Bacheloroppgave

NTNU
Norges teknisk-naturvitenskapelige universitet
Fakultet for samfunns- og utdanningsvitenskap
Institutt for lærerutdanning

Kandidatnummer: 10010, 10016, 10017

Arduino

Et FoU-arbeid med fokus på læring om Arduino på Vg1 EL og TIF

Bacheloroppgave i yrkesfaglærerutdanningen
Veileder: Jon Sverre Hårberg og Arne Midjo
Juni 2022

NTNU
Norwegian University of
Science and Technology

Kandidatnummer: 10010, 10016, 10017

Arduino

Et FoU-arbeid med fokus på læring om Arduino på Vg1 EL og TIF

Bacheloroppgave i yrkesfaglærerutdanningen
Veileder: Jon Sverre Hårberg og Arne Midjo
Juni 2022

Norges teknisk-naturvitenskapelige universitet
Fakultet for samfunns- og utdanningsvitenskap
Institutt for lærerutdanning

NTNU
Kunnskap for en bedre verden

Sammendrag

Samfunnet er inne i en tid hvor vi står ovenfor en stor teknologisk utvikling. Dette vil føre til at framtidens fagarbeider trenger kompetanse som kan møte samfunnets krav og behov (NOU 2020:2).

Gjennom vårt FoU-arbeid i denne bacheloroppgaven har vi funnet ut at flere elever opplever det som utfordrende å finne faglitteratur om Arduino. Med bakgrunn i dette startet vi utviklingen av et kompetansehefte for elever ved Vg1 elektro- og datateknologi, EL, og Vg1 teknologi- og industrifag, TIF. Dette er et kompetansehefte som er ment for å kunne brukes innen den yrkesfaglige opplæringen, som inneholder teori, differensierte oppgaver, fordypningsoppgaver og tverrfaglige oppgaver.

I vår yrkesfaglig fordypning har fokuset vært å få fram hva Arduino er, og hvorfor programmering vil være en så ettertraktet kompetanse i fremtiden. Vi kom fram til følgende problemstilling for den yrkesfaglige fordypningen;

«Hva er Arduino, og hvorfor har programmering blitt så sentralt innen de teknologiske fagene EL og TIF?»

I vår profesjonsfaglige del har vi hatt som fokus å teste ut kompetanseheftet som vi har utviklet, og hvordan vi kan fremme læring om Arduino og programmering. Vi kom fram til følgende problemstilling i den profesjonsfaglige delen;

«Hvordan legge til rette for læring om Arduino på Vg1 EL og TIF?»

Vi utviklet et eget undervisningsopplegg som inneholdt elementer fra både den yrkesfaglige- og den profesjonsfaglige delen, for å teste ut noen av oppgavene fra kompetanseheftet. Selve aksjoneringen ble gjennomført ved en videregående skole innen programfaget Vg1 elektro- og datateknologi.

Abstract

In the present time is society facing a major technological development. This will lead to the skilled workers of the future needing competence that can meet society's demands and needs (NOU 2020: 2). Through our R&D work in this bachelor thesis, we have found results that many students find it challenging to find subject content about Arduino. Based on this, we started the development of a competence booklet for students at Vg1 electrical and computer technology, EL, and Vg1 technology and industry subjects, TIF. This is a competence booklet that is intended to be used in vocational education, which contains theory, differentiated assignments, specialization assignments and interdisciplinary assignments.

In our professional specialization, the focus has been on revealing what Arduino is, and why programming will be such a popular competence in the future. We have decided to use this following issue for the vocational bachelor task;

"What is Arduino, and why has programming become so central in the EL and TIF technology disciplines?"

In our professional part, our focus has been to check the competence booklet that we have developed, and how we can promote learning about Arduino and programming. We came decided to use this following issue in the professional part;

"How to facilitate learning about Arduino at Vg1 EL and TIF?".

We developed a separate teaching program that contained elements from both the vocational and the professional part, to check some of the tasks from the competence booklet. The actual teaching arrangements was tested at an upper secondary school within the program subject Vg1 electrical and computer technology.

Forord

Denne bacheloroppgaven er vår avsluttende oppgave i utdannelsesreisen ved NTNU, Norges teknisk- naturvitenskapelige universitet i Trondheim, yrkesfaglærerutdanningen 2019-2022. Igjennom oppgavens tema som har vært Arduino, har vi utarbeidet et kompetansehefte for elever og lærere på Vg1 elektro- og datateknologi (EL) og teknologi- og industrifag (TIF). Arbeidet med denne oppgaven har foregått i tidsrommet høsten 2021 til våren 2022. Samarbeidet har vært avgjørende for resultatet i denne besvarelsen. Ukentlige møter og studiesamlinger i Trondheim har bidratt til et viktig samhold i vårt team.

Vi vil takke vår veileder Jon Sverre Hårberg i profesjonsfaglig fordypning og Arne Midjo i yrkesfaglig fordypning, for veiledning og støtte i denne bacheloroppgaven. Vi må også benytte anledningen til å rette en stor takk til alle ansatte og gjesteforelesere ved NTNU, som har vært undervisningsansvarlige og delaktige i fag som har omfattet studiet. Vi aksjonerte ved en videregående skole, og har fått god hjelp av Halvor Hove og Bjørnar Mortensen Vik fra NDLA. Til slutt ønsker vi å rette en stor takk til familie, kollegaer og venner som har vært en del av denne utdannelsesreisen i 3 år.

Takk!

Trondheim, juni 2022.

Innhold

Figurliste

1.0 Innledning

I forbindelse med fagfornyelsen som trådte i kraft høsten 2020, har læreplanene og fagene gjennomgått både store og små endringer (Kunnskapsdepartementet, 2022). Vg1 elektrofag har nytt navn, elektro og datateknologi. Med dette skiftet kan vi høre at det har skjedd en endring. Det skal legges mere vekt på å møte framtidige samfunnsbehov, fordi teknologien er fremtredende i samfunnet. Dette krever at fremtidens fagarbeidere har kunnskaper innenfor programmering (NOU 2020:2). På Vg1 har det etter fagfornyelsen blitt tatt i bruk Arduino, som er en plattform for prototyping av elektronikk, basert på program- og maskinvare med åpen kilde.

Vg1 teknologi- og industrifag har også gjennomgått endringer i forbindelse med fagfornyelsen (Kunnskapsdepartementet, 2022). I fagenes relevans og verdier står det blant annet at det skal legges vekt på å utvikle selvstendige og omstillingsdyktige fagarbeidere, som har en grunnleggende forståelse for blant annet programmering og elektrofag. Det er her vi ser vårt snitt i å få til et samarbeidsprosjekt mellom fagene elektro og datateknologi, og teknologi og industrifag. Begge disse yrkesretningene skal utvikle morgendagens fagarbeidere, med teknologiske kunnskaper innen programmering.

I starten på denne bacheloroppgaven søkte vi internett og forskjellige læringsressurser etter en god tilnærming til emnet Arduino. Det finnes svært mye, samtidig fant vi ingen god samling for nybegynnerstadiet. På grunn av dette og undersøkelser vi gjorde i en Vg1 EL klasse produserte vi et kompetansehefte om Arduino for elever og lærere. Dette kompetanseheftet er en samling av teori og oppgaver, med forslag på hvordan man kan oppnå kompetansemålene og pensum innenfor programmering for Vg1 EL og TIF.

1.1 Oppgavens tema

Temaet for denne bacheloroppgaven vil være Arduino. Vi har en yrkesfaglig del, hvor vi går i bredden innen Arduino, og en profesjonsfaglig del hvor vi har planlagt et undervisningsopplegg for en Vg1 elektro- og datateknologi klasse.

Det yrkesfaglige temaet i denne oppgaven er programmering og Arduino, og hvorfor dette har blitt så sentralt i forhold til framtidens samfunnsbehov. Videre i besvarelsen har vi sett

nærmere på hvordan Arduino kan brukes innenfor den videregående opplæringen for å fremme læring om programmering. I vårt FoU-arbeid har vi gjennomført en aksjonering hos en Vg1 EL klasse som testet ut kompetanseheftet. Læringsarbeidet under aksjoneringen var basert på en sosiokulturell læringsteori (Sylte, 2018, s. 158-159).

1.2 Problemstilling

Da vi startet arbeidet med bacheloroppgaven ble vi veldig raskt enige om at Arduino var det vi ville forske på. Dette på grunn av at læreplanene innen EL og TIF tar for seg programmering. Arduino har blitt et mye omdiskutert programmeringsverktøy etter at fagfornyelsen kom høsten 2020. Ved å utforske hvorfor programmering har blitt en sentral del for de teknologiske fagene førte dette til at vi kunne se behovet for et grunnleggende system for opplæring om Arduino.

Vi har to forskjellige problemstillinger, en som gjelder for den yrkesfaglige delen og en for den profesjonsfaglige delen.

Problemstilling innen den yrkesfaglige delen;

«Hva er Arduino, og hvorfor har programmering blitt så sentralt innen de teknologiske fagene EL og TIF?»

Problemstilling innen den profesjonsfaglige delen;

«Hvordan legge til rette for læring om Arduino på Vg1 EL og TIF?»

Sammenhengen mellom oppgavedelene kobles gjennom breddeforståelsen av Arduino i den yrkesfaglige delen og ble grunnlaget for faginnholdet i kompetanseheftet. Vi utarbeidet et undervisningsopplegg basert på innhold fra kompetanseheftet, som ble brukt i forbindelse med aksjoneringen gjennom vårt FoU-arbeid.

1.3 Avgrensing

Innenfor den yrkesfaglige delen har vi skrevet mer i bredden enn i dybden, dette på bakgrunn av den profesjonsfaglige delen som omhandler en Vg1 klasse i startgropen for kompetansebygging innen programmering. Vi har også en avgrensning med tanke på

muligheter innenfor Arduino, fordi grunnleggende programmering ikke nødvendigvis omhandler avanserte moduler som kan benyttes.

I den profesjonsfaglige delen har vi satt søkelyset på hvordan vi igjennom kompetanseheftet kan legge til rette for læring om Arduino. Samtidig har vi hatt fokus på relevant læringsteori med tanke på undervisningsopplegget.

1.4 Oppgavens struktur

I kapittel 2.0 finner man tilnærmingen vår til den yrkesfaglige fordypningsdelen. Problemstillingen presenteres i innledningen avsnitt 1.2, og gjennom aktuell teori og litteratursøk i kapittel 2.2. Kapittel 2.4 tar for seg drøfting og refleksjon rundt vår yrkesfaglige problemstilling. Her tar vi for oss hvordan vi kan presentere hva Arduino er, og hvorfor programmering har fått en sentral plass innen de teknologiske yrkesretningene EL og TIF.

Kapittel 3.0 tar for seg vår profesjonsfaglige del av bacheloroppgaven. Vi har i kapittel 3.1 belyst vår tilnærming til bakgrunn og tema. Videre i kapittel 3.2 presenterer vi vår problemstilling for den profesjonsfaglige delen, og i kapittel 3.3 presenteres pedagogisk og yrkesdidaktisk teori. Metode for datainnsamling blir presentert i kapittel 3.5 og presentasjon av funn fra aksjoneringen fremstilles i kapittel 3.9. Til slutt i del 3.0 finner vi kapittel 3.11 som tar for seg vår drøfting innen den profesjonsfaglige delen, og gjennom denne drøftingen ønsker vi å belyse noen av de resultatene som aksjoneringen vår, fordypningen og evalueringen viser. I kapittel 4.0 kommer vi med en avslutningsdel hvor vi oppsummerer, og kommer med en konklusjon på problemstillingen fra både yrkesfaglig fordypning-, og profesjonsfaglig del.

2.0 Yrkesfaglig fordypning

2.1 Bakgrunn

I en faglig utredning fra regjeringen (NOU 2020:2) setter de søkelyset på at det i fremtiden vil være et stort behov for kompetanse basert på samfunnets behov. Utvalget for denne faglige utredningen har hatt et treårig prosjekt pågående, hvor de har lagt hovedvekt på samfunnets, og da særlig arbeidslivets behov for kompetanse i fremtiden. Det er viktig for virksomhetene å ha god tilgang på riktig og nødvendig kompetanse, slik at de kan være mest mulig produktive og konkurransedyktige i næringslivet.

Ifølge Rossing og Stausland (Rossing & Stausland, 2021, s. 9) skal programmering og digital kompetanse inn i mange fag, helt fra grunnskolen til videregående skole. Grunnen til at programmering og digital kompetanse skal inn i mange fag etter hvert, henger sammen med flere påstander:

- Digitale hjelpemidler blir stadig mer og mer tatt i bruk i samfunnet, hvor de tar over funksjoner som mennesker før har utført. Vi vil være nødt til å kunne utvikle, tilpasse eller bruke digitale maskiner og hjelpemidler, noe som vil kreve at fremtidens arbeidere har en annen kompetanse enn dagens arbeidere.

- Programmering kan være med på å øke evnen til å strukturere tankeprosesser som er nyttig ved problemløsning.

- Programmering og mikrokontrollere er et godt designverktøy innen produktutvikling. Dette er noe som også omhandler digitale verktøymaskiner som f.eks. 3D-printere, CNC-freser, laserkuttere, symaskiner o.l. hvor alle disse styres av en programvare.

Hvis vi ser inn i krystallkulen og inn i framtiden vil vi leve i et demokratisk samfunn hvor digitalisering blir mer og mer utbredt. Det vil i stadig større grad anvendes mer teknologi og digitale metoder, som gjør at vi må være i stand til å ha meninger om viktige samfunnsspørsmål som involverer teknologiske metoder (Rossing & Stausland, 2021, s. 9).

2.1.1 Elektro- og datateknologi og teknologi- og industrifag (EL, TIF)

I løpet av årene som har gått, har læreplanene gjennomgått endringer i takt med samfunnets endringer og behov. Både innen EL og TIF har det i senere tid skjedd fremskritt med tanke på programmering.

Høsten 2020 kom fagfornyelsen for fullt inn i Norsk videregående skole og fagretningene EL og TIF ble berørt av den teknologiske utviklingen i samfunnet (Kunnskapsdepartementet, 2022). Endringene fra de gamle læreplanene og frem til fagfornyelsen bærer preg av at programfagene har blitt mer fremtidsrettede, og legger mer til rette for dybdelæring. Videre går det ut på at elevene skal forberedes på et arbeidsliv som stiller krav til både praktisk- og teknologisk kompetanse. Innen EL skal elevene være i stand til å møte framtidige kompetansebehov i et stadig mer teknologisk samfunn (Kunnskapsdepartementet, 2022). Det samme gjelder for TIF, og her legges det også større vekt på å ta i bruk digitale og robotiserte verktøy (Kunnskapsdepartementet, 2021).

Grunnen til at vi har valgt å belyse programmering gjennom Arduino er fordi videregående skoler benytter seg av Arduino. Det er utviklet en god del pedagogisk materiell til Arduino og det finnes mye tilleggsutstyr som er basert på Arduinoens plattform. Arduino er svært rimelig i innkjøp sammenlignet med andre programmeringsverktøy og den har både blokk og tekstbasert programmering. En annen fordel som gjør at skoler bruker Arduino er at det fungerer på både PC og MAC, og kan bygges på med flere moduler med web-brukergrensesnitt (Skaperskolen, u.å.).

2.1.2 Teknologi i samfunnet

I dagens samfunn kan vi finne teknologi i alle samfunnslag og micro-chiper finnes i svært mange produkter som vi bruker i hverdagen. PC, vaskemaskin, TV og mange flere redskaper vi bruker innehar en micro-chip eller -kontroller. I arbeidslivet har teknologien ofte fått som oppgave å forbedre et arbeide på forskjellige områder. En kan for eksempel sette opp en robot som ikke blir lei av en enkel operasjon som skal gjøres et uendelig antall ganger. På de fleste fagfelt er slike teknologiske løsninger viktig for å kunne ivareta arbeiderens helse og velvære, samtidig som at det over tid kan være billigere med en automatisert løsning. Konkurransen for bedrifter er globalt økende og gjennom internett gis det mulighet for

bestillinger fra produksjonsbedrifter over hele verden. Lavkostland kan utkonkurrere høykostland, vi er derfor avhengige av automatisering for å være konkurransedyktige (TRIPLE-S, 2020).

Å vurdere behovet for hjelpemidler innenfor velferdsteknologien ligger ofte i læreplanmålene til mange fag. Helse og oppvekstfag, teknologi- og industrifag og elektro- og datateknologi er noen av disse (Kunnskapsdepartementet, 2020). En kan tenke seg til at ved utførelse av et arbeid er jakten på bedre resultat, forbedret HMS, eller effektivisering en legal jakt. Allerede på barneskolen begynner de med enkel programmering, hvor elevene setter sammen forskjellige blokker som videre skaper et ønsket resultat. For eksempel kan vi ta frem en robot som plukker opp noe fra gulvet. Et slikt søkelys på velferdsteknologi i hverdagen og i skolen underbygger viktigheten for teknologisk læring.

2.1.3 Arduino i et historisk perspektiv

Vi skal tilbake til Ivera i Italia i 2005, hvor det det hele startet som et prosjekt. Målet var å lage en anordning for å kontrollere prototypene til interaksjonsdesignstudenter på en rimelig måte (Wikipedia, 2022).

