiLDR1 < terskelLDR1)
    {
        verdiLDR1 = analogRead(pinLDR1);
        delay(100);
    }
    flaggSluse1 = 1;
    delay(100);
}

if((verdiLDR2 < terskelLDR2) && (flaggSluse1 == 1) && (flaggSluse2 == 0))
{
    while(verdiLDR2 < terskelLDR2)
    {
        verdiLDR2 = analogRead(pinLDR2);
        delay(100);
    }
    flaggSluse1 = 0;
    antall = antall + 1;
    delay(100);
}

// registrerer utganger

if((verdiLDR2 < terskelLDR2) && (flaggSluse2 == 0) && (flaggSluse1 == 0))
{
    while(verdiLDR2 < terskelLDR2)
    {
        verdiLDR2 = analogRead(pinLDR2);
        delay(100);
    }
    flaggSluse2 = 1;
    delay(100);
}

if((verdiLDR1 < terskelLDR1) && (flaggSluse2 == 1) && (flaggSluse1 == 0))
{
    while(verdiLDR1 < terskelLDR1)
    {
```



```
    verdiLDR1 = analogRead(pinLDR1);
    delay(100);
}
flaggSluse2 = 0;
antall = antall - 1;
delay(100);
}

//Styr servoen
if(flaggSluse2 == 1 || flaggSluse1 == 1)
{
    myservo.write(posAapen);          // Åpne bommen
    delay(100);
    servoFlagg = 1;
}
else if(flaggSluse2 == 0 && flaggSluse1 == 0)
{
    myservo.write(posLukket);          // Lukket bommen
    delay(100);
    servoFlagg = 0;
}

display.showNumberDec(antall);        // Vis tall

Serial.print("Det er foreløpig: ");
Serial.print(antall);
Serial.print(" inne i rommet ");

if (servoFlagg == 1)
{
    Serial.println("og porten er åpen!");
}

if (servoFlagg == 0)
{
    Serial.println("og porten er stengt!");
}

//Kommandoer som kan brukes for feilsøking
//Serial.print("FlaggSluse1: ");
```



```
//Serial.print(flaggSluse1);  
//Serial.print(", FlaggSluse2: ");  
//Serial.print(flaggSluse2);  
}
```

B.10 Oppdrag 10 – Avansert optisk portal med bom, display og rele

Programmet kombinerer den avanserte optiske portalen med en bom som går opp og ned når personer kommer og går i tillegg til at antallet i rommet til en hver tid vises på displayet. Dessuten kobles et rele inn som kan slå på lyset i rommet når det er folk der.

```
// Tell antallet som går inn og ut og åpne og lukke bom  
// Nils Kr. Rossing 06.07.20  
// Kåre-Benjamin Hammervold Rørvik 16.7.2020
```

```
#include <TM1637Display.h>  
#include <Servo.h>
```

```
Servo myservo; // Lag et objekt med navnet myservo av typen Servo
```

```
const byte PIN_CLK = 32; // Definer CLK pin og velg port D32  
const byte PIN_DIO = 33; // Definer DIO pin og velg port D33
```

```
TM1637Display display(PIN_CLK, PIN_DIO); // Lag et objekt med navnet display av  
typen TM1637Display
```

```
int pinLED1 = 16;  
int pinLDR1 = 36;  
int verdiLDR1;  
int terskelLDR1 = 1500;  
int flaggSluse1 = 0;  
int pinLED2 = 18;  
int pinLDR2 = 39;  
int verdiLDR2;  
int terskelLDR2 = 1500;  
int flaggSluse2 = 0;
```

```
int pinRele = 14;
```

```
int pinServo = 13;  
int posLukket = 0; // Angir posisjonen til servoen i lukket tilstand  
int posAapen = 90; // Angir posisjon til servoen i åpen tilstand  
int servoFlagg = 0;
```



```
int antall = 0;

void setup()
{
    Serial.begin(115200);

    pinMode(pinLED1, OUTPUT);
    pinMode(pinLED2, OUTPUT);
    pinMode(pinServo, OUTPUT);
    pinMode(pinRele, OUTPUT);
    digitalWrite(pinLED1, HIGH);
    digitalWrite(pinLED2, HIGH);

    myservo.attach(pinServo);          // Koble servoen til port D13
    myservo.write(posLukket);          // Lukk bommen
    delay(100);

    display.setBrightness(7); // Setter lysstyrken på displayet. Fra 0 (min) til 7
(max).
    display.showNumberDec(antall); //Starter Displayet (får da 0).

    delay(500);
}

void loop()
{
    verdiLDR1 = analogRead(pinLDR1);
    verdiLDR2 = analogRead(pinLDR2);

    // Registrerer innganger

    if((verdiLDR1 < terskelLDR1) && (flaggSluse1 == 0) && (flaggSluse2 == 0))
    {
        while(verdiLDR1 < terskelLDR1)
        {
            verdiLDR1 = analogRead(pinLDR1);
            delay(100);
        }
        flaggSluse1 = 1;
        delay(100);
    }
}
```



```
if((verdiLDR2 < terskelLDR2) && (flaggSluse1 == 1) && (flaggSluse2 == 0))
{
    while(verdiLDR2 < terskelLDR2)
    {
        verdiLDR2 = analogRead(pinLDR2);
        delay(100);
    }
    flaggSluse1 = 0;
    antall = antall + 1;
    delay(100);
}

// registrerer utganger

if((verdiLDR2 < terskelLDR2) && (flaggSluse2 == 0) && (flaggSluse1 == 0))
{
    while(verdiLDR2 < terskelLDR2)
    {
        verdiLDR2 = analogRead(pinLDR2);
        delay(100);
    }
    flaggSluse2 = 1;
    delay(100);
}

if((verdiLDR1 < terskelLDR1) && (flaggSluse2 == 1) && (flaggSluse1 == 0))
{
    while(verdiLDR1 < terskelLDR1)
    {
        verdiLDR1 = analogRead(pinLDR1);
        delay(100);
    }
    flaggSluse2 = 0;
    antall = antall - 1;
    delay(100);
}

//Styr servoen
if(flaggSluse2 == 1 || flaggSluse1 == 1)
{
```



```
myservo.write(posAapen);          // Åpne bommen
delay(100);
servoFlagg = 1;
}
else if(flaggSluse2 == 0 && flaggSluse1 == 0)
{
    myservo.write(posLukket);      // Lukket bommen
    delay(100);
    servoFlagg = 0;
}

display.showNumberDec(antall);     // Vis tall

Serial.print("Det er foreløpig: ");
Serial.print(antall);
Serial.print(" inne i rommet ");

if (servoFlagg == 1)
{
    Serial.print("og porten er åpen!");
}

if (servoFlagg == 0)
{
    Serial.print("og porten er stengt!");
}

if(antall <= 0)
{
    digitalWrite(pinRele, LOW);
    Serial.println("Reléet er AV.");
}

if(antall >= 1)
{
    digitalWrite(pinRele, HIGH);
    Serial.println("Reléet er PÅ.");
}

//Kommandoer som kan brukes for feilsøking
//Serial.print("FlaggSluse1: ");
```



```
//Serial.print(flaggSluse1);  
//Serial.print(", FlaggSluse2: ");  
//Serial.print(flaggSluse2);  
}
```

B.11 Oppdrag 11 – Lesing av ID-kort informasjon

/* Programmet leser informasjon fra RFID-briken og skriver ut til monitoren

* Programmet er skrevet av Kåre-Benjamin Rørvik 08.07.20
* Programmet er videreutviklet av Nils Kr. Rossing 17.07.20
*/

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

```
#include <MFRC522.h>
```

```
const int RST_PIN = 22; // Reset
```

```
const int SS_PIN = 21; // Slave
```

```
MFRC522 mfrc522(SS_PIN, RST_PIN); // Oppretter instansen mfrc522 av typen MFRC522.
```

//Her lagres kortet sitt UID i 4 variabler. Disse brukes til å identifisere et RFID kort.

```
int ID0 = 124; // Erstatt "124" med avlest byte 0.
```

```
int ID1 = 17; // Erstatt "17" med avlest byte 1.
```

```
int ID2 = 94; // Erstatt "94" med avlest byte 2.
```

```
int ID3 = 116; // Erstatt "116" med avlest byte 3.
```

```
int RFIDflagg = 0; // Flagg som indikerer at nytt kort er lest.
```

```
void setup() {
```

```
Serial.begin(115200); // Starter seriell kommunikasjon med Baud rate på  
// 115200 bit/sekund (husk å endre i terminalen).
```

```
SPI.begin(); // Starter SPI - Initierer SPI-buss som kommuniserer  
mot RFID RC522.
```

```
mfrc522.PCD_Init(); // Starter RFID leseren.
```

```
Serial.println("***Klar til å lese kort***");
```

```
}
```



```
void loop()
{
  // Identifiser kort og les
  if (mfr522.PICC_IsNewCardPresent() && mfr522.PICC_ReadCardSerial())
  {
    // Skriv ut kortinformasjon.
    Serial.print("Identifisert kort med ID: ");
    Serial.print(mfr522.uid.uidByte[0]); // Skriv ut ID byte 0.
    Serial.print(" ");
    Serial.print(mfr522.uid.uidByte[1]); // Skriv ut ID byte 1.
    Serial.print(" ");
    Serial.print(mfr522.uid.uidByte[2]); // Skriv ut ID byte 2.
    Serial.print(" ");
    Serial.print(mfr522.uid.uidByte[3]); // Skriv ut ID byte 3.
    Serial.println(" ");
    mfr522.PICC_HaltA(); // Stopp dersom kortet er uendret.
    RFIDflagg = 1; // Sett et flagg som indikerer at nytt kort er lest.
  }

  // Sjekk at kortet er godkjent og ønsk velkommen inn
  if((mfr522.uid.uidByte[0]==ID0) &&
      (mfr522.uid.uidByte[1]==ID1) &&
      (mfr522.uid.uidByte[2]==ID2) &&
      (mfr522.uid.uidByte[3]==ID3) && RFIDflagg == 1)
  {
    Serial.println("Velkommen inn");
  }
  RFIDflagg = 0; // Resett flagg som forteller at kortet er sjekket
}
```

B.12 Oppdrag 12 – Lesing av ID-kort informasjon og åpning av bom

Programmet leser ID-kortet og åpner bommen dersom koden godkjennes.

```
/* Programmet leser informasjon fra RFID-briken og skriver ut til monitoren
 * Programmet er skrevet av Kåre-Benjamin H. Rørvik 08.07.20
 * Programmet er videreutviklet av Nils Kr. Rossing 17.07.20
 */
```

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




```
#include <MFRC522.h>
#include <Servo.h>

const int RST_PIN = 22; // Reset.
const int SS_PIN = 21;  // Slave.