Dette var et prosjekt som var bygget på Wiring-plattformen, som bygger på Processing og IDE-grensesnittet. Massimo Banzi, David Mellis (IDII-student) og David Curtielles startet et prosjekt hvor de kopierte Wiring koden, og startet et eget prosjekt som de kalte for Arduino. Arduino og Wiring hadde mye til felles av utviklingen som Nicholas Zambetti hadde gjort, og i en kort periode var også Zambetti ansett som et medlem av Arduino-teamet. Rundt samme tidsperiode ble også Gianluca Martino en del av Arduino-teamet, og hans oppgave var å hjelpe til med produksjonen og maskinvareutviklingen. Han utviklet sammen med David Cuartielles en billigere maskinvare, for å redusere kostnadene på brettene deres, ved å bruke Atmega8 (Barragàn, 2016).

2.1.4 Programmering i et historisk perspektiv

Tidlige datamaskiner

Programmering har eksistert lenge før de elektroniske datamaskinene kom og flere tenker nok at programmering er noe nytt som har kommet i senere tid. I 1843 laget Ada Lovelace

det første programmet på analysemaskinen til matematikeren Charles Babbage og fikk i etterkant et eget programmeringsspråk oppkalt etter seg. Nesten 100 år senere i 1936, lanserte Alan Turing en automatisk maskin som senere ble kalt Turingmaskin. Dette var en maskin hvor hensikten var å utføre enkle operasjoner etter bestemte instrukser. Alan Turing's beskrivelser skulle bli viktige for dagens datamaskiner.

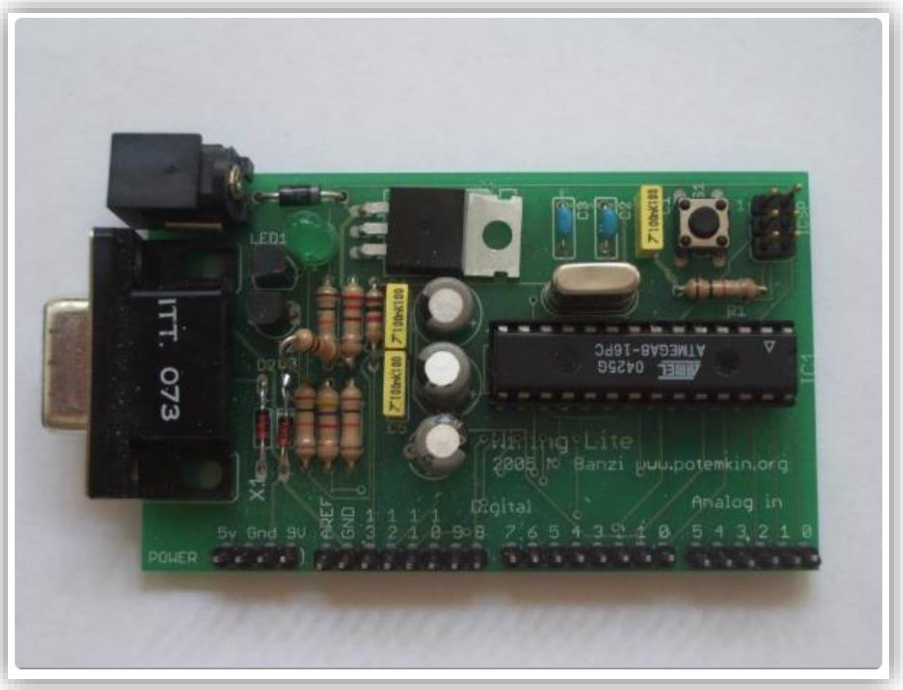
Elektroniske datamaskiner

Den første programmerbare elektroniske datamaskinen het ENIAC og ble først i 1946 satt i drift. Det var den amerikanske hæren som utviklet ENIAC for å kalkulere utskytingen av artillerier. Programmererne av ENIAC var seks kvinner og gjennom sitt arbeid ble de viktige for utviklingen av programmering (Lindsø, 2020).

På femtitallet kom de første programmeringsspråkene og de hadde navnene COBOLT, FORTRAN og Autocode. Disse ble viktige for å legge grunnlaget for programmeringsspråkene vi har i dag. Fra åtti- og nittitallet frem til i dag har de fleste programmeringsspråk vi kjenner til blitt utviklet. I 1980 kom C++ som er videre utviklet av det gamle programmeringsspråket C. Dette er programmeringsspråk som har hatt stor innflytelse på andre språk innenfor programmering (Lindsø, 2020).

Nittitallet var det kommersialisering av internett og i denne sammenhengen kom det flere programmeringsspråk. JavaScript og PHP er eksempler på programmeringsspråk som er tilknyttet internett. Det er også andre kjente programmeringsspråk som kom på nittitallet, for eksempel Python, Visual Basic og Java. Dette var programmeringsspråk som var objektorienterte (Lindsø, 2020).

2.2 Teori



Figur 1 Den første Arduino prototypen - Wiring Lite

2.2.1 Hva er programmering, og hvorfor programmere?

Om vi enten skriver, designer, feilsøker, tester eller vedlikeholder koden til et program som skal tolkes av en datamaskin, programmerer vi. En person som arbeider med programmering blir omtalt som en programmerer eller utvikler, og det er denne personen som skriver koder etter en spesiell syntaks. Denne syntaksen inneholder instruksjoner for hvordan programmet skal oppføre seg (Wikipedia, 2019).

Programmering er en metode for å fortelle hva en datamaskin skal utføre. For eksempel hva skal skje dersom en knapp blir trykt ned, eller fortelle en robot at han skal gå 10 meter. Når vi arbeider med programmering gir vi programmet til en datamaskin, robot eller mikrokontroller, for å fortelle hva den skal gjøre. Vi kan sammenligne et program med en oppskrift. I og med at samfunnet stadig utvikler ny teknologi, vil det være viktig at vi i tiden fremover er i stand til å kunne programmere (Learnlink, 2020).

Grunnen til at vi holder på med programmering er bundet til flere årsaker. En av årsakene er at datamaskiner utfører en kompleks og repeterende oppgave mer effektivt enn et menneske, og i tillegg har datamaskiner kapasiteten til å arbeide hele døgnet. Programmering kan brukes til å effektivisere arbeidslivet og gjøre arbeidsoppgaver mindre belastende for mennesker. Smarthus har kommet mer og mer inn i samfunnet den siste tiden, og det er med på å gjøre livet til mennesker bedre. Smarthus kan programmeres og gjøre dagen vår mer praktisk, og være en løsning for hjelpemiddel til mennesker med funksjonsnedsettelser. Ved å programmere smarthusløsninger kan vi også bidra til en grønnere hverdag og bli mer miljøvennlige (Lindsø, 2020).

2.2.2 Programmeringsspråk

Arduino er plattformen som denne oppgaven baseres rundt, men en skal vite at det finnes andre programmeringsspråk som også er vanlige å bruke innen nåtidens teknologi. Læring innen programmering for barn baseres ofte på blokkprogrammering. Blokkprogrammering er «blokker» som gjør bestemte funksjoner som videre settes inn i et system (Haraldsrud et al., 2020, s. 15). Allerede fra barneskolen begynner undervisning rundt slike systemer, hvor programmeringskonseptene og algoritmisk tenkning er i hovedsetet (Haraldsrud et al., 2020, s. 87). Dette er et enkelt system og gir god læring for grunnleggende prinsipper innen programmering. Mer avanserte systemer finner vi når programmeringen rettes mot industrielle PLS`er og operasjonene blir mer avanserte. Arduino kan ansees som en fin start for VG1 elever, ikke for enkelt, men en grunnleggende læring innen programmering.

Arduino bruker programmeringsspråket C++. Det er et språk som ble utviklet av danske Bjarne Stroustrup først på 1980-tallet ved Bell Laboratories i USA. Første versjon av programmeringsspråket ble lansert i 1985, og ble senere standardisert av ISO og ANSI på slutten av 1990-tallet. C++ er et forbedret programmeringsspråk videreutviklet fra språk C. Det ble utviklet for å skape et språk som hadde funksjonaliteten for høynivåspråk og hastigheten til lavnivåspråk (Almås & Rossen, 2021). Høynivåspråk er et programmeringsspråk for datamaskiner der nivået er mellom assemblerspråk og fjerdegenerasjons verktøy. Python, Java og C er eksempler på slike språk. Programmer utformet i høynivåspråk må gjøres om til en maskinkode av et eget program for å kunne brukes. Ved å oversette hele programmet, kalles programmet som oversetter for kompilator, og det blir da et kompilert språk. Eksempler på slike kompilerte høynivåspråk er

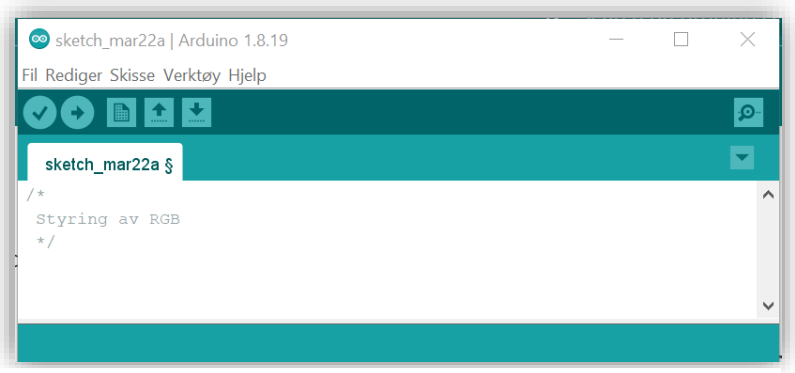
C, Cobol og Fortran. Omtaler vi et tolket språk, så oversetter programmet en linje av gangen. Java er det største tolkete språket.

I et høynivåspråk som beskrevet ovenfor kan en instruksjon respondere til lange sekvenser av maskininstruksjoner. Derimot ett lavnivåspråk er et programmeringsspråk som er skrevet slik at hver instruksjon svarer til en instruksjon i maskinkoden. En annen forskjell er også at lavnivåspråk er til spesifikke prosessorfamilier eller datamaskiner, mens høynivåspråk kan kompileres til flere maskintyper (Rossen, 2020).

2.2.3 Navnekonvensjon

Når vi skal skrive koder er det viktig at de er tydelige og lette og lese. Når koden blir skrevet så er det ikke sikkert det er tenkt å lese den flere ganger, men det kan for eksempel oppstå endringer i hvordan kunden vil ha et produkt til å fungere. Da må det være lett å sette seg inn i koden for å kunne gjøre ønskete endringer, slik at man slipper å skrive et helt nytt program. Vi skal vise to eksempler på hvordan en kode kan skrives.

camelCase er en metode, og da deler vi ord med stor bokstav. Da skriver vi for eksempel int minVariabel. En annen måte å dele ord på er med bruk av understrek. Da vil samme eksempelet stå som int min_variabel. Det er viktig å følge den metoden man velger gjennom hele programmet.



Figur 2 viser et eksempel på forklaring av programmet i starten av programmet.

Det er også god programmeringsskikk å forklare i begynnelsen av koden hva programmet går ut på. Det skriver vi mellom «/*» og «/*» (Venås & Vangsnes, 2020).

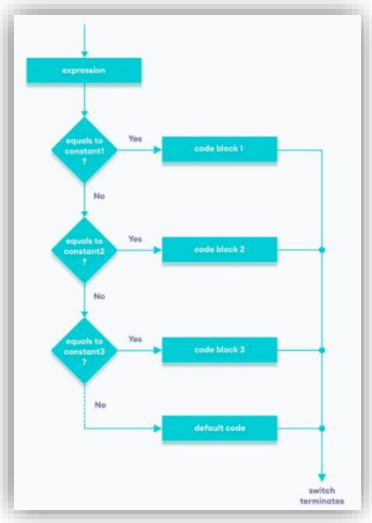
2.2.4 Struktur

Hvordan koden er skrevet, rekkefølgen og hvilke funksjoner som er benyttet, er strukturen i programmeringsspråket. Etter det er forklart hva som skal skje i programmet, deklareres globale verdier og konstanter. Deretter kommer de to hovedfunksjonene «void setup» og «void loop». Funksjonen setup kjøres bare en gang ved oppstart eller resetting av arduino-kortet. Den brukes til å inkludere biblioteker, initialisere variabler og pin-moduser (Arduino.cc, 2019). Loop funksjonen er som det ligger i ordet, den går om og om igjen slik at programmet kan oppdage endringer og reagere og styrer arduino-brettet (Arduino.cc, 2019).

Figur 3 Eksempel på kode.

En setningsblokk eller en kodeblokk, består av en eller flere programsetninger gruppert sammen. Kompilatoren betrakter dem som sammenlenket, og de må starte og slutte med krøllparentes: { }.

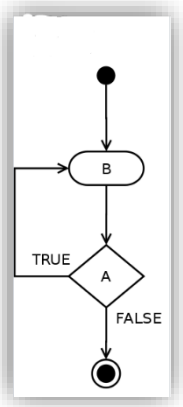
Videre deles det inn i flere kontrollstrukturer. Break brukes for å omgå den normale sløyfetilstanden ved å gå ut av en for, while eller do-while sløyfe. Brukes også for å gå ut av en switchcase (Arduino.cc, 2019)



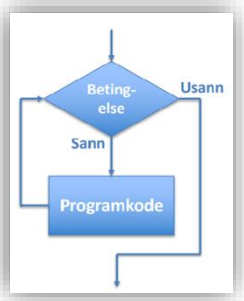
Figur 4 Eksempel på switch-case.

En switch case kontrollerer flyten i programmene ved at forskjellige koder skal benyttes under ulike forhold (Arduino.cc, 2019). If-else som blir nevnt senere kan også brukes til å ha mange alternativer, men blir fort uoversiktlig. Da er switch-case'n mer praktisk å bruke, den kan utføre mange ulike instruksjoner avhengig av hvilken verdi variabelen har (Arduino.cc, 2019).

En while-løkke vil repetere seg så lenge uttrykket i parentesen er sann (Arduino.cc, 2020). While løkken kan for eksempel brukes til å kjøre noe så lenge en bryter er betjent.



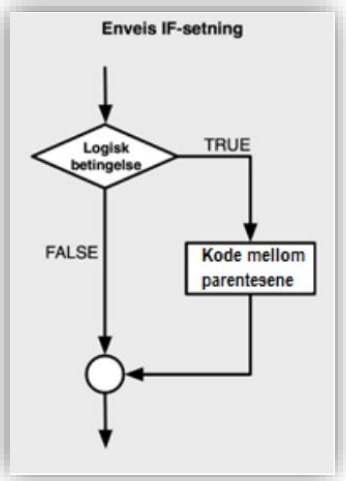
Figur 6 Eksempel på en do-while-løkke.



Figur 5 Eksempel på en while-løkke.

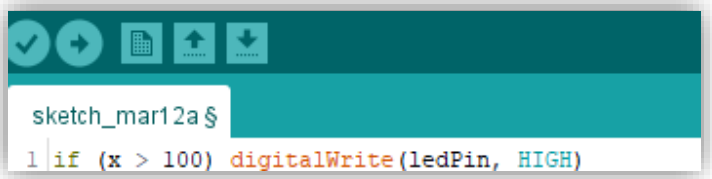
Do-while løkken virker likt som while-løkken, men i tillegg testes tilstanden i slutten av løkken som gjør at do-løkken kjører minst en gang (Arduino.cc, 2021).

If ser etter om en tilstand er sann, og kjører en funksjon eller flere når den er sann (Arduino.cc, 2020). Det vil si at den som programmerer bestemmer hva som skal skje hvis en bestemt betingelse er oppfylt. Ett enkelt eksempel kan være at når en verdi er over 100, skal ledPin være HIGH. Er verdien over 100 så er betingelsen true, og koden som gir ledPin HIGH, kjøres. Programmet kjøres så videre. Er betingelsen false, går programmet bare videre forbi den delen av programkoden, og ingenting skjer med ledPin, den forblir LOW.



Figur 7 Eksempel på enveis If-setning.

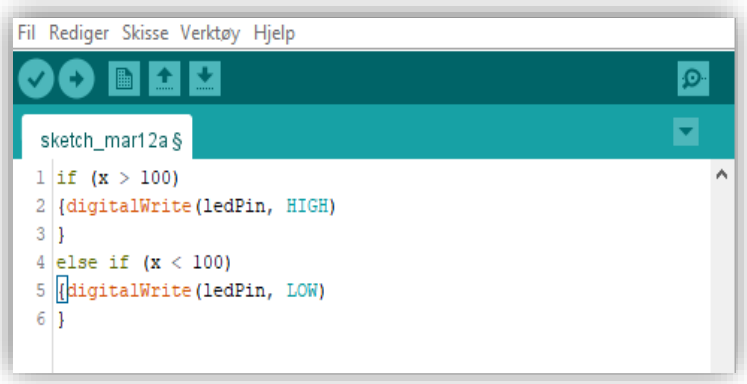
Eksempler på hva som ofte sammenlignes ved hjelp av if-setninger:



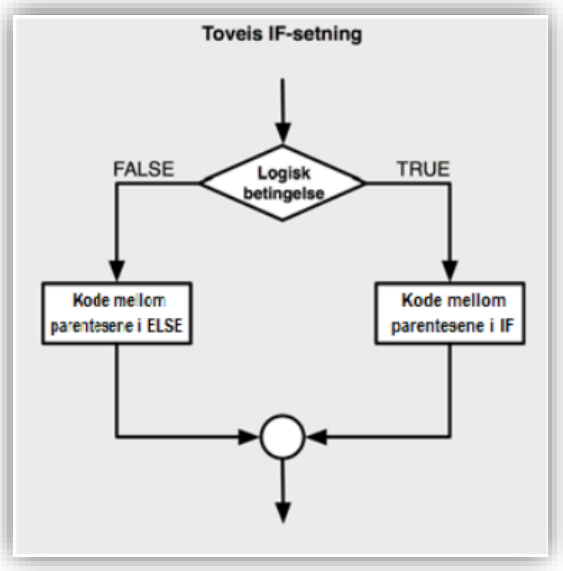
Figur 8 Eksempel på en If-setning i en sketch.