MFRC522 mfrc522(SS_PIN, RST_PIN); // Oppretter instansen mfrc522 av typen MFRC522.
Servo myservo;      // Lag et objekt med navnet myservo.
                    // Hele 12 slike objekter kan lages ved hvert kort.

int posLukket = 0; // variabel som holde lokket posisjon for bom i grader.
int posApen = 90;  // Variabel som holde åpen posisjon for bom i grader.
int ID0 = 124;     // Erstatt "124" med avlest byte 0.
int ID1 = 17;      // Erstatt "17" med avlest byte 1.
int ID2 = 94;      // Erstatt "94" med avlest byte 2.
int ID3 = 116;     // Erstatt "116" med avlest byte 3.
int RFIDflagg = 0; // Flagget som indikerer at nytt kort er lest.

void setup() {
  Serial.begin(115200); // Starter seriell kommunikasjon med Baud rate på 115200
                        // bit/sekund (husk å endre i terminalen).
  SPI.begin();         // Starter SPI - Initierer SPI-buss som kommuniserer mot
                        // RFID RC522.
  mfrc522.PCD_Init();  // Starter RFID leseren.
  myservo.attach(13);  // attaches the servo on pin 13 to the servo object.
  myservo.write(posLukket); // be servoen om å gå til posisjon lukket.
  Serial.println("***Klar til å lese kort***");
}

void loop()
{
  // Identifiser kort og les
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial())
  {
    // Skriv ut kortinformasjon
    Serial.print("Identifisert kort med ID: ");
    Serial.print(mfrc522.uid.uidByte[0]); Serial.print(" "); // Skriv ut ID byte 0.
    Serial.print(mfrc522.uid.uidByte[1]); Serial.print(" "); // Skriv ut ID byte 1.
    Serial.print(mfrc522.uid.uidByte[2]); Serial.print(" "); // Skriv ut ID byte 2.
    Serial.print(mfrc522.uid.uidByte[3]); Serial.println(" "); // Skriv ut ID byte 3.
    mfrc522.PICC_HaltA(); // Stopp dersom kortet er uendret.
    RFIDflagg = 1;       // Sett et flagget som indikerer at nytt kort er lest.
  }
}
```



```
}

// Sjekk at kortet er godkjent og ønsk velkommen inn
if((mfrc522.uid.uidByte[0]==ID0) &&
    (mfrc522.uid.uidByte[1]==ID1) &&
    (mfrc522.uid.uidByte[2]==ID2) &&
    (mfrc522.uid.uidByte[3]==ID3) && RFIDflagg == 1)
{
    Serial.println("Velkommen inn");
    myservo.write(posApen);           // be servoen om å gå til posisjon pos.
    delay(5000);                      // Stå i posisjon 0 i 5 sek.
    myservo.write(posLukket);         // be servoen om å gå til posisjon pos.
}

RFIDflagg = 0; // Resett flagg som forteller at kortet er sjekket.

}
```

B.13 Oppdrag 13 – Endelig sammenstilling – Løsning av det totale oppdraget

Her settes de ulike delene av oppdraget sammen for å oppfylle den endelige målsetningen med prosjektet. Legg spesielt merke til hvor enkelt hovedprogrammet blir når vi bruker funksjoner:

```
// Total Løsning med RFID implementert.
// Kåre-Benjamin Hammervold Rørvik og Nils Kr. Rossing 23.7.2020

#include <SPI.h>
#include <TM1637Display.h>
#include <MFRC522.h>
#include <Servo.h>

//Setter alle PIN som blir brukt på ESP32:
const int pinRele = 14;
const int pinServo = 13;
const int pinLED2 = 18;
const int pinLDR2 = 39;
const int pinLED1 = 16;
const int pinLDR1 = 36;
const int SS_PIN = 21; // Slave hos RFID.
const int RST_PIN = 22; // Reset hos RFID.
const int PIN_CLK = 32; // CLK hos Displayet.
const int PIN_DIO = 33; // DIO hos Displayet.

// Lager objekter
TM1637Display display(PIN_CLK, PIN_DIO); // Lag et objekt med navnet display av
typen TM1637Display.
MFRC522 mfrc522(SS_PIN, RST_PIN); // Oppretter instansen mfrc522 av typen MFRC522.
```



```
Servo myservo; // Lag et objekt med navnet myservo.
               // Hele 12 slike objekter kan lages ved hvert kort.

// Her lagres kortet sitt UID i 4 variabler. Disse brukes til å identifisere et
// RFID kort.
// Det kan legges inn 2 kort, hvor begge kan brukes lagret.

// RFID-Kort 0
int ID0_0 = 124; // Erstatt "124" med avlest byte 0.
int ID0_1 = 17;  // Erstatt "17" med avlest byte 0.
int ID0_2 = 94;  // Erstatt "94" med avlest byte 0.
int ID0_3 = 116; // Erstatt "116" med avlest byte 0.

//RFID-Kort 1
int ID1_0 = 185; // Erstatt "185" med avlest byte 0.
int ID1_1 = 191; // Erstatt "191" med avlest byte 1.
int ID1_2 = 45;  // Erstatt "45" med avlest byte 2.
int ID1_3 = 86;  // Erstatt "86" med avlest byte 3.

// LDR Variabler
int verdiLDR1; //Verdien som leses ut av LDR1.
int terskelLDR1 = 1500; //Terskelen for målingene hos LDR1.
int verdiLDR2; //Verdien som leses ut av LDR2.
int terskelLDR2 = 1500; //Terskelen for målingene hos LDR2.

// Flagg som styrer programflyten, disse stilles til 0 eller 1 for å iverksette
// endringer i programmet.
int flaggSluse1 = 0; // Satt til 1 når sluse 1 skal til å åpne.
int flaggSluse2 = 0; // Satt til 1 når sluse 2 skal til å åpne.
int RFIDflagg = 0;   // Satt til 1 når nytt kort er lest.
int nokkelFlagg = 0; // Satt til 1 når RFID er lest og godkjent.
int releFlagg = 0;   // Satt til 1 når reléet er aktivt.
int servoFlagg = 0;  // Satt til 1 når porten er åpen.

// Diverse variabler
int antall = 0; // Antall personer i rommet.
int posLukket = 0; // Angir posisjonen til servoen i lukket tilstand.
int posAapen = 90; // Angir posisjon til servoen i åpen tilstand.

// Her er variablene som definerer ordene som blir skrevet på 7-segment displayet.
// De er definert for å danne ønskede bokstaver på 7-segment displayet.
// Bokstavene er laget ved å skru på eller av ønskede segmenter ...
// i displayet for å danne bokstaver.

//
//      A
//      ---
//  F |   | B
//      -G-
```



```
// E |   | C
//    ---
//    D

// Lager et array av typen byte (byte = 8bit).
// Dette brukes ofte for å styre 7-segment display.
// Merk: Displayene har enten 7 eller 8 LED segment, som akkurat går opp i en byte.
const byte OPEN[] = {
    SEG_A | SEG_B | SEG_C | SEG_D | SEG_E | SEG_F,    // O
    SEG_A | SEG_B | SEG_E | SEG_F | SEG_G,           // P
    SEG_A | SEG_D | SEG_E | SEG_F | SEG_G,           // E
    SEG_C | SEG_E | SEG_G,                           // n
};

const byte Err[] = {
    SEG_A | SEG_D | SEG_E | SEG_F | SEG_G,           // E
    SEG_E | SEG_G,                                   // r
    SEG_E | SEG_G,                                   // r
    0,                                                // Blir ikke belyst...
};

void setup()
{
    programInit();
}

void loop()
{
    lesLDR();
    lesOgGodkjen();
    tellPasseringer();
    servoStyring();
    skrivTilDisplay();
    styrRele();
    skrivTilMonitor();
}

void lesLDR()
// Her leses ut verdiene på LDR1 og LDR2.
// Disse lagres til verdiLDR1 og verdiLDR2.
{
    verdiLDR1 = analogRead(pinLDR1);
    verdiLDR2 = analogRead(pinLDR2);
}

void programInit()
// Inneholder initialiseringen av programmet.
// Starter kommunikasjon med terminalen, SPI, RFID,
// pinnene sin orientering (de er INPUT per default),
// starter servoen og displayet.
```



```
{
  Serial.begin(115200); // Starter seriell kommunikasjon med Baud rate på 115200
bit/sekund (husk å endre i terminalen).
  SPI.begin();          // Starter SPI - Initierer SPI-buss som kommuniserer mot
RFID RC522.
  mfrc522.PCD_Init();    // Starter RFID leseren.

  pinMode(pinLED1, OUTPUT);
  pinMode(pinLED2, OUTPUT);
  pinMode(pinServo, OUTPUT);
  pinMode(pinRele, OUTPUT);
  digitalWrite(pinLED1, HIGH);
  digitalWrite(pinLED2, HIGH);

  myservo.attach(pinServo);          // Koble servoen til port D13 .
  myservo.write(posLukket);          // Lukk bommen.
  delay(100);

  display.setBrightness(7); // Setter lysstyrken på displayet. Fra 0 (min) til
7 (max).
  Serial.println("***Klar til å lese kort***");
  display.showNumberDec(antall); //Starter Displayet (får da 0).

  delay(500);
}

void lesOgGodkjen()
// Leser RFID-kort, skriver ut til monitor og sjekker om kortet er registrert.
// Det avleste kortet blir sjekket opp mot registrerte kort i minnet (ID1 og ID2).
// Det vil deretter å indikere om kortet som er tilkoblet har tilgang.

// Styres internt av RFIDflagg (1 når kort er kompatibelt og avlest).
// Påvirker resten av programmet med nøkkelFlagg (1 låser opp porten for utsiden).
{
  // Identifiser kort og les
  if (mfrc522.PICC_IsNewCardPresent() && mfrc522.PICC_ReadCardSerial())
  {
    // Skriv ut kortinformasjon
    Serial.print("Identifisert kort med ID: ");
    Serial.print(mfrc522.uid.uidByte[0]);
    Serial.print(" "); // Skriv ut ID byte 0.
    Serial.print(mfrc522.uid.uidByte[1]);
    Serial.print(" "); // Skriv ut ID byte 1.
    Serial.print(mfrc522.uid.uidByte[2]);
    Serial.print(" "); // Skriv ut ID byte 2.
    Serial.print(mfrc522.uid.uidByte[3]);
    Serial.println(" "); // Skriv ut ID byte 3.
    mfrc522.PICC_HaltA(); // Stopp dersom kortet er uendret.
    RFIDflagg = 1; // Sett et flagg som indikerer at nytt kort er lest.
  }
}
```



```
}

// Sjekk at kortet er godkjent og ønsk velkommen inn
if(((mfr522.uid.uidByte[0]==ID0_0)|| (mfr522.uid.uidByte[0]==ID1_0))&&
    ((mfr522.uid.uidByte[1]==ID0_1)|| (mfr522.uid.uidByte[1]==ID1_1)) &&
    ((mfr522.uid.uidByte[2]==ID0_2)|| (mfr522.uid.uidByte[2]==ID1_2)) &&
    ((mfr522.uid.uidByte[3]==ID0_3)|| (mfr522.uid.uidByte[3]==ID1_3)) && RFID-
flagg == 1)
{
    if (RFIDflagg = 1)
    {
        nokkelFlagg = 1;
        Serial.println("*** Kort er registrert og godkjent! ***");
    }
    RFIDflagg = 0; // Resett flagg som forteller at kortet er sjekket.
}