- Om en int-variabel er høyere enn en bestemt verdi.
- Om en boolean-variabel er sann eller usann.
- Om en char-variabel er et bestemt tegn.

If-else setningen er slik at uansett om betingelsen i koden er sann eller usann, vil en programkode bli utført. If-else kan også brukes på eksempelet for if-setningen. Hvis verdien er over 100, kjører programkoden som gjør at ledPin blir HIGH, og er den under 100, kjører programkoden som gir at ledPin er LOW.



Figur 9 Eksempel på en If-else setning i en sketch.



Figur 10 Eksempel på en toveis If-setning.

2.2.5 Funksjoner

En kan si at for å oppnå struktur i kodesegmentene er en helt avhengig av funksjoner, strukturen bestemmer betingelsene for funksjonen. Funksjonene brukes ofte ved gjentakende handlinger i et program, kalt for rutine, metode eller prosedyre (Rossing & Stausland, 2021, s. 35).

Analoge Funksjoner på Arduino brettet finnes på pinne A0-A7

analogRead () – Oppgir verdi på en pin ved for eksempel temp sensor, 0-5 (3.3)V, og blir oppgitt i verdier fra 0-1024 enheter, 1 enhet er lik 0.0049V.
analogWrite () – Kan brukes for å drive elektromotorer, sette lys på led pærer eller andre forbrukere i oppsettet. I analogWrite () kan verdiene justeres og turtall eller lysstyrke vil variere etter ønsket programmering.
Funksjoner for tid, disse settes inn i programkoden for å velge en varighet for en funksjon.
delay () – setter programmet som kjøres på pause i en angitt tid, oppgis i millisekunder. En kan for eksempel bruke denne ved en blinkende LED. I det den ikke lyser kan delay () brukes.
delayMicroseconds () – brukes på samme måte som delay (), men tidsenheten er

microsekund.

Micros () og millis () tar tid fra programmet starter og gå tilbake til null etter en stund.

Digitale funksjoner I/O på Arduino brettet fines på pin D0-D13 (PWM 3,5, 9, 10 og 11)

digitalRead () -Disse er digitale funksjoner som er enten på eller av I/O eller (HIGH) og (LOW).

digitalWrite () – Input eller Output, må velges som LOW eller HIGH i pinMode ().

pinMode () -- presenterer hvilket nummer ut/inngangen finnes på.

```
12 void setup(){
13  pinMode(whiteLed, OUTPUT) ; // setter whiteLed som en OUTPUT Frontlys
14
15 }
16 void loop (){
17
18
19 digitalWrite(whiteLed, HIGH);   // Får Fremlysene til å lyse)
```

Figur 11 Eksempel på HIGH eller LOW.

En illustrasjon av hvordan (HIGH) eller (LOW) funksjonen kan se slik ut:

I neste eksempel er det satt betingelser for funksjonene. Disse er if og else mens funksjonene er digitalWrite() og pinMode(). Innenfor parentesene settes pin-nummer og verdi av funksjonen. Andre funksjoner som kan brukes er mattefunksjoner, å gjøre arduino om til en regnemaskin for vanlige formler eller til trigonometri.

```
1  int switchState = 0;
2
3  const int greenLed = 3;
4  const int redLed = 4;
5  const int blueLed = 5;
6  const int buttonPin = 2;
7
8  void setup() {
9    pinMode(greenLed, OUTPUT);
10   pinMode(redLed, OUTPUT);
11   pinMode(blueLed, OUTPUT);
12   pinMode(buttonPin, INPUT);
13 }
14
15 void loop() {
16   switchState = digitalRead(buttonPin);
17
18   if (switchState == LOW) {
19     digitalWrite(greenLed, HIGH);
20     digitalWrite(redLed, LOW);
21     digitalWrite(blueLed, LOW);
22   }
23
24   else {
25     digitalWrite(greenLed LOW);
26     digitalWrite(redLed, LOW);
27     digitalWrite(blueLed, HIGH);
28   }
```

Figur 12 Eksempel på en programkode hvor det er satt betingelser.

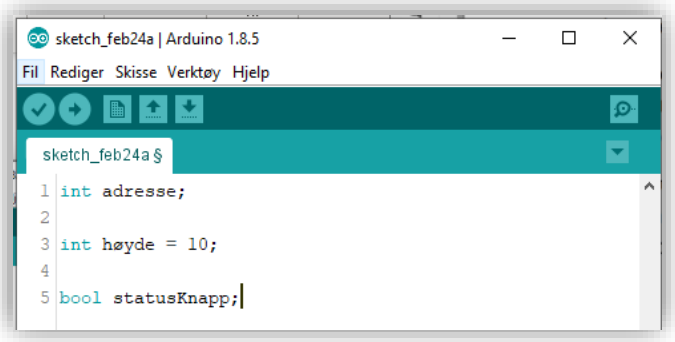
2.2.6 Variabler

Hvorfor benytter vi variabler? Jo, av to ting:

- Tilordning: vi gir den en verdi
- Leser verdien

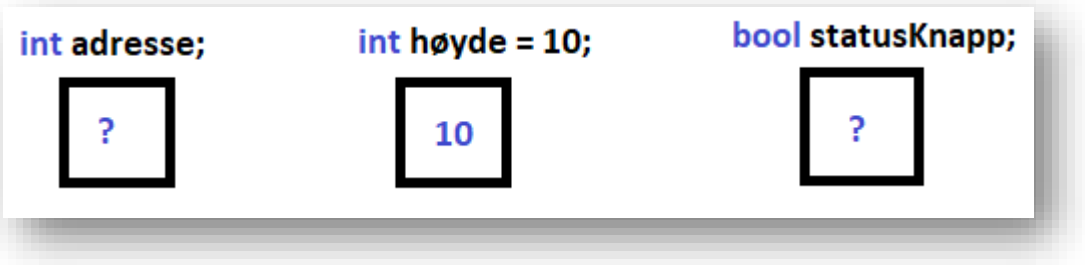
Når vi gir noe en verdi, vil det skje ting flere ganger, det vil si at den kan få verdien endret flere ganger i løpet av programkjøringen. Vi sier at en variabel er en plassholder i datamaskinen for informasjon som vi ønsker å ta vare på, eller endre i løpet av programkjøringen (Arduino.cc, 2022).

I bilde til høyre ser vi et eksempel på tre variabler, hvor de deklareres ved at vi gir dem både navn og knytter dem til en datatype. Den første variabelen er adresse, som har ingen verdi enda, men vi kan fortelle mikrokontrolleren at det må settes av plass til den. Denne variabelen skal være av datatypen int.



Figur 13 Eksempel på 3 forskjellige variabler.

Variabelen høyde har vi i dette eksempelet gitt verdien 10. Dette er en verdi som blir lagret, og det vil gjøre det mulig for oss å hente den ut senere i programmet ved å referere til variabelnavnet temperatur. Her er også int datatypen til variabelen. Det siste eksempelet i bildet er variabelen statusKnapp, og er av datatypen bool. Vi kan se at også denne variabelen er uten verdi. I illustrasjonen under ser vi de tre variablene som er deklarert i eksempelet over med sine plassholdere i datamaskinen.



Figur 14 Illustrasjon av tre variabler.

I enkelte programmer får vi bruk for variabler som ikke skal endres senere i koden og disse kalles for konstanter. Dette kan være en verdi som ikke vil bli forandret og det kan være en verdi som ikke blir forandret selv om programmet kjøres. For å illustrere dette kan vi se for oss et eksempel hvor vi skal lage et program som sjekker hvilke rettigheter vi har, basert på alderen vår (Lindsø & Bårdgård, 2019).

Først i programmet blir vi spurt om hvor gamle vi er, og svaret blir lagret i en vanlig variabel:

```
int personAlder;
```

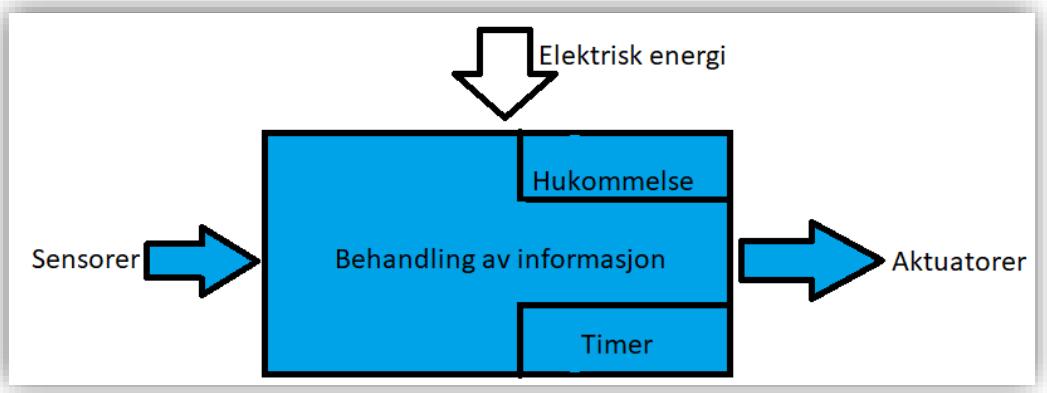
Deretter vil programmet sammenligne verdien av denne variabelen opp mot en konstant som vi har laget. Denne konstanten skal sjekke om brukeren er 18 år eller eldre:

```
const myndigAlder = 18;
```

Videre testes alderen som brukeren legger inn opp mot konstanten myndighetsalder. Da kan programmet lage en liste over hvilke rettigheter en bruker har. Fordelen med å skrive inn 18 som en konstant er fordi hvis dette tallet skal endre en gang, trenger vi bare å endre konstantens verdi. Hadde vi skrevet verdien gjennom hele programmet har vi måttet endre verdien flere plasser. Ved å løse det ved å benytte seg av konstanter fjerner vi også mulige feilkilder hvis en tallverdi blir skrevet feil i løpet av koden (Lindsø & Bårdgård, 2019).

2.2.7 Rammeverk

Elektronisk utstyr, uansett hva det er kan tilpasses beskrivelsen i figuren under:

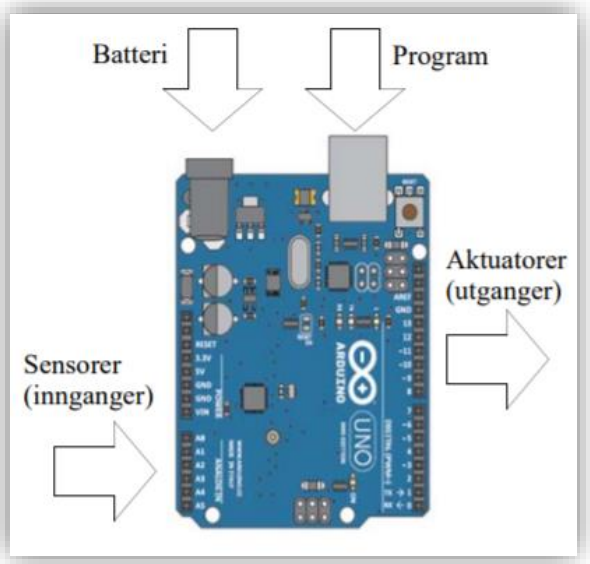


Figur 15 Behandling av informasjon av elektronisk utstyr.

Sensorer er en elektronisk enhet som innhenter informasjon fra omgivelsene rundt. Eksempler på slike sensorer kan være temperatur-, fuktighet-, lys-, bevegelses- , røyksensorer og flere. Det finnes uendelig mange forskjellige typer sensorer og dens bruksområder. Vi kan for eksempel se for oss en bryter som vi bruker til å slå på lyset med, eller en mikrofon som registrerer lyder.

Aktuatoren er en elektronisk enhet som vi bruker til å utføre en oppgave. Det kan være en motor som åpner en port, et alarmhorn som gir en lyd, en lampe som lyser, eller et varmeelement som gir varme. Aktuatoren har som oppgave å utføre en aksjon (Rossing & Stausland, 2021, s. 19).

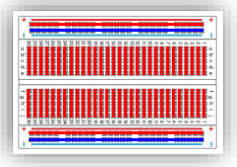
Databehandlingsenheten er en elektronisk enhet som kan være svært enkel eller veldig avansert. Denne elektroniske enheten samler inn informasjon fra omgivelsene av sensorer. Deretter gis aktuatorene beskjed om handling. Sensorene tilkobles innganger, og aktuatorer tilkobles utganger. Vi tilfører også enheten elektrisk energi fra et batteri eller strømnettet. Enheten kan også i noen tilfeller inneholde en timer som passer på tiden. Er arbeidsoppgavene til enheten sammensatt, må den også inneholde en oppskrift, et program, som beskriver hvordan informasjonen fra sensoren skal behandles (Rossing & Stausland, 2021, s. 18-19).



Figur 16 Databehandlingsenhet.

2.2.8 Komponenter

Koblingsbrett og Arduino Uno



Figur 19 Koblingsbrett

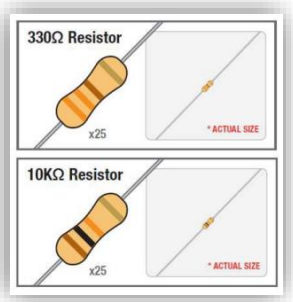
Figur 18 Jumper wire

Figur 17 Arduino UNO kort

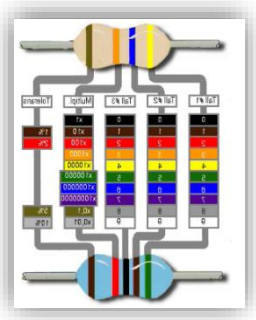
Bilde av koblingsbrettet viser hvilke koblingspunkter som er koblet sammen på undersiden. Vi bruker jumper wire 'e for å forbinde komponentene på koblingsbrettet med Arduino-en. På sidene av koblingsbrettet har vi to langsgående sammenhengende hullrader. Disse er ment for å føre + og -. Den røde langsgående linjen er for pluss, og den blå langsgående linjen er for minus. Dette er for å indikere at alle kontaktpunktene langs en linje er koblet elektrisk sammen (Rossing & Stausland, 2021, s. 22).

Motstander

Motstander finnes i mange ulike verdier, og det er fargekoden på motstanden som bestemmer hvilken verdi motstanden har. Det er viktig å velge riktig verdi ut ifra hva du skal bruke det til.



Figur 20 Motstand



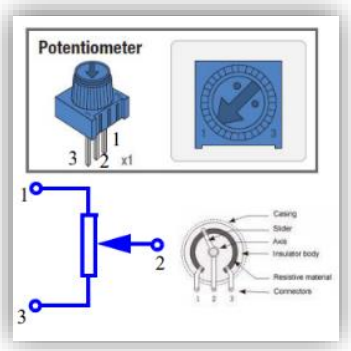
Figur 21 Motstandsverdier

Grunnen til at motstander brukes er at en i enkelte tilfeller ønsker å begrense strømmen i en komponent, eller for å etablere et spenningsnivå, som oppnås med en spenningsdeler. Strømmen i en motstand vil ifølge ohms lov øke proporsjonalt med spenningen over den, fordi det for eksempel er ønskelig å omdanne en varierende mostandsverdi hos mange sensorer for å få en varierende spenning. Hvilken vei motstanden plasseres har ingen betydning for dens virkemåte.

Det er fargene på ringene til mostanden som bestemmer verdien den har, og hver farge står for et av sifrene 0 til 9 (Rossing & Stausland, 2021, s. 23).

Potensiometer

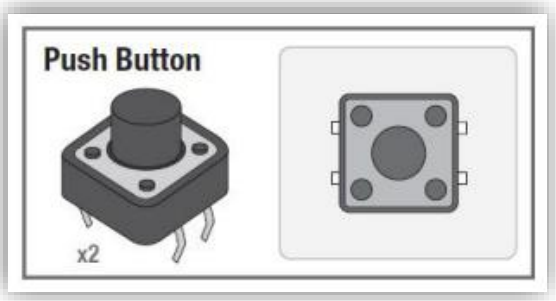
Et potensiometer er en motstand, men i motsetning til beskrivelsen av motstander i avsnittet ovenfor, er dette en motstand med et variabelt uttak. Potensiometer benyttes der hvor vi ønsker å etablere en variabel spenning mellom spenningen på ytterpunktene, eller bare ønsker at den skal fungere som en volumkontroll som f.eks. for et signal som



Figur 22 Potensiometer.

sendes inn mellom pinne 1 og 3, og som tas ut mellom 2 og 3 (Rossing & Stausland, 2021, s. 23-24).

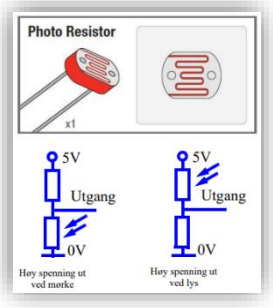
Bryter



Figur 23 Bryter.

Bryteren som følger med arduino-en har fire bein, som er forbundet med hverandre to og to. Trykker man på knappen vil de to parene bli koblet sammen, og så lenge knappen er trykket inn opprettholdes forbindelsen helt til den slippes og brytes. Vi bruker bryterne til å gi kretsen en enkel informasjon om av og på, akkurat som en lysbryter. Trykket på bryteren må omdannes til en spenning for at kretsen skal forstå informasjonen (Rossing & Stausland, 2021, s. 24).