}

void tellPasseringer()
// Leser av LDR for å vurdere om porten skal åpnes eller lukkes.
// Teller også opp eller ned antall variablen.
// nokkelFlagg må være satt til 1 for at porten skal kunne åpnes via LDR1 (utsiden).
// Imidlertid kan porten åpnes åpnes fra innsiden uten at nokkelFlagget er satt
...
// altså porten kan åpnes fra innsiden uten at døren er låst opp.
{
    // Registrerer innganger

    if((verdiLDR1 < terskelLDR1) && (flaggSlusel == 0) && (flaggSluse2 == 0) && (nok-
kelFlagg == 1))
    {
        while(verdiLDR1 < terskelLDR1)
        {
            verdiLDR1 = analogRead(pinLDR1);
            delay(100);
        }
        flaggSlusel = 1;
        delay(100);
    }

    if((verdiLDR2 < terskelLDR2) && (flaggSlusel == 1) && (flaggSluse2 == 0) && (nok-
kelFlagg == 1))
    {
        while(verdiLDR2 < terskelLDR2)
        {
            verdiLDR2 = analogRead(pinLDR2);
            delay(100);
        }
        flaggSlusel = 0;
    }
}
```



```
    antall = antall + 1;
    RFIDflagg = 0;
    nokkelFlagg = 0;
    Serial.println("Brukeren har åpnet døren, og adgangen er brukt opp!");
    delay(100);
}

// registrerer utganger

if((verdiLDR2 < terskelLDR2) && (flaggSluse2 == 0) && (flaggSluse1 == 0))
{
    while(verdiLDR2 < terskelLDR2)
    {
        verdiLDR2 = analogRead(pinLDR2);
        delay(100);
    }
    flaggSluse2 = 1;
    RFIDflagg = 0;
    nokkelFlagg = 0;
    delay(100);
}

if((verdiLDR1 < terskelLDR1) && (flaggSluse2 == 1) && (flaggSluse1 == 0))
{
    while(verdiLDR1 < terskelLDR1)
    {
        verdiLDR1 = analogRead(pinLDR1);
        delay(100);
    }
    flaggSluse2 = 0;
    antall = antall - 1;
    delay(100);
}
}

void servoStyring()
// Styrer direkte servoen avhengig av flaggSluse variablene.
// Åpner og lukker bommen med antall grader gitt av posAapen og posLukket.
{
    //Styr servoen
    if((flaggSluse2 == 1 || flaggSluse1 == 1))
    {
        myservo.write(posAapen); // Åpne bommen.
        delay(100);
        servoFlagg = 1;
    }
    else if(flaggSluse2 == 0 && flaggSluse1 == 0)
    {
        myservo.write(posLukket); // Lukk bommen
    }
}
```



```
    delay(100);
    servoFlagg = 0;
}

void skrivTilDisplay()
// Skriver til Displayet avhengig av teller variablen.
// Setter også OPEN når døren er låst opp samt Err når et kort uten tilgang ble
brukt.
{
    if(nokkelFlagg == 1 && !(flaggSluse2 == 1 && flaggSluse1 == 0))
    {
        display.setSegments(OPEN);
    }

    else{
        if(RFIDflagg == 1)
        {
            display.setSegments(Err);
        }
        else
        {
            display.showNumberDec(antall);          // Vis tall
        }
    }
}

void styrRele()
// Reléet er av når ingen er i rommet (eller negativt antall personer...),
// skrur på når minst én person er inne. Oppdaterer relé flagget.
{
    if(antall <= 0)
    {
        digitalWrite(pinRele, LOW);
        releFlagg = 0;
    }

    if(antall >= 1)
    {
        digitalWrite(pinRele, HIGH);
        releFlagg = 1;
    }
}

void skrivTilMonitor()
// Sender mesteparten av kommunikasjon med monitor.
// Brukes hovedsaklig til debugging (feilsøking)
// Den hjelper med feilsøking og prototyping.
// Kan fjernes når programmet er ferdigprogramert, eller skrus av.
```




```
{
  Serial.print("Det er foreløpig: ");
  Serial.print(antall);
  Serial.print(" inne i rommet ");
  Serial.print(" RFID lest: ");
  Serial.print(RFIDflagg);
  Serial.print(" kort er akseptert: ");
  Serial.print(nokkelFlagg);