Fotomotstand

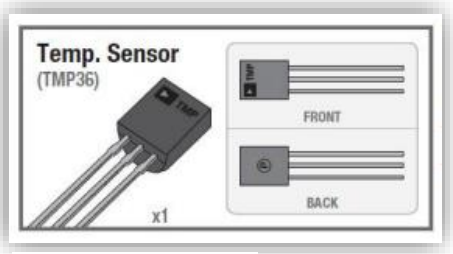


Figur 24 Fotomotstand.

Fotomotstanden er rett og slett en motstand der hvor verdien blir bestemt av intensiteten på lyset som treffer den. Vi vil oppleve å få lavere motstandsverdi, desto mer den belyses. Sterkt lys vil normalt gi 100 ohm, mens mørkt lys gir typisk 300 Kohm. Fotomotstanden består av to bein, og det har ingen betydning hvilken vei den kobles inn i kretsen.

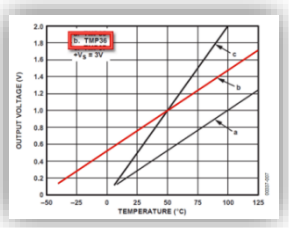
Ønsker man å styre en funksjon ved hjelp av lysstyrken, benyttes gjerne en fotomotstand. Det kan f.eks. være tenning av lys når det blir mørkt ute. Fotomotstanden kobles som oftest i en spenningsdeler, og plasseringen bestemmes av funksjonen (Rossing & Stausland, 2021, s. 25).

Temperatursensor



Figur 25 Temp. sensor.

Temperatursensoren (TMP36) registrerer temperatur og gir ut en spenning, som er proporsjonal med temperaturen. Denne typen sensorer brukes gjerne i elektroniske termometre eller i termostater for å regulere en varmeovn. Vi kan også bruke temperatursensor for å beskytte elektronikk, og det gjøres ved at strømmen brytes når temperaturen overskrider et maksimalt nivå. Spenningen øker med 10 mV per grad. Ved 0 ℃ vil spenningen være 0,5 V.



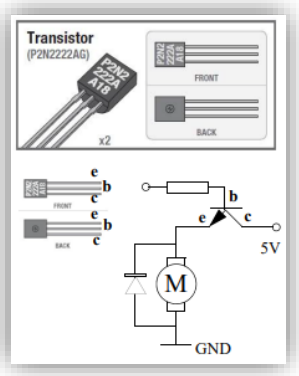
Figur 26 Diagram av spenning som funksjon.

I bilde til høyre ser vi et diagram som viser spenningen som funksjon av temperaturen. Temperatursensoren (TMP36) er den røde kurven (Rossing & Stausland, 2021, s. 25).

Transistor

En transistor har som regel to bruksområder:

- Forsterker når strømmen/basespenningen varierer innen et lite område
- Bryter når strømmen/basespenningen er så stor at den slår av eller på transistoren

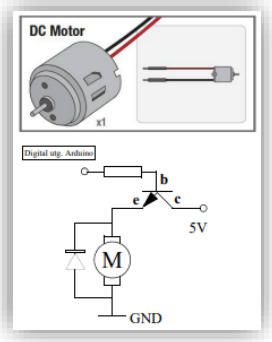


Figur 27 Transistor.

I elektroniske forsterkere, radiosendere og mottakere, TV-er o.l. brukes transistoren som en signalforsterker. I styringssystemer og datamaskiner brukes den oftest som bryter. På bilde til høyre ser vi at en transistor har tre bein, eller terminaler som det heter. Bein «c» kalles for collector, bein «e» kalles for emitter og bein «b» for basen. Det kan gå en relativ stor strøm fra collector beinet til emitter beinet, og størrelsen på denne collectorstrømmen kan styres av spenningen imellom basen og emitter, og når den blir stor nok kan det gå strøm i transistoren (Rossing & Stausland, 2021, s. 27).

Motor

Når en motor får tilført en spenning på over 3 V vil akslingen begynne å rotere. Arduinoer klarer ikke å styre store strømmer helt opp til 200 mA, derfor kan vi benytte en transistorbyter som vil tåle den store strømmen.



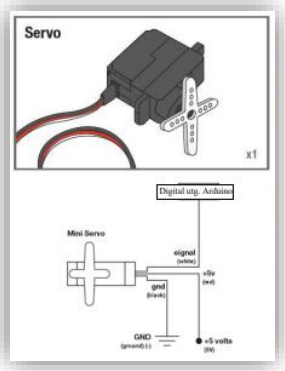
Figur 28 Motor.

I dag er bruksområdet for motorer veldig kjent. Vi finner de igjen hvor det er noe som enten skal gå rundt eller bevege seg. Det kan være alt fra leketøy, datadisker til vifter, pumper, elektriske biler etc.

I bilde til høyre ser vi en krets hvor det er koblet inn en «fly back» diode over motoren. Den har som oppgave å hindre overspenninger som kan oppstå på transistoren idet strømmen brytes for å stoppe motoren (Rossing & Stausland, 2021, s. 27).

Servo

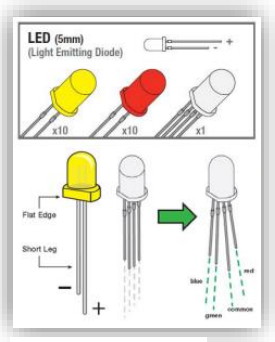
En annen motor vi blir kjent med igjennom arduino er servo motoren. Den kan dreie akslingen en bestemt vinkel eller rotere med en definert fart. På kommando fra mikrokontrolleren kan servo motoren dreie akslingen i en vinkel fra 0 – 180 ^{0}. Servoen mottar pulser, og det er lengden på en puls som bestemmer dreievinkelen. Har vi en pulslengde på 1,5 ms gir det oss en vinkel på 90 ^{0}. Disse pulsene må gjentas med jevne mellomrom, akkurat som man pulsbreddemodulerer et signal. Vi kan koble servo motoren til en utgang som har en slik utgang (pmw), f.eks. analog port nummer 9.



Figur 29 Servo.

Servo motorene finner vi som oftest i modellfly, hvor det ønskes å styre side-, høyderor og flaps. Dette er også en komponent som er viktig i mange robot styringer, hvor målet kan være å bevege for eksempel en arm (Rossing & Stausland, 2021, s. 28).

Lysdioder



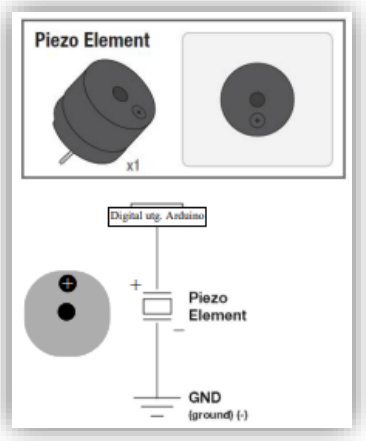
Figur 30 Lys dioder.

Lysdioder leder strøm bare en vei, og det er fra anoden til katoden. Det vil si, for at den skal lyse er vi avhengige av å koble den riktig vei. Lysdioden begynner å lyse svakt når strømmen i den overstiger ca. 1,5 – 2 mA. Ettersom strømmen øker, øker også lysstyrken på dioden. Lysdioden kobles gjerne i serie med en motstand som har en verdi på 220 – 330 ohm, dette for å begrense strømmen, fordi uten denne seriemotstanden er det stor sannsynlighet for at dioden går i stykker.

Det finnes også lysdioder som har navnet RGB, og består av en rød-, en grønn og en blå lysdiode som er montert i samme kapsel.

Lysdioder eller LED, som står for light emitting diode, brukes til blant annet signallamper, displayer og i den senere tid til belysning (Rossing & Stausland, 2021, s. 28).

Buzzer



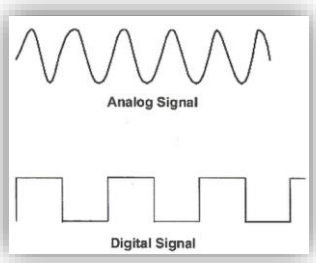
Figur 31 Piezo buzzer.

En Buzzer, eller et piezo-elektrisk element, består av et krystall som trekker seg sammen når det påføres en spenning og et klikk kan høres. Når denne spenningen forsvinner, vil krystallet i buzzeren få tilbake sin opprinnelige form og et klikk vil høres. Ved å la spenningen variere hurtig høres lyd som i en høyttaler. For at buzzeren skal fungere korrekt må pluss og minus være riktig tilkoplet.

For å lage lyder og toner med forskjellig frekvens bruker man et piezo-elektrisk element. Buzzeren trenger bare en spenning for å gi en tone, på grunn av at buzzeren består av et piezo-element og en enhet som lager en varierende spenning (Rossing & Stausland, 2021, s. 29).

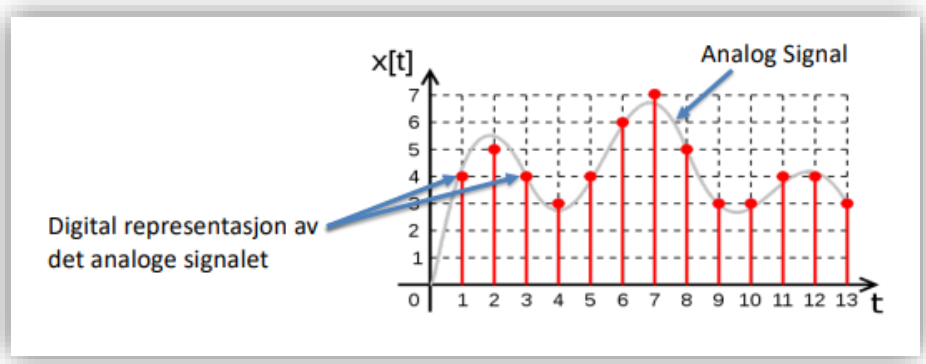
2.2.9 Analoge og digitale signaler

Når det skal fraktes informasjon innen arduino, finnes det to typer signaler. Analoge og digitale. Analoge signaler er et kontinuerlig elektrisk signal, mens digitale er et ikke-kontinuerlig elektrisk signal.



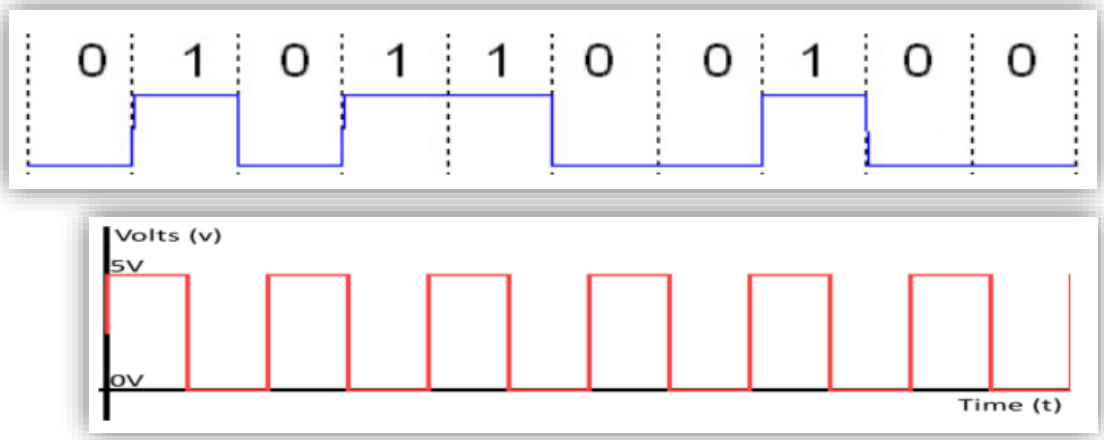
Figur 32 Analogt signal

Et analogt signal er et signal for strøm eller spenning. Styrken på signalet representerer andre fysiske størrelser som blant annet lydtrykk, lysstyrke osv. Det kan brukes et digitalt signal som alternativt signalformat, hvor det kan presentere de fysiske parameterne ved en rekke tall som angir signalstyrken på gitte tidspunkt (Halvorsen, 2018).



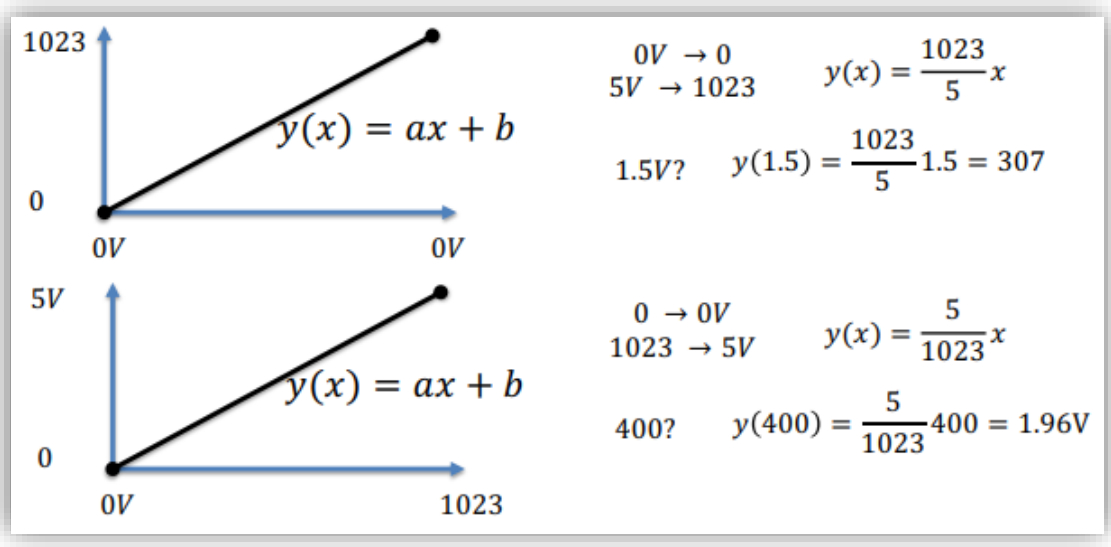
Figur 33 Digital presentasjon av analogt signal

Et digitalt signal har kun to forskjellige verdier. «0» og «1». Innen arduino bruker vi 5V som «1» (true), og 0V som «0» (false).



Figur 34 Forskjellige verdier på et digitalt signal.

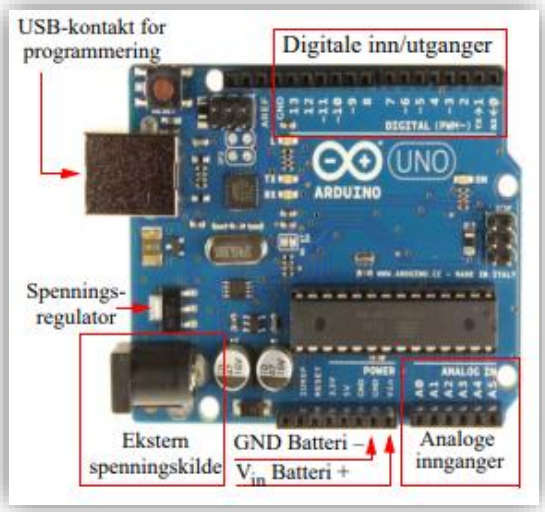
Det er mulig å konvertere analoge signal til digitale signal. Arduinoen bruker 0-5V som analogt signal. For å konvertere dette analoge signalet til et digitalt signal, konverterer arduinoen det analoge signalet til en verdi mellom 0 og 1023, ved hjelp av en 10 bits, altså 2^{10} = 1024 (Halvorsen, 2018).



Figur 35 Konvertering av analogt signal til digitalt signal.

2.2.10 Mikrokontrollerkortet – Arduino UNO

Mikrokontrollerkortet kalles for databehandlingsenheten, og her finnes inn- og utganger for sensorer og aktuatorer. I tillegg til inn- og utganger har den som nevnt i avsnitt 2.2.7 rammeverk, en intern timer og et internt lager for program og data. Kortet har også en USB-inngang, hvor det er mulig for brukeren å legge inn programvare og en plugg for batteritilkobling eller en batteriadapter.



Figur 36 Mikrokontrollerkort.

Flere og flere produkter benytter i større grad mikrokontrolleren i dagens samfunn, og den har et av de mest populære mikrokontrollerkortene innen arduino familien, UNO R3. Dette er et kort som er bygget opp på prinsipper fra Atmel mikrokontrolleren Atmega328P. Den

har en klokkefrekvens på 16 MHz, og et flash lager i størrelsen 32 kbyte, SRAM 2 kbyte og en EEPROM 1 kbyte (Rossing & Stausland, 2021, s. 29).

Mikrokontrollerkortet består av følgende inn- og utganger:

- **Digitale I/O-porter**

  Det er 14 digitale inn- og ut porter på mikrokontrolleren. Dette er porter som kan programmeres til å enten være en inn- eller utgang. Port 3, 5, 6, 9, 10 og 11 er porter som kan pulsbreddemoduleres, såkalt pwm signal. Disse portene er merket med et sinus symbol (~) på kortet. Maksimal strøm er 40 mA på hver av I/O portene (Rossing & Stausland, 2021, s. 30).

- **Analoge innganger**

  Mikrokontrolleren har 6 analoge innganger på kortet, som kan tilføres en spenning fra 0 – 5V. Tilføres det spenninger høyere enn 5V vil det ødelegge mikrokontrolleren (Rossing & Stausland, 2021, s. 30).