  if (servoFlagg == 1)
  {
    Serial.print(" og porten er åpen!");
  }

  if (servoFlagg == 0)
  {
    Serial.print(" og porten er stengt!");
  }

  if (releFlagg == 0)
  {
    Serial.println("Reléet er AV.");
  }

  if (releFlagg == 1)
  {
    Serial.println("Reléet er PÅ.");
  }

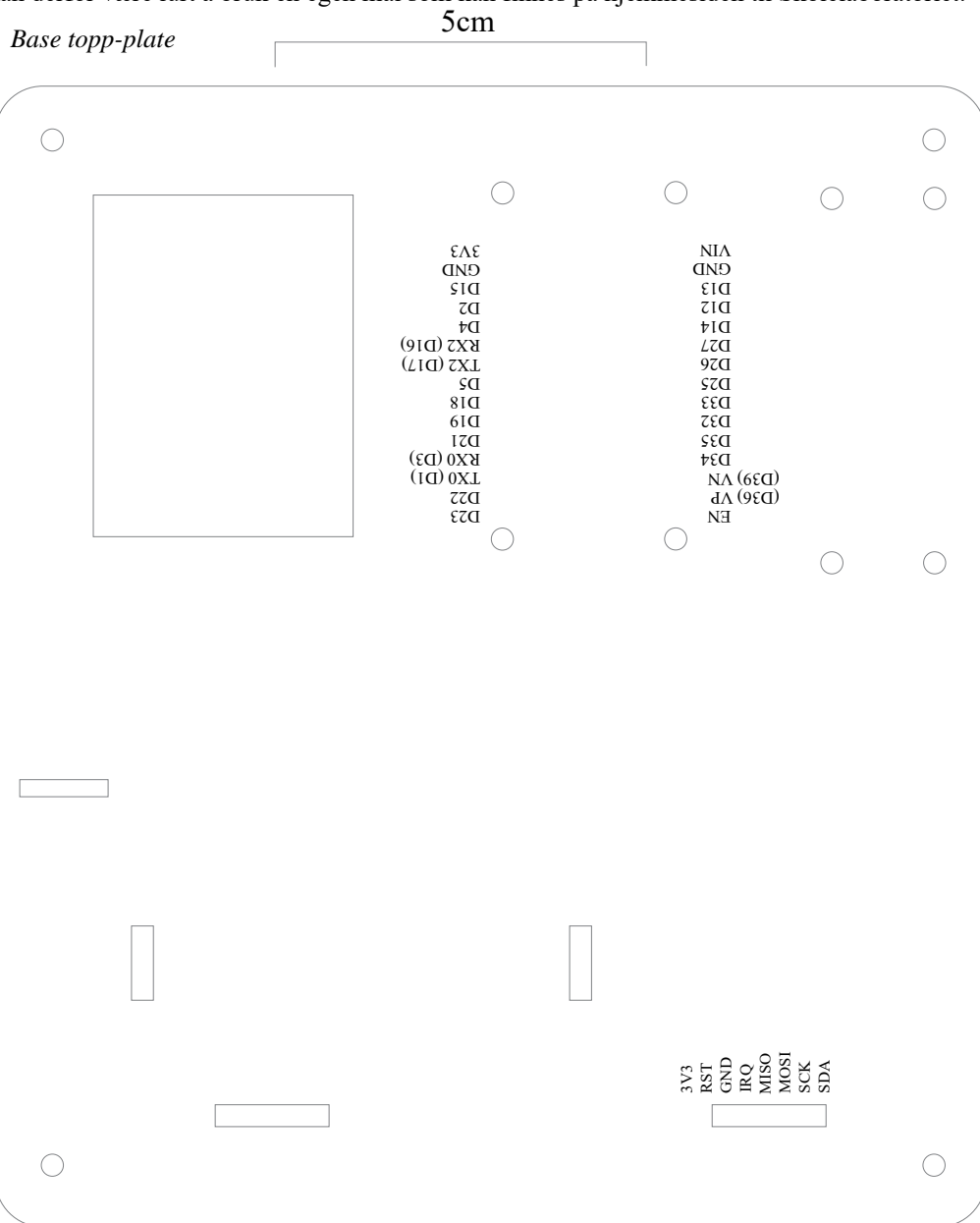
  //Kommandoer som kan brukes for feilsøking ...
  //Serial.print("FlaggSlusel: ");
  //Serial.print(flaggSlusel);
  //Serial.print(", FlaggSluse2: ");
  //Serial.println(flaggSluse2);
}
```



Vedlegg C Maler for laserkutting

C.1 Målejigg

Vær oppmerksom på at målene er kritisk da de ulike komponentene skal gli rett inn i hullene. Det kan derfor være lurt å bruk en egen mal som kan finnes på hjemmesiden til Skolelaboratoriet.

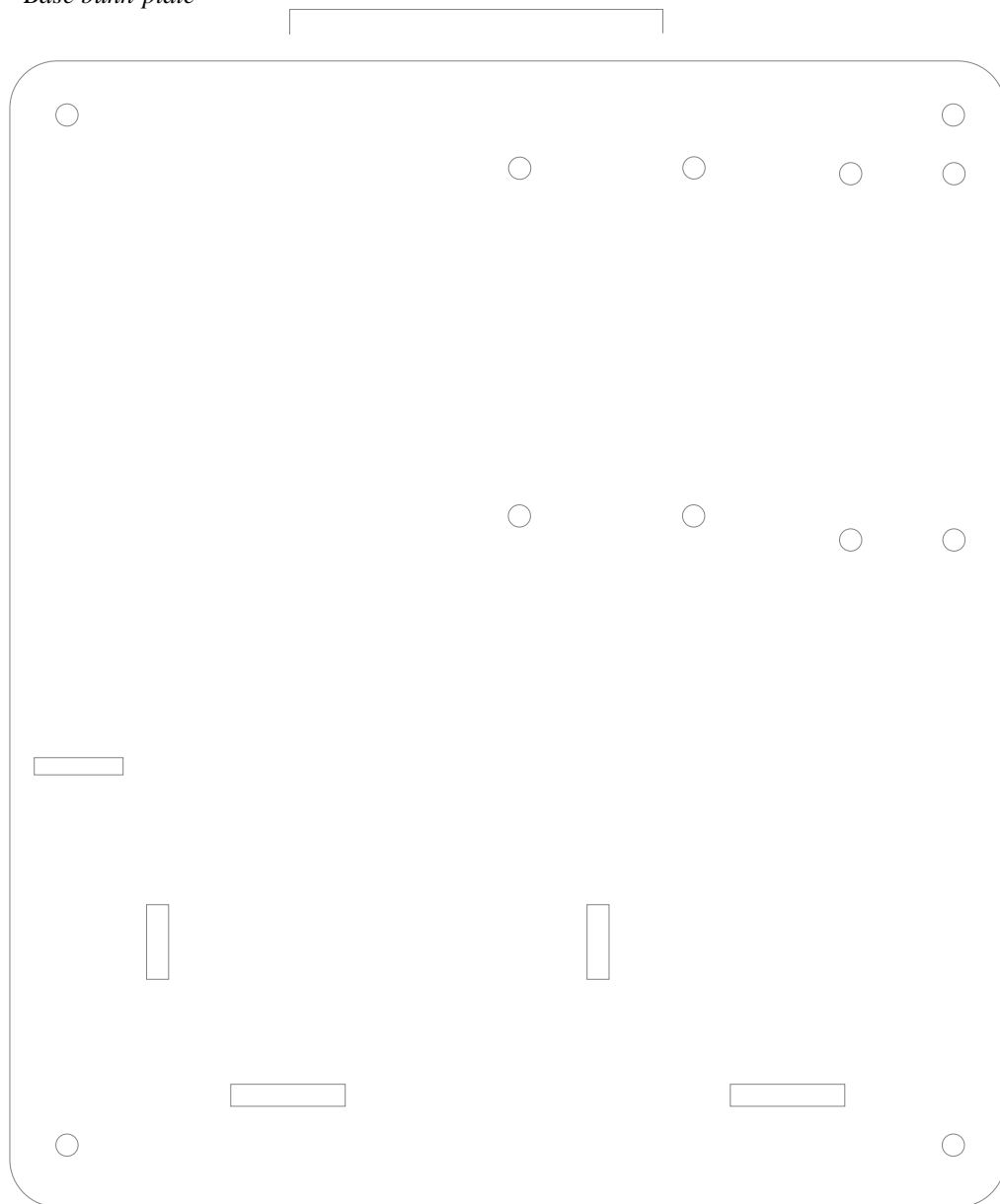




Baseplaten og veggene er skåret i MDF 3.1 mm reell tykkelse

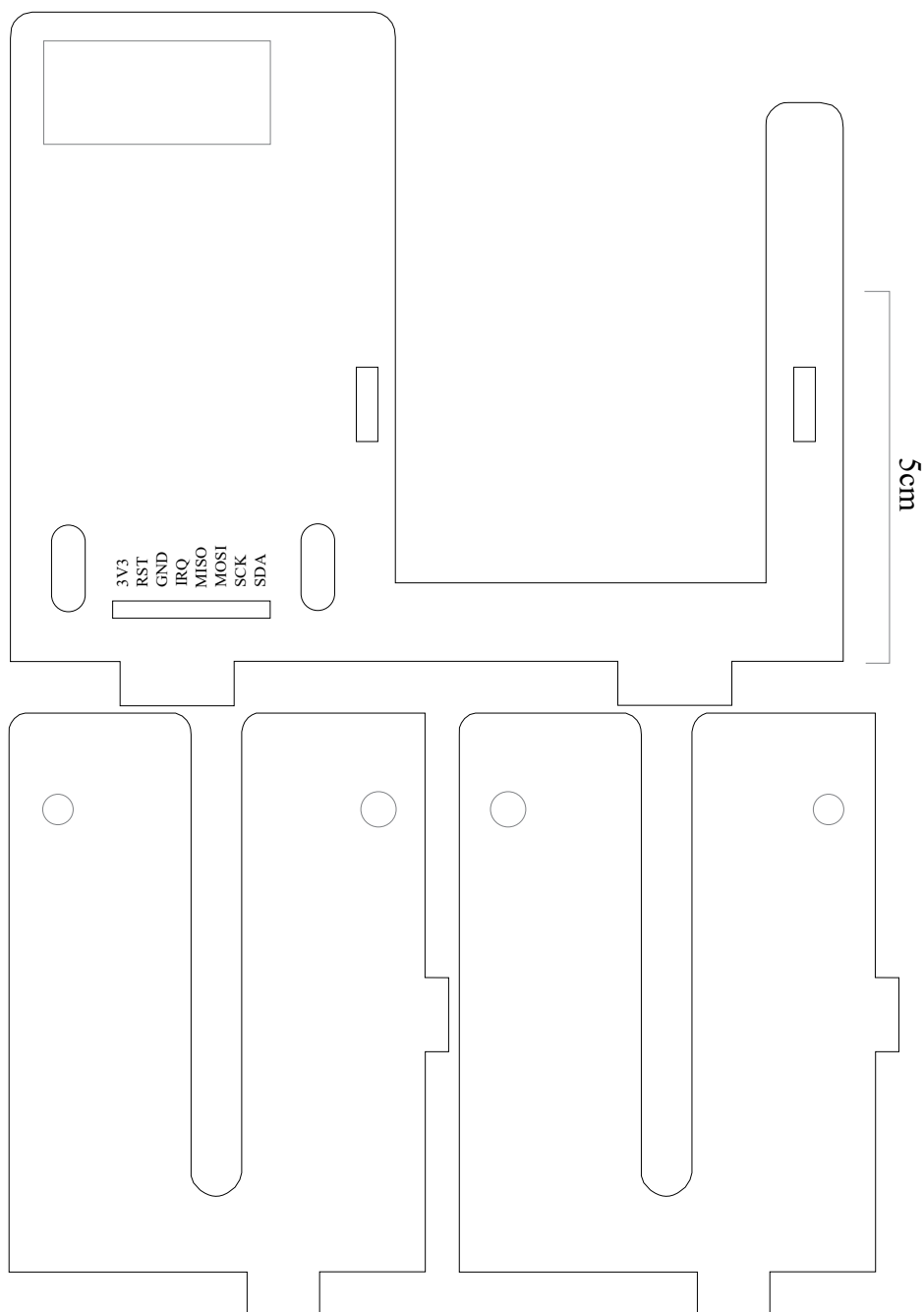
Base bunn-plate

5cm



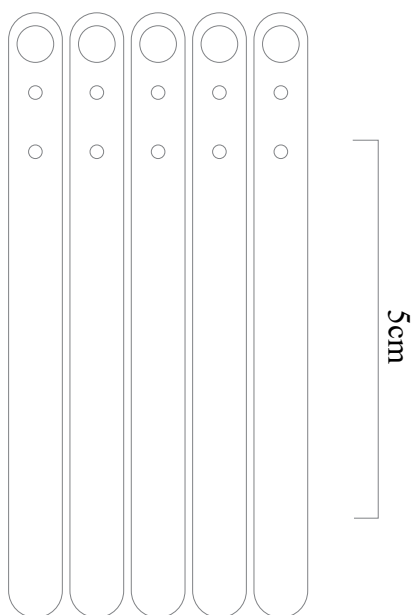


Veggene – sjekk at 5 cm har riktig lengde. Veggene er skåret i 3.1 mm MDF reell tykkelse





Bommen er skåret i 2 mm MDF





Vedlegg D Administrative tiltak

D.1 Oversikt over arbeidsoppgaver byggesett

Følgende arbeidsoperasjoner må gjøres for å lage byggesett:

- Skjære ut 30 stk basisplate på laserkutter
- Skjære ut 30 stk veggstykker (3 deler) på laserkutter
- Skjære ut 30 stk bommer på laserkutter
- Skrive ut 120 stk føtter på 3D-printer
- Lodde opp 30 stk ESP32 med 2 stk hylselister
- Klippe opp og tilpasse lengden på 3 pins stiftlist for servo
- Klippe opp og lodde på 4 pins stiftlist på display TM1637
- Klippe opp og lodde på 7 pins hylselist på RFID RC522
- Montere 30 stk krympestrømpe (skjerming) på LDR
- Montere krympestrømpe på 120 stk to pins kabel hann → hunn
- Tilpasse beina på 60 stk hvite lysdioder (klippes kortere, langt og kort)
- Følgende komponenter legges i pose med lås
 - 1 stk baseplate
 - 1 sett med veggplater (3 stk)
 - 1 bom
 - 4 stk M3 12 mm skruer m/ M3 mutter
 - 4 stk M2,5 12 mm skruer m/ M2,5 mutter
 - 4 stk M3 16 mm skruer m/ M3 mutter
 - 2 stk 220Ω 0,25W 5% motstand
 - 2 stk 10kΩ 0,25W 5% motstand
 - 2 stk LDR m/krympestrømpe
 - 2 stk hvite lysdioder
 - 1 stk ESP WROOM 32 DEVKIT V1 30 pins m/påloddet 30 pins hylsekontakter
 - 1 display TM1637 m/ 4 pins stiftlist
 - 1 rele, 1 polet
 - 1 stk RFID RC522 m/påloddet 7 pins hylsekontakt
 - 1 stk micro servo SG90 m/ 3 pins stiftlist og med plastdel for montasje av bom
 - 1 stk mini koblingsbrett



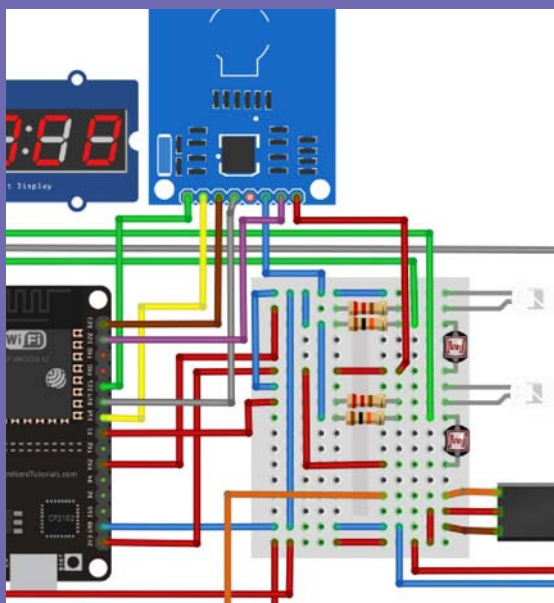
- 4 stk doble jumpere med krypestrømpe på den ene siden
- Div jumpere (kan ev. legges ved utenom)

D.2 Legges ut på hjemmesiden til Skolelaboratoriet

Følgende filer legges ut på hjemmesiden til Skolelaboratoriet (<https://www.ntnu.no/skolelab/bla-hefteserie>):

- Arbeidshefte
 - Arbeidshefte
 - Løsningsforslag legges ikke ut i første omgang
- Programvare
 - Testprogram for TM1637
 - Testprogram for RFID RC511
- Biblioteker