- **USB-kontakt**

  Mikrokontrollerkortet er også utstyrt med en USB-kontakt, denne brukes for direkte tilkobling av PC og for programmering av kortet og overføringer. Det kan også overføre data mellom PC-ens monitor og kortet. Det tilføres en spenning fra USB-kontakten under programmeringen, og hvis denne belastes med mer enn 500 mA vil strømforsyningen brytes inntil strømtrekket er redusert til under denne grensen. Bruker man en USB3 vil denne kunne levere en vesentlig høyere strøm (Rossing & Stausland, 2021, s. 30).

- **Strømtilførsel**

  Anbefalt spenning for batterieliminator er på 7 – 12V, og grenseverdiene er mellom 6 – 20V. Man kan enten tilkoble batteriet via eliminatorpluggene, eller via V_{in} og GND. Datainnsamlingskortet får spenningen sin fra 5V utgangen, og enkelte komponenter må ha en lavere spenning som leveres fra 3.3V utgangen (Rossing & Stausland, 2021, s. 30).

- **Reset**

  Mikrokontrollerkortet inneholder en RESET-knapp, som kan brukes for å resette programmet slik at det starter på nytt. Det er også montert kantkontakter som er ment for å montere til tilleggskort, shield card, rett på arduino-kortet (Rossing & Stausland, 2021, s. 30).

- **Rx/Tx**

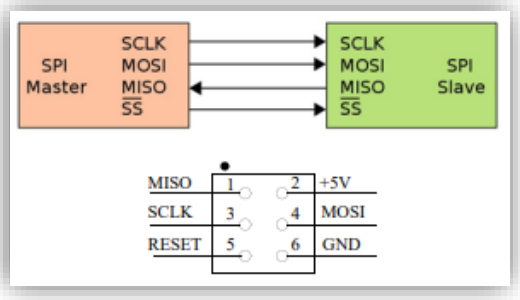
  Via portene Rx og Tx (I/O-port 0 og 1) støtter kortet UART, altså seriell datakommunikasjon. Disse portene benyttes for programmering av mikrokontrolleren (Arduinokortet). Rx og Tx er tilkoblet to lysdioder som blinker når kortet kommuniserer med USB/PC, og tilsvarende vil skje når vi overfører data til programeditoren for monitorering av data på PC-skjermen (Rossing & Stausland, 2021, s. 30).

- **I^2C-databus**

  I^2C er en Inter IC-bus, og dette er en BUS som er svært enkelt å bruke med sine to linjer som består av klokke og datalinje. Hver krets har sin egen unike adresse som enten er satt av fabrikken, eller kan settes med strap'er på brikken. BUS'en er også utstyrt med kollisjonsdeteksjon, som i starten var definert med en hastighet på 100 kb/s, men ettersom at vi stadig trenger raskere dataoverføring har vi nå Fast mode som er 40 kb/s, og High speed som er på 3,4 Mb/s (Rossing & Stausland, 2021, s. 30).

- **SPI-databus**

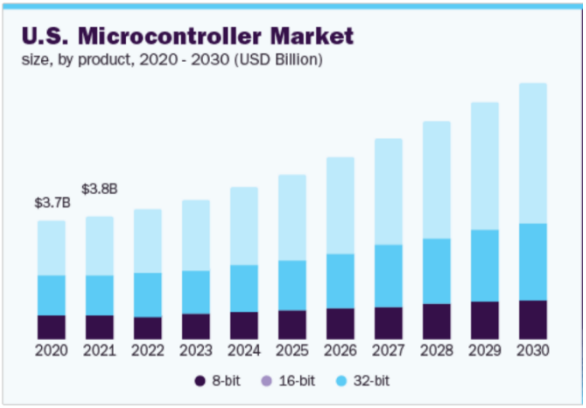
  SPI står for Seriel Peripheral Interface, og er en flerlinjers databuss som brukes for å overføre data til utstyr nedenfor mikroprosessoren, også kalt periferutstyr. Her fungerer den ene modulen som en master, sjef, og den andre modulen som slave, tjener.



Figur 37 SPI.

Master modulens oppgave er å ta initiativ til dataoverføring av SS-signallinjen, for så å sende sine data på en linje som heter MOSI, Master Output Slave Input. Slave modulens oppgave er å motta data på sin MOSI linje. Når data sendes mottar masteren svar på sin forespørsel til slaven som sender data tilbake på linjen MISO, Master Input Slave Output, som mottas av master på linjen MISO. Klokkesignalet har beskrivelsen SCLK og det er den som bestemmer takten til dataoverføringen. Slave Select signalet, SS-signalet, har som oppgave å varsle slaven om at master modulen ønsker kontakt. Er det flere slaver i et system trengs flere SS-linjer, en til hver slave. Figur 37 viser masteren på venstre side og slaven på høyre side (Rossing & Stausland, 2021, s. 31).

Flere og flere produkter benytter i større grad mikrokontrolleren i dagens samfunn, og den har fått en viktigere plass enn tidligere nettopp på grunn av at markedet er i stor vekst. I figuren viser grafen veksten i markedet for mikrokontrollere målt i amerikanske dollar, hvor det fortelles at det forventes en enorm vekst innen 2030 (Grand View Research, 2022).



Figur 38 Veksten i markedet for mikrokontrollere.

2.3 Samarbeid med NDLA

I starten av arbeidet med bacheloroppgaven begynte tanken om at vårt kompetansehefte kunne brukes som en ressurs i det norske skoleverket. Vi fikk et tips om å prøve muligheten hos NDLA. Tipset førte til en e-post med informasjon om kompetanseheftet samt. hvilke elever og nivå dette kan brukes på (Vg1 TIF og EL). Vi fikk et positivt svar fra NDLA og at dette var av verdi for deres nett-ressurs. Vi har en kontakt person i NDLA, Halvor Hove (H. Hove, personlig kommunikasjons, 16. Mars 2022). Han har kjempet en liten kamp for å få til et samarbeid mellom bransjer og skolene, for å fremme morgendagens teknologi. De ser på Arduino som en løsning for å enkelt støtte lærere og elever på ulike nivå på en god måte.

Samarbeidet med NDLA ble sett på som ønskelig, fordi som fremtidige lærere, er det viktig at kunnskapen som har blitt tilført kompetansehefte kan ha en praktisk betydning i skolen.

Samarbeidet med NDLA var viktig for å gjøre teksten og oppgavene gode nok for videregående opplæring.

2.4 Drøfting av yrkesfaglig fordypning

I dette kapittelet skal vi drøfte hva en Arduino er og videre skal det belyses hvorfor programmering har fått en så sentral rolle innen de teknologiske fagene EL og TIF.

Arduino er en plattform hvor vi har mulighet for prototyping av elektronikk, som baserer seg på maskin- og programvaren åpen kildekode. Arduino plattformen ble utviklet i Italia i 2005 (Wikipedia, 2022), og er et resultat av et prosjekt som er basert på Wiring-plattformen. Formålet med å utvikle Arduinoen var å skape en plattform og maskinvare som var billigere (Barragàn, 2016). Når en arbeider med Arduino kommer man innenfor temaet programmering. Programmering er en metode for å skrive, designe, feilsøke, teste eller vedlikeholde en kode i et program og koden skal gjerne tolkes av en datamaskin (Wikipedia, 2019). Programmering benyttes for å blant annet fortelle en datamaskin, eller en robot hva som skal skje, hvis gitte parameter er oppfylt (Learnlink, 2020).

Arduino er en rimelig måte for en skole å frembringe grunnleggende læring innen programmering (Skaperskolen, u.å.). Samtidig er C++ språket som brukes i Arduino en god plattform til nybegynnere innen programmering. En kan påstå at C++ språket har en viss fornuft innenfor hva en kan tenke seg til for å utføre en funksjon. Bokstavene, tallene og tegnene må være presise og inntreffe på en bestemt måte for at det skal fungere. Mange av funksjonene i koden er mulig å tenke seg til dersom en er orientert over hvilke valg en har i Arduino. Videre kan dette for mange være en prosess, hvor de etter hvert blir klare for mer avanserte programspråk og funksjoner på avanserte enheter. I dette segmentet finner vi blant annet PLS`er som i noen tilfeller brukes på store og avanserte systemer (Simplilearn, 2021).

Utfordringene med å bruke Arduino kan være ulikhetene mellom de forskjellige programmeringsspråkene som finnes. Sammenligner vi C++ språket med for eksempel Phyton kan en se at de har forskjellige tilnærminger til syntaks. Ved å bruke C++ kan en påstå at programmeringen er mer strukturert som gir en bedre oversikt over programkoden. Phyton organiserer språket ved å bruke innrykk pågående i programkoden, samtidig som

den har en mindre komplisert syntaks. En annen sammenligning er datahastigheten og minnehåndteringen som C++ er kjent for, en av grunnene til at C++ er raskere er at programkoden bruker mindre tid gjennom CPU. En enklere og mer systematisk programkode er mer effektiv for behandling. En må også huske på at bruken av de forskjellige språkene er tilpasset hva de skal brukes innen. C++ brukes oftest hvor det kreves høy hastighet og minnehåndtering, mens Phyton ofte brukes innen maskinlæring (Simplilearn, 2021).

Programmering har for fullt kommet inn i den videregående opplæringen. Dette på grunn av at i fagfornyelsen som kom i 2020 (Kunnskapsdepartementet, 2022) har de vendt blikket fremover på samfunnets behov for den teknologiske utviklingen. Yrkesfagretningene elektro- og datateknologi, og teknologi- og industrifag bærer preg av at fagfornyelsen har rettet søkelyset på programmering, noe som betyr at programmering vil være en ettertraktet kompetanse hos fremtidens fagarbeidere. Det er viktig for virksomheter å ha god tilgang på riktig og nødvendig kompetanse for at de skal kunne være mest mulig produktive og konkurransedyktige i næringslivet og samfunnet (NOU, 2020:2).

For å belyse programmering innen den videregående opplæringen har de valgt å bruke Arduino som programmeringsverktøy. Dette på bakgrunn av det som har blitt nevnt tidligere i avsnitt 2.1.1, fordi den er rimelig, utarbeidet pedagogisk materiell, den har blokk og tekstbasert programmering og det finnes mye tilleggsutstyr som er basert på Arduinoens plattform. Framtidens fagarbeidere skal inneha kompetanse om programmering og ikke nødvendigvis kompetanse om Arduino, men dette er en rimelig løsning for skolene (Skaperskolen, u.å.).

Hovedforskjellen for en elev som skal arbeide med programmering innenfor EL og TIF, er at EL eleven skal kunne både bygge og programmere et produkt som er selvvalgt. Dette produktet skal bestå av en mikrokontroller, analoge kretser, relevante sensorer og aktuatorer slik at vi oppnår ønsket virkemåte (Kunnskapsdepartementet, 2020a). Innenfor TIF så er det forståelsen av sammenhengen av et program og tilhørende sensorer, aktuatorer, CNC og variabler som er arbeidsoppgavene (Kunnskapsdepartementet, 2020b).

Grunnen til at programmering har fått en slik sentral betydning, er forbundet til flere årsaker. En av grunnene er at samfunnet blir mer teknologisk, og vi benytter i større grad datamaskiner som utfører komplekse og repeterende oppgaver mer effektivt enn

mennesker. En datamaskin eller robot, har også større arbeidskapasitet i og med at den kan arbeide hele døgnet. Et annet argument for at programmering har fått en sentral plass innen yrkesfaglige utdanningsprogram er veksten og utviklingen av smarthus. Ved å benytte smarthus kan vi få skreddersydd et program som vil gjøre dagen vår mer effektiv. Man kan også bruke smarthus for å programmere løsninger som er ment som hjelpemiddel til mennesker med funksjonsnedsettelse, og kan være med på å bidra til det grønne skiftet ved å bli mer miljøvennlig (Lindsø, 2020).

Mange mener at vi i dagens samfunn lever i en tid for den fjerde industrielle revolusjonen. Med dette menes at blikket er vendt mot digitalisering og gir en mer automatisert verden. En mer automatisert verden enn det vi nå lever i, inneholder trolig mer robotisering med fjerntilkopling og er utstyrt med kunstig intelligens. Alt dette finnes allerede, så en kan påstå at det ikke er en revolusjon, men en evolusjon av allerede eksisterende teknologi (TRIPLE-S, 2020). Kanskje er en slik evolusjon med på å forenkle hverdagen til mange, men for å vinne områder må prisen ned og kunnskapen til teknologien ut blant samfunnets borgere.

2.5 Overgang fra yrkesfaglig del til profesjonsfaglig del

I den yrkesfaglige delen har vi fokusert på framtidige behov for kompetanse innen programmering. Ut ifra dette valgte vi å belyse programmering ved å fordype oss i bredden innenfor Arduino. Dette førte til at vi utviklet et kompetansehefte som vi laget et undervisningsopplegg rundt, som vi brukte i vår aksjonering i FoU-arbeidet i profesjonsfaglig del. Videre i den profesjonsfaglige delen har vi fokusert på våre funn fra aksjoneringen, læringsteorier og læring om Arduino.

3.0 Profesjonsfaglig del

3.1 Bakgrunn og tema

Her skal vi ta for oss det pedagogiske utgangspunktet for selve undervisningsopplegget. Denne undervisningen baseres på gruppearbeid og oppgavene de skal jobbe med er hentet fra kompetanseheftet. Ved gruppearbeid bruker vi den sosiokulturelle læringsteorien og læreren har en observerende rolle.

Temaet for profesjonsfaglig del omhandler vårt arbeid med undervisningsopplegget som vi brukte i vår aksjonering. I aksjoneringen gjennomførte elevene en spesifikk oppgave som vi hentet ut fra kompetanseheftet. Vår rolle under aksjoneringen var å observere hvordan elevene brukte kompetanseheftet og hvordan de løste utfordringer som dukket opp underveis i arbeidet med oppgaven. Hensikten med disse observasjonene er at vi kan trekke de inn i vårt arbeide med å videre utvikle kompetanseheftet.

3.2 Problemstilling

På bakgrunn av temaet i profesjonsfaglig del har vi valgt å rette søkelyset på en problemstilling som handler om å legge til rette for læring om Arduino.

Problemstilling innen den profesjonsfaglige delen;

«Hvordan legge til rette for læring om Arduino på Vg1 EL og TIF?»

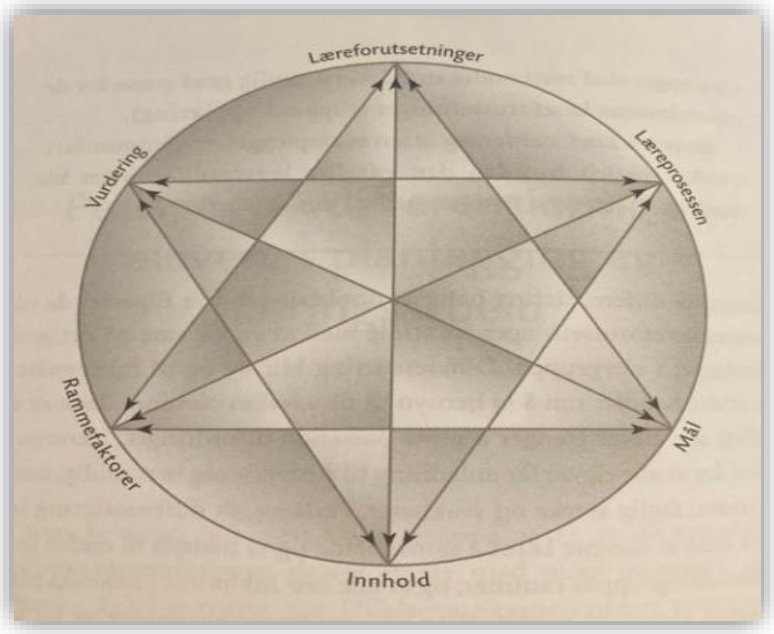
3.3 Oppbygning av profesjonsfaglig del

Oppbygningen består av et teoretisk rammeverk som danner grunnlaget for aksjoneringen. Gjennom arbeidet med aksjoneringen og FoU-arbeidet har vi brukt teoriene til å belyse hvordan vi kan legge til rette for læring om Arduino. I drøftingskapittelet vil vi reflektere rundt våre funn og hvordan vi kan legge til rette for læring om Arduino, ved bruk av kompetanseheftet.

3.4 Pedagogisk og yrkesdidaktisk teori

3.4.1 Den didaktiske relasjonsmodellen

Undervisningsøkten i denne oppgaven baseres på den didaktiske relasjonsmodellen, som vi anser som et viktig verktøy for planlegging, analyse og evaluering av god praktisk pedagogisk opplæring. I relasjonsmodellen er tilnærmingen av innholdet i undervisningen tilpasset elevene i klassen. Det legges til rette for å kunne få en praktisk og teoretisk erfaring innen programmering. Begynner læringen på feil kunnskapsnivå kan læringen innen emnet fort virke meningsløst og kjedelig (Idsøe, 2020, s. 16) . Den didaktiske relasjonsmodellen er ofte fremstilt slik:



Figur 39 Den didaktiske relasjonsmodellen. Hentet fra undervisningsplanlegging for yrkesfaglærere (Hiim & Hippe, 2018, s. 35).

Denne illustrasjonen viser det tydelig at alle punktene i modellen avhenger av hverandre. Mangler en strek eller pil vil en finne hull eller mangler i undervisningen. Et hvert punkt er like viktig for undervisningens kvalitet (Hiim & Hippe, 2009, s. 36). Punktene på bildet er grunnsteiner for et godt undervisningsopplegg og vil derfor bli gjennomgått i mer detalj i teksten under.

Elevenes læreforutsetninger

En kan kort forklare elevens læreforutsetninger til å være dens iboende kunnskap og interesser for emnet det skal gis læring innen. Det er viktig å kartlegge forhold som gjelder enkelteleven og elevgruppe, når man skal tilpasse arbeidet til elevene. Dette er faktorer som det er viktig å få kartlagt (Hiim & Hippe, 2018, s. 32). I vår planlegging og kartlegging hadde vi en samtale med kontaktlæreren for klassen vi skulle aksjonere hos. Kontaktlæreren ga oss

en indikasjon på elevenes kunnskapsnivå og enkelt elevers læreforutsetninger. Denne informasjonen ga oss en pekepinn på hva elevene kan fra før, og hva som er nytt lærestoff.

I opplæringsloven § 1-3 tilpassa opplæring, står det følgende; «Opplæringa skal tilpassast evnene og føresetnadene hjå den enkelte eleven, lærlingen, praksisbrevkandidaten og lærekandidaten» (Opplæringslova, 1998, § 1-3). Dette er viktig fordi elevene må holde seg innenfor flytsonen i opplæringen (Haaland & Nilsen, 2020, s. 198).

Vi har i dette undervisningsopplegget laget flere oppgaver, samt en kartlegging av elevens interesse for emnet i forkant. Slik kartlegging gir muligheter for differensiering og elevmedvirkning. Kartleggingen vil være til hjelp for hvordan en skal muliggjøre en undervisnings økt. Se vedlegg 1, planlegging- og veiledningsdokument.

Rammefaktorer

Hva gjør læring mulig? Rammefaktorene begrenses til forholdene som påvirker undervisningen i form av fysiske og tidsmessige begrensinger. En begrenses nesten alltid på tid, helt klart en avgjørende faktor innen kvaliteten for læringen. Videre er rammefaktorene de verktøyene og lokalene en har tilgang på, som klasserom eller verksteder læringen foregår i. En ser også på nødvendig utstyr som en rammefaktor. Ved hjelp av Arduino Uno, koplingsbrett, legobiler og noen datamaskiner skal elevene få prøve seg på programmering i klasserommet. Ved å klargjøre så mye som mulig i forkant, vil gi mer tid til læring i tidsrommet elevene skal gis undervisning om programmering (Hiim & Hippe, 2009, s. 33). Til sist må det nevnes en avgjørende faktor, lærerens kunnskap og formidlingsevne.

Mål

Det er stor sannsynlighet for at mange av elevene kan møte programmering i fremtidig yrke. Målet er en grunnleggende forståelse for hvordan programmering fungerer, kanskje så godt grunnlag at de er klare for å kunne begynne med egen læring på programmeringsfeltet. De vil underveis få trening i praktiske ferdigheter, samt teoretiske kunnskaper og gode holdningsverdier som kan være med på å forsterke kunnskapen til elevene i ettertid (Hiim & Hippe, 2009, s. 33).

Undervisningsarbeidet er rettet mot aktiviteter og læring som er bygget på læringsmål. Læringsmålene er hentet ut ifra kompetansemål i læreplanen (Hiim & Hippe, 2009, s. 64-65). Målene ser vi på som det elevene skal sitte igjen med av læring, etter endt læringsaktivitet. Det er viktig at målene er utarbeidet fra gjeldende kompetansemål i læreplanen (Hiim & Hippe, 2009, s. 33).

Innhold

Innholdet i undervisningen vi har planlagt vil bli presentert i kapittel 3.5, og vedlegg 1. Her vil vi se på hva som menes med innhold sett i lys av den didaktiske relasjonsmodellen. Ut ifra hva undervisningen skal handle om, skal man velge innhold og ut ifra dette skal man også velge hvordan innholdet blir tilrettelagt og lagt frem (Hiim & Hippe, 2009, s. 33).

Ved å la elevene velge mellom oppgaver med forskjellige vanskelighetsgrader, vil vår metode være med å bidra til at elevene kan få velge oppgaver ut ifra sitt kompetansenivå. Vg1 elever skal også få muligheten til å skaffe seg noen erfaringer til flere yrkesvalg, men i undervisningen vi har planlagt har vi vektlagt differensierte oppgaver i form av forskjellige vanskelighetsgrader (Hiim & Hippe, 2009, s. 86-87).

Læreprosess

Læreprosessen omhandler mange punkter, hva, hvordan og hvorfor utføre forskjellige prosesser innenfor læring. Den beskriver hva lærer og elev skal gjøre og hvordan samarbeidet mellom de skal være, og tiltakene som blir iverksatt skal begrunnes. Det er viktig å stille spørsmål rundt arbeidsform og aktivitetsform. Stiller vi slike spørsmål i planleggingen av undervisningen kan vi se for oss en aktiv læreprosess. I en aktiv læreprosess vil elevene i større grad arbeide selvstendig, og læreren går inn i en veilederrolle (Hiim & Hippe, 2009, s. 34). I vårt undervisningsopplegg har vi planlagt at elevene skal arbeide i grupper og vi vil veilede elevene underveis i arbeidet med oppgavene. På grunnlag av den sosiokulturelle læringsteorien rettes læringsaktiviteten mot gruppearbeid, hvor vi ønsker at eleven skal samarbeide og reflektere (Sylte, 2018, s. 158-159).

Vurdering

Vurdering vil være med på å gi oss svar på om undervisningen/læringen har fungert etter sitt formål, og foretas i forhold til læringsprosessen eller undervisningen (Hiim & Hippe, 2009, s. 34-35). Mange forbinder nok vurderinger med karakterer og prøver, som skal være med på å gi lærere en kontroll over elevenes kunnskaper. I dag har vi blitt introdusert for en rekke nyere vurderingsmetoder, som legger vekt på å vurdere arbeidsmetodene eller læreprosessen. Det er derfor viktig at det i forkant av undervisningen blir reflektert over hva som skal vurderes, hvorfor det skal vurderes og hvilken vurderingsmetode som skal brukes i vurderingen (Hiim & Hippe, 2009, s. 117).

Målet med vurdering er å fremme læring og bidra til lærelyst underveis hos elevene. Det skal gis informasjon og kompetanse underveis, og ved avslutningen av opplæringen i faget (Forskrift til opplæringslova, 2006, § 3-3).

Underveisvurdering er all vurdering som foregår før avslutningen av opplæringen. Underveisvurderingen skal være en integrert del av opplæringen, og formålet er å fremme læring, tilpasse opplæringen og øke kompetansen i fagene. Underveisvurderingen kan gjennomføres som både muntlig og skriftlig (Kunnskapsdepartementet, 2020).

3.4.2 Differensiert opplæring

Med differensiering menes det å gjøre ulikt (Haaland & Nilsen, 2020, s. 195). Det er omhandlet to hovedformer for differensiering i "Læring gjennom praksis" av Haaland og Nilsen. De to forskjellige er organisatorisk og pedagogisk differensiering. Organisatorisk differensiering er å dele inn elever i grupper etter kompetansenivå, egenskaper eller karakter. I stortingsmelding nr. 31, Kvalitet i skolen beskrives sammenhengen mellom tilpasset opplæring og læringsfellesskapet. "Tilpasset opplæring skal i all hovedsak skje innenfor rammen av fellesskapet, i klasser eller grupper (…) Opplæringen må legges opp slik at elevene kan dra nytte av at læring skjer i et sosialt arbeidsfellesskap der medelevene er ressurser i læringsarbeidet" (St.meld. nr. 31 (2007-2008), s. 74). Med disse føringene er organisatorisk differensiering lite ønskelig (Haaland & Nilsen, 2020, s. 195). I kompetanseheftet som vi har produsert har vi valgt å fokusere på pedagogisk differensiering, som betyr at det er tatt hensyn til elevenes læreforutsetninger og det gjør vi

ved å lage oppgaver med forskjellige vanskelighetsgrader. For eksempel kommer noen av arbeidsoppgavene med en godt beskrevet prosedyre for å komme fram til løsningen, mens noen har kun en oppgavetekst der elevene må tenke ut fremgangsmåter og løsninger selv. Mellom der kan det ligge oppgaver med noen retningslinjer som gjør det lettere å komme i gang i forhold til de rene tekstoppgavene. Å lage ekstra oppgaver som elevene har å velge i er en tradisjonell måte å differensiere på (Haaland & Nilsen, 2020, s. 199).

3.4.3 Motivasjon for læring

Ifølge Idsøe (Idsøe, 2020, s. 27) blir differensiering beskrevet i forhold til hvilket læringspotensial elevene har. Elevene blir inndelt i fire forskjellige nivå, stort og ekstraordinært læringspotensial, lavt læringspotensial, elever i midtre del og elever med minoritetsspråklig bakgrunn.

Uavhengig av hvilken grad av læringspotensial de ulike elevene har, så må undervisningen være tilpasset slik at alle har noe å hige etter (Damsgaard & Eftedal, 2018, s. 32). Men oppgavene kan heller ikke bli for vanskelige. Ved både for små og for store utfordringer, kan elevene havne utenfor flytsonen. I flytsonen har elevene kontroll på det de arbeider med, takler utfordringene de møter og er dermed i en gunstig læringssituasjon. Blir derimot oppgavene for vanskelige eller enkle, mister elevene flytsonen sin. Ved for lette oppgaver har elevene mer kunnskap enn det som trengs, de kjeder seg og får lite utbytte av læringen. Hvis oppgavene har en slik vanskelighetsgrad at eleven ikke har grunnlag for å klare å finne ut av det, kan de bli rådville, usikre og abdiserer (Haaland & Nilsen, 2020, s. 198).

3.4.4 Læringsteori

Sosiokulturell læring

Grunnleggende for sosiokulturell læring er at praktisk og teoretisk forståelse fører til utvikling på flere områder. Læringen er en gradvis læreprosess som ofte må repeteres, samtidig skaper den erfaring gjennom refleksjon og samhandling. Samhandlingen er i søkelyset og teorien baseres på at en er en større ressurs som gruppe enn enkeltindivid. På individnivå kan en lære gjennom dialog, dette kan videre føre til avklaring på egne tanker

samtidig som en får andre perspektiver på en problemstilling enn ens egen (Sylte, 2018, s. 158-159).

Oppgavene som brukes i vårt undervisningopplegg, bygger på denne læringsteorien. For at elevene skal lykkes må de samarbeide, samtidig må de ha en praktisk-teoretisk forståelse for hva oppgaven dreier seg om. I kompetanseheftet finner de all informasjon som trengs for å utføre oppgavene. Samarbeid og erfaringsveksling er en vesentlig faktor for å finne de rette komponentene for å få programkoden riktig. Ikke alle får denne riktig på første forsøk, da må de gjennom refleksjon og samarbeid oppdage hvor feilen ligger i arbeidet som ikke ble helt riktig. Oppgavene elevene løste ble utført på en induktiv tilnærmingsteori. Kjennetegnet for induktiv læring er at hjørnesteinen for utviklingen av kunnskap er at noe blir utført og de tilegner seg erfaringer. Motsetningen er deduktiv tilnærming der det først undervisers teoretisk får å så bruke den på problemstillinger (Haaland & Nilsen, 2020, s. 189). Vygotsky (Karlsdottir & Hybertsen, 2020, s. 255) påpeker at kultur, språk og fellesskapet er sammenhengen mellom læring og kunnskap i den sosiokulturelle tilnærmingen. Her vil språket være hovednøkkelen.

3.5 Undervisningsopplegg

Det er viktig at undervisningsopplegget knyttes til kompetansemålene i læreplanen for EL. Innenfor EL er det flere kompetansemål som støtter oppunder det læringsarbeidet som er tiltenkt undervisningsopplegget, nemlig Arduino. I vedlegg 1 finner man planleggings- og veiledningsdokumentet for undervisningsøkten.

3.6 Datainnsamling og metode

Vår metode for innsamling av data består både av en kvantitativ og en kvalitativ metode. Under aksjoneringen gikk vi for en kvalitativ metode, hvor vi observerte elevgruppen imens de jobbet med oppgavene de fikk utdelt. Ved å benytte en kvalitativ metode innhenter man informasjon om virkeligheten ved hjelp av ord og språk (Postholm & Jacobsen, 2018, s. 89). Vi fordelte gruppene oss imellom, slik at vi fulgte hver vår gruppe under aksjoneringen. Observasjonsfunnene ble diskutert oss imellom i etterkant av gjennomført aksjonering. Det ble ikke skrevet ned notater underveis i observasjonen. Fordelen med denne type

observasjon er at vi i større grad er aktive i elevenes læreprosess som veiledere, kontra om vi har hatt en rolle kun som observatør.

En kvantitativ metode baserer seg på at informasjonen fra virkeligheten formidles ved å bruke tall (Postholm & Jacobsen, 2018, s. 89). Før og etter aksjoneringen stilte vi elevene spørsmål, se vedlegg 3, som var utformet ut ifra en kvantitativ datainnsamlingsmetode (Postholm & Jacobsen, 2018, s. 171). Denne spørreundersøkelsen er også det nærmeste vi er en kvantitativ undersøkelse i denne bacheloroppgaven.

Vi gjennomførte også uplanlagte samtaler med kollegaer og faglærere om vårt FoU-arbeid. Disse dialogene og resultatene fra spørreundersøkelsene er informasjonen som vi har innhentet til arbeidet med denne oppgaven.

Spørsmål og svar fra spørreundersøkelsen finnes i vedlegg 3. Disse ble gitt for å undersøke behovet for et kompetansehefte.

3.7 Aksjonering

Undervisningsopplegget som har utgangspunkt i den yrkesfaglige fordypningen, har som et ledd i bacheloroppgaven blitt prøvd ut. Søkelyset i forbindelse med aksjoneringen har vært om kompetanseheftet har ført til læring om Arduino, og om det har vært et hjelpemiddel for elevene. Aksjoneringen ble gjennomført ved en Vg1 elektro- og datateknologi klasse, i faget elektroniske kretser og nettverk.

Aksjoneringen hadde et tidsaspekt på 4 undervisningstimer. I utgangspunktet var tanken å la elevene fritt velge hvilken oppgave de ville arbeide med fra kompetanseheftet. Dette med bakgrunn av differensiert opplæring (Haaland & Nilsen, 2020, s. 195). Underveis i planleggingen hadde vi en samtale med kontaktlæreren til klassen. Under denne kartleggingsprosessen kom det fram at det beste for elevgruppen var at de fikk arbeide med en bestemt oppgave. Vi satte sammen gruppene ved hjelp av kontaktlærer ut fra elevenes læreforutsetninger. På denne måten fikk vi inkludert de elevene som trengte litt mer tilpasning inn i en gruppe med faglig sterkere elever. Under aksjoneringen var det godt læringsmiljø, fordi elevene var engasjerte og aktive i læreprosessen. Dette kan komme av at vi valgte å benytte oss av legobiler som de skulle programmere. Årsaken til at vi valgte å

bruke legobiler var fordi i kartleggingen kom det frem at elevene var mindre motiverte for å arbeide med Arduino. Dette på grunn av lite variert opplæring tidligere innen emnet.

3.8 Anonymitet og etikk

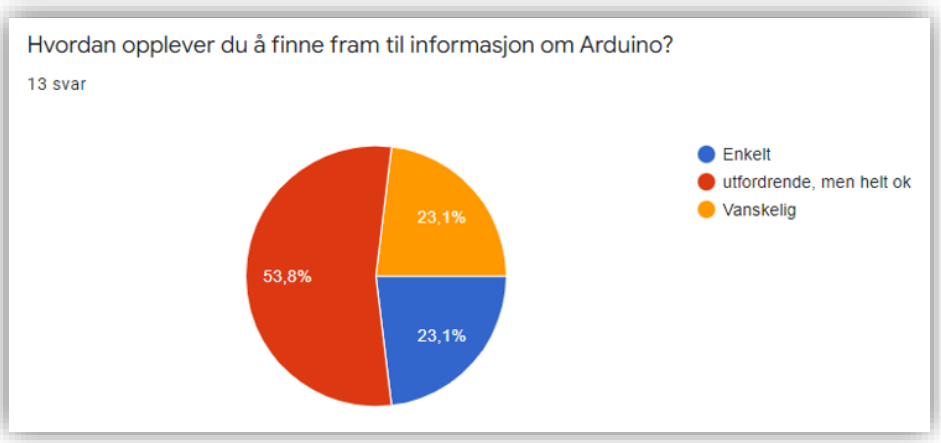
I forkant av den første spørreundersøkelsen ble elevgruppen informert om bakgrunnen for prosjektet, og de ble informert om at det var frivillig å delta og at alt av svar fra spørreundersøkelsen ville være anonymisert. Selv om undersøkelsen var frivillig å delta på, er det viktig å ta høyde for gruppepress. Postholm og Jacobsen (Postholm & Jacobsen, 2018, s. 248) definerer frivillighet som valg uten press fra noen andre. Videre påpeker de også viktigheten med full informasjon om undersøkelsen, for at folk skal kunne få velge fritt i å delta i en undersøkelse. I dette tilfellet var alle elevene enige i å delta i undersøkelsen, og spørreundersøkelsen ble gjennomført ved at de scannet en qr-kode som tok de videre til et spørreskjema i google forms.

3.9 Presentasjon av funn

Om vi ser på våre funn kan vi påstå at dette er en nyttig og god måte å formidle kunnskapen rundt emnet. Gjennom kartleggingen i forkant kunne vi legge til rette for at elevene fikk mer spesifikke detaljer rundt hvordan de skal finne informasjon om Arduino. Det siste spørsmålet var rettet mot nyttigheten av et kompetansehefte.

I forkant av aksjoneringen ble elevgruppen bedt om å svare på noen spørsmål, som vi skulle bruke i vårt planleggingsarbeid. Det første spørsmålet gikk ut på:

- Hvordan opplever du å finne fram til informasjon om Arduino?