 - Bibliotek til TM1637 for ESP32
 - Bibliotek til RFID RC522
 - Bibliotek micro servo for ESP32





Heftet er et kurshefte laget som en videreføring av kursmodulen: *Arduino – grunnkurs programmering*, og er utarbeidet spesielt med tanke på å brukes overfor yrkesfagelever. Øvringsopplegget som kurset tilbyr er rettet mot problemstillinger som kan være relevant for yrkesfag Elektro og Bygg og anlegg. Likevel ønsker vi å arbeide på grunnplanet der elektronikken ikke er mer kompleks enn at det er mulig å forstå de ulike elementene som blir brukt.

Vi har også valgt å forlate Arduino UNO og gå over til ESP32 som inkluderer WiFi og Bluetooth. ESP32, og som gir mikrokontrolleren mulighet til å fungere som server koblet opp mot Internett, en sensornode. Kretsen er derfor spesielt egnet til *Internett of Things* (IoT). Samtidig kan den programmeres omtrent som en Arduino UNO ved hjelp av det samme programmeringsverktøyet hvilket gjør at overgangen føles overkommelig. Vi kommer ikke til å benytte WiFi eller Bluetooth i denne sammenheng, men legger opp til at det blir hovedtemaet i en påfølgende modul.

Nils Kr. Rossing

Dosent ved Skolelaboratoriet
E-post: nils.rossing@ntnu.no

Kåre-Benjamin Rørvik

Bachelor Inst. for elektroniske systemer, NTNU
E-post: kbrorvik@stud.ntnu.no



Trondheim

Skolelaboratoriet

for matematikk, naturfag
og teknologi

Tlf. 73 55 11 43

<https://www.ntnu.no/skolelab>

Institutt for fysikk

Institutt for elektroniske systemer