Figur 40 Presentasjon av svar fra spørreundersøkelsen før aksjoneringen

Ut ifra dette spørsmålet kan vi konkludere med at over halvparten av elevene i klassen opplever det utfordrende å finne informasjon om Arduino.

Helt til slutt i spørreundersøkelsen stilte vi elevene spørsmålet:

- Tror du det ville vært nyttig med et kompetansehefte, hvor både pensum, oppgaver med forskjellige nivåer og løsningsforslag har vært samlet på 1 plass?

De svarene vi satt igjen med etter spørreundersøkelsen var:

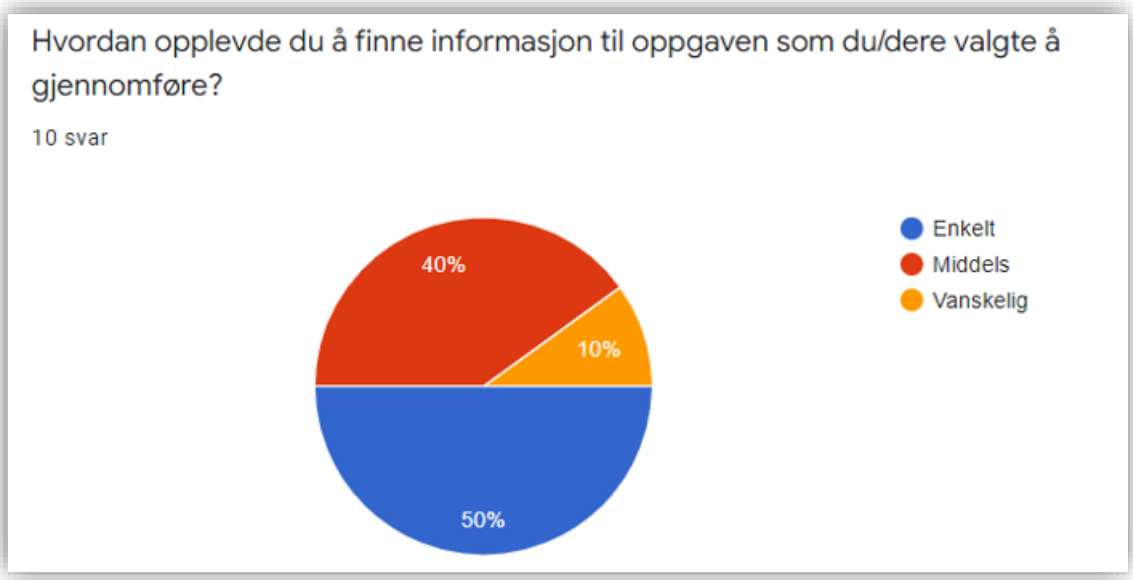
1. Flere elever opplever det som utfordrende å finne fram til informasjon om Arduino
2. De fleste elevene i klassen mente det ville vært nyttig med et kompetansehefte om Arduino

Figur 41 Presentasjon av funn fra spørreundersøkelsen før aksjoneringen

I etterkant av aksjoneringen ble elevene igjen bedt om å svare på en spørreundersøkelse. Disse spørsmålene gikk ut på nyttigheten av kompetanseheftet, og deres opplevelse av å bruke det.

Det første spørsmålet som var interessant for oss å få svar på var:

- Hvordan opplevde du å finne informasjon til oppgaven som du/dere valgte å gjennomføre?



Figur 42 Presentasjon av funn fra spørreundersøkelsen etter aksjoneringen

Her ser vi en betraktelig endring i forhold til det svaret vi fikk før aksjoneringen. Før aksjoneringen opplevde over halvparten av klassen at det var utfordrende å finne informasjon om Arduino. Under aksjoneringen skulle elevene bruke kompetansehefte i arbeidet med oppgavene. Svaret vi sitter igjen med etter at elevene har blitt kjent med kompetanseheftet er at halvparten opplevde det som enkelt å finne informasjon.

Det siste spørsmålet i den siste og avsluttende spørreundersøkelsen gikk ut på:

- Hvordan opplevde du å bruke kompetanseheftet?

Figur 43 Presentasjon av funn fra spørreundersøkelsen etter aksjoneringen

Etter den siste spørreundersøkelsen kan vi trekke en konklusjon med at:

1. Flere elever opplevde det enklere å finne informasjon med kompetanseheftet
2. De fleste elevene hadde en positiv opplevelse av å bruke kompetanseheftet

Funn i observasjonen bekrefter elevenes positive opplevelse og nyttigheten av kompetanseheftet. Under aksjoneringen ble det observert at bruk av kompetanseheftet var enklere enn å bruke google. I denne situasjonen oppstod det et faglig spørsmål rundt oppgaven. Den ene eleven benyttet internett for å søke svar, mens den andre eleven fant svaret raskere ved å bruke kompetanseheftet og viste dette til de andre elevene i gruppen.

Vi har også hatt uformelle samtaler med kollegaer angående vårt FoU-arbeid underveis i prosessen med bacheloroppgaven. Der kom det også fram at de savnet en læringsressurs hvor de kunne finne informasjon og differensierte arbeidsoppgaver med løsningsforslag.

3.10 Svakheter og styrker ved oppgaven

Denne oppgaven har både styrker og svakheter. Gjennom samtaler med kollegaer, Halvor Hove i NDLA, tilbakemeldinger fra elever og framtidige samfunnsbehov anser vi kompetanseheftet i sin helhet som en styrke. Vi kan også se igjennom vårt FoU-arbeid at kompetanseheftet er en læringsressurs. En annen styrke ved denne oppgaven er at NDLA har sagt seg villig til et samarbeid for å senere publisere deler av kompetanseheftet som en læringsressurs på sine hjemmesider.

Svakheter ved oppgaven er at vi under aksjoneringen ikke noterte våre observasjoner. Vi har i ettertid av aksjoneringen reflekter i hvor stor grad vi burde fordelt observasjons- og veilederrollen oss mellom. Dette ville ha gitt oss flere vinklinger og refleksjoner rundt aksjoneringen. I tillegg var det få deltakere med tanke på våre funn. Her burde vi aksjonert på flere skoler for å oppnå en høyere validitet av vårt forskningsarbeid.

Tidsaspektet for aksjoneringen er også en svakhet i denne oppgaven. Dette på grunn av at vi ikke fikk testet hele kompetanseheftet på de 4 undervisningstimene vi aksjonerte på. Kompetanseheftet er tilpasset et lengre læringsperspektiv.

Kanskje hadde det vært en fordel å vært mer grundig i vårt arbeid med kartleggingen og spørreundersøkelsene. Med dette mener vi at det kunne vært hensiktsmessig å gjennomført dybde intervjuer med elever, i stedet for undersøkelsesformen vi benyttet. Resultatet av vårt forskningsarbeid var muligens påvirket av måten vi formulerte spørsmålene i kartleggingen.

3.11 Profesjonsfaglig drøfting

I denne delen av besvarelsen vil vi drøfte våre funn opp imot det teoretiske grunnlaget. Videre vil vi reflektere over vår læringsprosess og utviklingsarbeid med et forskende blikk. Innledningsvis og gjennom teorien presenterer denne oppgaven et forslag på hvordan vi ved å bruke den didaktiske relasjonsmodellen (Hiim & Hippe, 2009, s. 36) planla, gjennomførte og evaluerte et undervisningsopplegg. Undervisningsopplegget var et ledd i vårt FoU-

arbeidet, hvor vi ønsket å forske på hvordan vi kan legge til rette for læring om Arduino på Vg1 EL og TIF. Når det kommer til utdanningsprogram som er yrkesfaglige påstår Hiim og Hippe (Hiim & Hippe, 2018, s. 42-43) at elevene ofte foretrekker at læringsarbeidet skal foregå i et praktiskorientert miljø. Hvis vi som lærere baserer undervisningen på ren teoriformidling, kan det være med på å ta fra elevene motivasjonen. Baserer man opplæringen på noe som virker meningsfylt for deres fremtidige yrke, vil vi kunne oppleve at elevene får en økt læringsvilje (Hiim & Hippe, 2018, s. 42-43).

3.11.1 Planleggingsfasen

I planleggingsfasen av denne undervisningsøkten startet vi med å kartlegge elevenes læreforutsetninger (Hiim & Hippe, 2018, s. 32). Vi hadde en samtale med klassens kontaktlærer som ble et viktig bindeledd for oss. Læreren satt på viktig informasjon som gjorde at vi enklere kunne tilpasse undervisningsopplegget til elevgruppen. I klassen var det elever som hadde ulike utfordringer og læreforutsetninger, derfor mente vi det var viktig at kontaktlærer var med å sette sammen gruppene. Utfordringene som disse elevene hadde, var ADHD og de var avhengige av å ikke komme på samme gruppe. Dette ville muligens ha ført til lite læring hos de gjeldende elevene. Parallelt med kartleggingen som ble gjennomført sammen med kontaktlæreren, utførte vi også en spørreundersøkelse som elevene ble bedt om å svare på. Ut ifra dette klarte vi å danne oss et bilde av kompetansenivået i klassen og lettere legge til rette for å skape en undervisning som holder elevene innenfor flytsonen (Haaland & Nilsen, 2020, s. 198).

Rammefaktorene (Hiim & Hippe, 2009, s. 33) for denne undervisningen var tidsbruk, utstyr, klasserom og vår egen kompetanse innen faget. Gjennom vår fordypning innen Arduino i den yrkesfaglige delen og utviklingen av kompetanseheftet, har vi tilegnet oss relevant kunnskap som vi kan anvende i undervisningssammenheng. For at vi skulle kunne planlegge undervisningen godt og være sikre på at vi hadde nok utstyr til alle gruppene, måtte vi styre hvilke oppgaver de skulle arbeide med.

Kompetanseheftet utformet vi med differensierte oppgaver (Haaland & Nilsen, 2020, s. 195), hvor målet var å gi elever med forskjellige utgangspunkt størst mulig læringsutbytte om Arduino. Elevene vil kunne finne arbeidsoppgaver som møter deres interesser og faglige nivå.

For å oppnå best mulig relasjon til elevene på kort tid, valgte vi å være veiledere for en gruppe hver. Dette for å få en relasjon mot et mindre antall elever enn vi hadde fått ved å bytte gruppe underveis. Fordelen var at enhver veileder opparbeidet inngående kunnskap rundt nivået og interessene internt i gruppene.

3.11.2 Gjennomføringsfasen

Hvor lang tid en klasse bruker på å komme i flytsonen (Haaland & Nilsen, 2020, s. 198) vil variere og det kan være vanskelig å konkludere med noe etter en aksjonering på fire timer. Våre observasjoner tilsier at elevene hadde en positiv innstilling og var motiverte. Dette kunne vi se på hvordan de valgte å jobbe med oppgaven og interessen for å arbeide med Arduino. Kanskje kom dette på grunn av forberedelsene som vi gjorde i planleggingsfasen.

Utgangspunktet for undervisningen var at elevene selv skulle få velge oppgave med tanke på differensiering (Haaland & Nilsen, 2020, s. 195), men på grunn av tidsrammen på fire timer valgte vi å tildele første oppgave som elevene skulle utføre. Etter at elevene utførte oppgaven vi hadde valgt ut, fikk de som hadde tid igjen velge en oppgave fra kompetanseheftet som de fikk arbeidet med ut ifra utstyret de hadde til rådighet. Det ble her valgt litt forskjellige oppgaver, men ingen av de største og mest avanserte på grunn av både tid og kunnskapsnivå. Da observerte vi at gruppene valgte forskjellige oppgaver, som kunne være innenfor deres interessefelt og deres faglige nivå.

Undervisningens innhold (Hiim & Hippe, 2009, s. 33) var at elevene skulle i første del programmere en forhåndskoplet legobil med dioder og to RGB-dioder. Diodene skulle forestille fremlys og baklys. Disse var hvite og røde. På taket var det to RGB-dioder som varsellys i forskjellige farger. Grunnen til at vi valgte å lage bilene før undervisningen var at elevene skulle ha søkelys på programkoden og ikke arbeidet med oppkoplingen eller byggingen av bilene.

I selve læreprosessen (Hiim & Hippe, 2009, s. 34) valgte vi å gå for en sosiokulturell tilnærming (Sylte, 2018, s. 158-159) i vår undervisning. Vi var også interesserte i at dette skulle åpne opp muligheten for å la elevene trene på sosiale og kognitive ferdigheter. Vi anser sosiokulturell tilnærming for en godt egnet læringsteori for denne undervisningsøkten, fordi elevene vil i større grad få muligheten til fri tenkning, prøve ut og feile på egenhånd.

Dette i motsetning til sosial kognitiv læring (Karlsdottir & Hybertsen, 2020, s. 213) som baserer seg på at elevene observerer en handling for så å imitere. Elevene fikk ikke noen teoretisk gjennomgang i forkant av undervisningen og ble bedt om å starte på en angitt oppgave i kompetanseheftet, som baserer seg på en induktiv tilnærming (Haaland & Nilsen, 2020, s. 189). Hensikten vår med å velge en induktiv tilnærming var å la elevene utføre noe, for så å tilegne seg kunnskap og erfaringer. Mye av dette henger sammen med Vygotskys (Karlsdottir & Hybertsen, 2020, s. 255) syn på læring om at kultur, språk og fellesskapet er sammenhengen mellom læring og kunnskap i den sosiokulturelle tilnærmingen.

Elevene fikk arbeide med oppgavene i grupper fordi vi ønsket å gi de muligheten til å spille hverandre gode og samarbeide. Dette samarbeidet observerte vi i aksjoneringen og vi tar det videre med oss i utviklingen av kompetanseheftet. Med dette vil vi satse på å få utviklet flere egnede gruppeoppgaver og ulike pedagogiske tilnærminger.

3.11.3 Evaluering

Målet (Hiim & Hippe, 2009, s. 33) med undervisningen var å legge til rette for læring om Arduino, ved at elevene skulle arbeide med oppgaver ut ifra kompetanseheftet. Vi laget et delmål for elevene som gikk ut på at de skulle programmere lysstyringen til en bil. Vårt mål med undervisningen var å gjøre observasjoner som vi kunne bruke i vårt forskningsarbeid for å kartlegge hvordan elevene arbeidet med oppgavene og innhente informasjon om Arduino, fra kompetanseheftet. Dette ville være nyttige observasjoner for oss, med tanke på videre utvikling både av undervisningsopplegget og kompetanseheftet. Oppgavene som elevene fikk var tilpasset deres grunnleggende ferdigheter, noe vi fikk bekreftet av kontaktlærer da han fikk se oppgavene vi hadde laget for elevene.

Tidsrammen for undervisningen var på fire undervisningstimer. I etterkant av aksjoneringen ser vi at dette ikke var tilstrekkelig med tid for å få testet kompetanseheftets omfang. Det burde blitt satt av mye lengre tid og blitt testet på flere klasser enn bare en.

Elevene ble ikke stilt overfor en prøve eller test som vurdering (Hiim & Hippe, 2009, s. 34-35) etter denne undervisningsøkten. Vår hensikt med vurderingen av undervisningen var å finne ut om elevene opplevde læring om Arduino ved å benytte seg av kompetanseheftet. Ut ifra våre funn observerte vi at elevene løste oppgaven vi ga dem på en god måte ved hjelp av

kompetanseheftet. Dette kunne vi se i hvordan gruppene arbeidet sammen og støttet opp hverandre ved utfordringer. Elevene ble også bedt om å svare på en spørreundersøkelse i etterkant av undervisningen. Svarene fra undersøkelsen signaliserer at elevene opplevde det enklere å innhente informasjon om Arduino i kompetanseheftet. Dette ser vi på resultatene i vedlegg 3, hvor det i forkant av aksjoneringen kom frem at over halvparten av klassen syntes det var utfordrende å finne informasjon om Arduino.

En av oss observerte at den ene gruppen kom ut for en utfordring, hvor de måtte innhente informasjon om motstandsverdier. Den ene eleven var rask med å ta frem telefonen for å gjøre et kjapt internettsøk, mens den andre på gruppen tok frem kompetanseheftet. Eleven som brukte kompetanseheftet, fant frem til et svar raskere enn eleven som ville søke på internett. Dette ga oss indikasjon om at kompetanseheftet var på elevens nivå og brukervennligheten var god. Det som er en svakhet i denne oppgavens forskning, er at vi burde delegert observasjonsoppgaver oss imellom og det burde blitt tatt notater. Det vi sitter igjen med av funn baserer seg på vår hukommelse fra aksjoneringen og svar som elevene ga oss i etterkant via en spørreundersøkelse.

4.0 Avslutning

I dette avsluttende kapittelet kommer vi først med en liten oppsummering av oppgaven, videre presenterer vi vår konklusjon og svar på problemstillingene. Avslutningsvis ser vi fremover og kommer med noen refleksjoner om veien videre.

4.1 Oppsummering

Utgangspunktet for denne studien er to-delt. I del en som er den yrkesfaglige fordypningen vår, ville vi belyse hva Arduino er og hvorfor programmering har blitt sentralt innen de teknologiske fagene EL og TIF. I del to tar vi for oss en profesjonsfaglig del som bygger på et undervisningsopplegg vi gjennomførte i forbindelse med vårt FoU-arbeid. I den profesjonsfaglige delen var vi interessert i å se på hvordan man kan legge til rette for læring om Arduino innen Vg1 EL og TIF.

I arbeidet med disse to delene utarbeidet vi et kompetansehefte innen Arduino for elever og lærere ved Vg1 EL og TIF. Dette heftet inneholder først en teorikappe hvor vi tar for oss hva Arduino er, komponentlære, og videre kan man finne arbeidsoppgaver for elevene. Her finner man arbeidsoppgaver som er ment for hver enkelt fagretning, fordypningsoppgaver og tverrfaglige oppgaver. Samfunnet blir mer og mer teknologisk og den teknologiske utviklingen vil kreve at morgendagens fagarbeidere har kunnskap innen utviklingen på fagområdet. Programmering er i denne bacheloroppgaven hovedtema og søkelyset er rettet mot hvordan kan man gjøre opplæringen i emnet tilpasset for skolen og elevene. Samtidig som vi har jobbet med bacheloroppgaven har vi også opprettet et samarbeid med NDLA, om at de skal publisere deler av oppgavene som vi har utarbeidet slik at det kan være en ressurs for andre skoler, lærere og elever.

4.2 Konklusjon

I denne bacheloroppgaven har vi presentert vårt FoU-arbeid igjennom en yrkesfaglig fordypning, og en profesjonsfaglig del. Dette arbeidet har ført til at vi har utviklet et kompetansehefte som er ment for elever og lærere ved Vg1 EL og TIF, som omhandler læring om Arduino.

Problemstillingen vår for den yrkesfaglige fordypningen var;

«Hva er Arduino, og hvorfor har programmering blitt så sentralt innen de teknologiske fagene EL og TIF?».

Som svar på denne problemstillingen kan vi konkludere med at Arduino er en rimelig og god plattform for å gjennomføre læring om programmering innen videregående opplæring. Samtidig er utviklingen av samfunnet rettet mot en mer teknologisk hverdag, som krever at framtidens fagarbeidere innen EL og TIF har kompetanse om programmering.

Problemstillingen vår for den profesjonsfaglige delen var;

«Hvordan legge til rette for læring om Arduino på Vg1 EL og TIF?».

Som svar på denne problemstillingen kan vi ut ifra våre funn fra aksjoneringen konkludere med at kompetanseheftet legger til rette for læring om Arduino. Noe av grunnen til dette kan argumenteres om vi ser på spørreundersøkelsene vi gjorde før og etter aksjoneringen. Funnene fra før aksjoneringen og funnene fra etter aksjoneringen viser at elevene hadde utbytte av kompetanseheftet. Vi kan også påstå at kompetanseheftet er en god samling av teori og oppgaver til elever og lærere samlet i ett hefte.

4.3 Veien videre

Veien videre for vår del blir å opprettholde kontakten med NDLA, for å fortsette samarbeidet og videreutvikle kompetanseheftet ytterligere. Som forfattere av denne oppgaven har engasjementet steget i form av at vi ser verdien i kompetanseheftet som vi produserte. Derfor vil det være naturlig for oss å utvikle kompetanseheftet enda mer i ettertid. En kan da rette kompetanseheftet mot hvilke pedagogiske metoder en lærer kan bruke for å utøve læring. En kan her tenke seg til at en grunnleggende kunnskap rundt programmering kan være nyttig for de fleste i vårt samfunn, om ikke alle skal programmere så i alle fall tanken på hvordan en kan utnytte teknologien.

Referanser

Almås, S. & Rossen, E. (2021, 08. Februar 2021). *C++*. Store Norske Leksikon. https://snl.no/C++

Arduino.cc. *Structure*. Hentet 02 Mars fra https://www.arduino.cc/reference/en/#structure

Arduino.cc (2019). *Loop.* Hentet 24.04.2022 fra https://www.arduino.cc/reference/en/language/structure/sketch/loop/

Arduino.cc (2019). *Setup.* Hentet 24.04.2022 fra https://www.arduino.cc/reference/en/language/structure/sketch/setup/

Arduino.cc (2019). *Switchcase*. Hentet 02.03.2022 fra https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/

https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/

Arduino.cc (2020), *If*. Hentet 12.03.2022 fra https://www.arduino.cc/reference/en/language/structure/control-structure/if/

Arduino.cc (2020). *While*. Hentet 12.03.2022 fra https://www.arduino.cc/reference/en/language/structure/control-structure/while/

Arduino.cc (2021). *Do…while.* Hentet 12.03.2022 fra https://www.arduino.cc/reference/en/language/structure/control-structure/dowhile/

Arduino.cc. (2022). *Using Variables in Sketches*. Arduino.cc. Hentet 24.04.22 fra https://docs.arduino.cc/learn/programming/variables

Barragàn, H. (2016). *The Untold History of Arduino*. Hentet 24.04.22 fra https://arduinohistory.github.io/

Forskrift til opplæringslova. (2006). *Forskrift til opplæringslova* (FOR-2006-06-23-724).

Lovdata. https://lovdata.no/dokument/SF/forskrift/2006-06-23-724

Haaland, G. & Nilsen, S. E. (2020). *Læring gjennom praksis*. Pedlex.

Halvorsen, H.-P. (2018). Arduino Programmering. *Arduino Programmering*. https://www.halvorsen.blog/documents/technology/resources/resources/Arduino/Arduino%20Programmering.pdf

Haraldsrud, A. D., Sveinsson, H. A. & Løvold, H. H. (2020). *Programmering i skolen*. Universitetsforlaget.

Hiim, H. & Hippe, E. (2009). *Undervisningsplanlegging for yrkesfaglærere* (3, Red.). Gyldendal Norsk Forlag.

Idsøe, E. C. (2020). *Differensiereing i skolen*. Cappelen Damm AS.

Karlsdottir, R. & Hybertsen, I. D. (Red.). (2020). *Læring - utvikling - læringsmiljø*

En innføring i pedagogisk psykologi. Vigmostad & Bjørke AS.

Kunnskapsdepartementet. (2020). *God underveisvurdering*. https://www.udir.no/laring-og-trivsel/vurdering/om-vurdering/underveisvurdering/

Kunnskapsdepartementet. (2020). *Søk i læreplaner*. https://sokeresultat.udir.no/finn-lareplan.html?fltypefiltermulti=Kunnskapsl%C3%B8ftet%202020&utdanningsprogram=VGO%20yrkesfag%20LK20%3BElektro%20og%20datateknologi%3BTeknologi-%20og%20industrifag%3BHelse-%20og%20oppvekstfag

Kunnskapsdepartementet. (2021). *Hva er nytt i teknologi- og industrifag?* https://www.udir.no/laring-og-trivsel/lareplanverket/fagspesifikk-stotte/nytt-i-fagene/hva-er-nytt-i-teknikk-og-industriell-produksjon/#a160013

Kunnskapsdepartementet. (2022). *Hva er nytt i elektro og datateknologi?* https://www.udir.no/laring-og-trivsel/lareplanverket/fagspesifikk-stotte/nytt-i-fagene/hva-er-nytt-i-elektrofag/#a159978

Kunnskapsdepartementet. (2022). *Innføring og overgangsordninger for nye læreplaner*. https://www.udir.no/laring-og-trivsel/lareplanverket/innforing-og-overgangsordninger-for-nye-lareplaner/

Kunnskapsdepartementet. (2020a). *Kompetansemål etter elektroniske kretser og nettverk*. Udir. https://www.udir.no/lk20/ele01-03/kompetansemaal-og-vurdering/kv254?Kjerneelementer=true&lang=nob

Kunnskapsdepartementet. (2020b). *Kompetansemål etter konstruksjons- og styreteknikk*. Udir. https://www.udir.no/lk20/tip01-03/kompetansemaal-og-vurdering/kv252

Learnlink. (2020). *Hva er programmering?* [Video]. Youtube. https://www.youtube.com/watch?v=v7AdA9mG0Is&t=7s

Lindsø, J. F. (2020). *Hvorfor programmere?* NDLA. Hentet 25.04.22 fra https://ndla.no/nb/subject:1:5a5cac3f-46ff-4f4d-ba95-b256a706ec48/topic:c7111b35-0621-4977-8a31-fe41e8e4a34d/resource:7e116b54-1c49-4a04-b9c3-250791824228

Lindsø, J. F. (2020). *Programmeringshistorie*. NDLA. https://ndla.no/nb/subject:1:1352b19e-e706-4480-a728-c6b0a57ba8ae/topic:1:e08eecc8-5e7e-4b85-9876-dfc5d1f3d920/resource:c26ebd1d-3a7f-4116-b7ff-44778098f45b

Lindsø, J. F. & Bårdgård, T. (2019). *Variabler*. NDLA. https://ndla.no/nb/subject:1:5a5cac3f-46ff-4f4d-ba95-b256a706ec48/topic:c7111b35-0621-4977-8a31-fe41e8e4a34d/resource:b515c789-bcbc-43d3-8597-acae75a6076a

NOU 2020:2. (2020). *Fremtidens kompetansebehov III.* Kunnskapsdepartementet. https://www.regjeringen.no/contentassets/053481d65fb845be9a2b1674c35d6d14no/pdfs/nou202020200002000dddpdfs.pdf

Opplæringslova. (1998). *Lov om grunnskolen og den vidaregåande opplæringa (opplæringslova)* (1998-07-17-61). Lovdata. https://lovdata.no/dokument/NL/lov/1998-07-17-61#KAPITTEL_3

Postholm, M. B. & Jacobsen, D. I. (2018). *Forskningsmetode for masterstudenter i lærerutdanning*. Cappelen Damm AS.

Research, G. V. (2022, Februar 2022). *Microcontroller Market Size, Industry Report, 2022-2030*. Grand View Research. https://www.grandviewresearch.com/industry-analysis/microcontroller-market

Rossen, E. (2020, 4. august). *Assemblerspråk*. https://snl.no/assemblerspr%C3%A5k

Rossing, N. K. (2014). *Arduino ressurshefte - Atmel*. Skolelaboratoriet. https://www.ntnu.no/documents/2004699/12108297/Atmel+2.2.pdf/d5a8f595-8b93-41e8-9b28-71c776551e01

Rossing, N. K. & Stausland, C. (2021). *Nordic ESERO Arduino - Grunnkurs programmering (CanSat)*. Skolelaboratoriet. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

Simplilearn. (2021, 29.10.21). *C++ Vs. Python: Overview, Uses & Key Differences*. https://www.simplilearn.com/tutorials/cpp-tutorial/cpp-vs-python?source=sl_frs_nav_playlist_video_clicked

Skaperskolen. (u.å.). *Mikrokontrollere*. Skaperskolen.

St.meld. nr. 31 (2007-2008). *Kvalitet i skolen*. Kunnskapsdepartementet. https://www.regjeringen.no/no/dokumenter/stmeld-nr-31-2007-2008-/id516853/

Sylte, A. L. (2018). *Profesjonspedagogikk* (2. utg.). Gyldendal Norsk Forlag AS.

TRIPLE-S. (2020). *Industriell digitalisering: Hva det er, hvorfor det er viktig og hvordan komme i gang*. TRIPLE-S. https://f.hubspotusercontent10.net/hubfs/3956663/Triple-S%20eBok%20Industriell%20digitalisering%20200803%20v5.pdf?__hstc=48107704.1776cfc2b3821141b74ee5a317618425.1648136607830.1648136607830.1648136607

830.1&hssc=48107704.1.1648136607830&hsfp=2264378951&hsCtaTracking=db96b7bf-cfc6-4a8a-9f13-4ad8b7b04826%7Cb43cf12c-ed06-446d-8407-145b6cb5db8d

Venås, H. & Vangsnes, S. (2020). *Elektroniske kretser og nettverk*. Elforlaget.

Wikipedia. (2019). *Programmering*. Wikipedia. Hentet 24.04.22 fra https://no.wikipedia.org/wiki/Programmering

Wikipedia. (2022, 24. feb.). *Arduino historie*. Wikipedia. Hentet 05.04.22 fra https://no.wikipedia.org/wiki/Arduino

Referanser til figurer

Figur	Beskrivelse	URL	Lisens	Side
Forside	Arduino UNO kort	https://www.google.com/search?q=Arduino+UNO+kort&rlz=1C1GCEU_noNO873NO873&sxsrf=ALiCzsYioyvzpXcJpY_UAuwzmNpOvZ7NUA:1653290171763&source=lnms&tbm=isch&sa=X&ved=2ahUKEwiV5ZfrifX3AhXRSvEDHVOqBOIQ_AUoAXoECAEQAw&biw=1920&bih=969&dpr=1#imgrc=hmrxw1zbjNymrM		Forside
1	Den første Arduino prototypen	https://arduinohistory.github.io/		16
2	Eksempel på forklaring i programmet	Selvlaget		18
3	Eksempel på kode	Selvlaget		19
4	Eksempel på switch-case	https://www.programiz.com/cpp-programming/switch-case		19
5	Eksempel på do-while-løkke	Selvlaget		20
6	Eksempel på en while-løkke	Selvlaget		20
7	Eksempel på enveis if-setning	https://www.ntnu.no/wiki/download/attachments/78972107/Uke38_Matlab_4.pdf?version=2&modificationDate=1474279615000&api=v2.		20

8	Eksempel på if-setning i en sketch	Selvlaget		20
9	Eksempel på en if-else-setning i en sketch	Selvlaget		21
10	Eksempel på en toveis if-setning	https://www.ntnu.no/wiki/download/attachments/78972107/Uke38_Matlab_4.pdf?version=2&modificationDate=1474279615000&api=v2.		21
11	Eksempel på HIGH eller LOW	Selvlaget		22
12	Eksempel på en programkode hvor det er satt betingelser	Selvlaget		22
13	Eksempel på forskjellige variabler	Selvlaget		23
14	Illustrasjon av tre variabler	Selvlaget		23
15	Behandlingsenhet	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.		24

		pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		
16	Databehandlingsenhet	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		25
17	Koblingsbrett	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		26
18	Arduino UNO kort	https://www.elfadistrelec.no/en/microcontroller-board-uno-arduino-a000066/p/11038919?ext_cid=shgooaqnoen-Shopping-campaign&gclid=CjwKCAiApfeQBhAUEiwA7K_UHwjGqcRmgn4TqUp6OJuY8U-gNasInXID3L6qUWvFA5JfwZCQi7au7hoCvBgQAvD_BwE		26
19	Jumper wire	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		26
20	Motstand	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		26

21	Motstandsverdier	https://www.tek.no/nyheter/guide/i/MRpbRr/elektronikkens-verden-del-1?page=3		26
22	Potensiometer	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		27
23	Bryter	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		27
24	Fotomotstand	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		27
25	Temp. sensor	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		28
26	Diagram av spenning som funksjon	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		28
27	Transistor	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.		28

		pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		
28	Motor	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		29
29	Servo	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		29
30	Lys dioder	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		30
31	Piezo buzzer	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		30
32	Analogt signal	http://mucins.weebly.com/31-analog-and-digital.html		31
33	Digital presentasjon av analogt signal	https://www.halvorsen.blog/documents/technology/resources/resources/Arduino/Arduino%20Programmering.pdf		31
34	Forskjellige verdier på	https://www.halvorsen.blog/documents/technology/resources/resources/Arduino/Arduino%20Programmering.pdf		31

	et digitalt signal			
35	Konvertering av analogt signal til digitalt signal	https://www.halvorsen.blog/documents/technology/resources/resources/Arduino/Arduino%20Programmering.pdf		32
36	Mikrokontrollerkort	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		32
37	SPI	https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218		34
38	Veksten i markedet for mikrokontrollere	https://www.grandviewresearch.com/industry-analysis/microcontroller-market		35
39	Den didaktiske relasjonsmodellen	Hiim, H. & Hippe, E. (2009). *Undervisningsplanlegging for yrkesfaglærere* (3, Red.). Gyldendal Norsk Forlag.		40
40	Presentasjon av funn fra spørreunde	Selvlaget		48

	rsøkelsen før aksjonering en			
41	Presentasjo n av funn fra spørreunde rsøkelsen før aksjonering en	Selvlaget		49
42	Presentasjo n av funn fra spørreunde rsøkelsen etter aksjonering en	Selvlaget		49
43	Presentasjo n av funn fra spørreunde rsøkelsen etter aksjonering en	Selvlaget		50

Vedlegg

Vedlegg 1 – Planleggings- og veiledningsdokument

Malen for denne er hentet fra Haaland og Nilsen (2013, s. 64-65)

<table>
<tr><td>Klasse: VG1 Elektro og datateknologi</td><td colspan="2">Fag: Elektroniske kretser og nettverk</td></tr>
<tr><td colspan="3">Tema: Arduino legobil med LED-lys</td></tr>
<tr><td>Dato: Uke 12</td><td colspan="2">Time: 3-6 time (10.00-13:45)</td></tr>
<tr><td colspan="3">Elevenes læringsmål:</td></tr>
<tr><td colspan="3">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

Digitale ferdigheter i Vg1 elektro og datateknologi innebærer å etablere digital kommunikasjon, utføre feilsøking og simulere og programmere ved hjelp av digitale verktøy. Digitale ferdigheter innebærer også å kunne produsere tegninger og tekniske underlag og å være kildekritisk ved søk etter informasjon.</td></tr>
<tr><td colspan="3">Elevens Læringsarbeid</td></tr>
<tr><td>Læreforutsetninger:</td><td colspan="2">• Kartlegging av klassen i forkant av undervisningsopplegget i form av en presentasjon av opplegget vi hadde planlagt for dem. Her hadde vi en samtale med kontaktlæren til klassen, som var behjelpelig når vi skulle sette sammen gruppene til gruppearbeidet.
• Noen av elevene hadde utfordringer med ADHD, og er avhengige av at gruppesammensetningene stemmer ut ifra deres utfordringer.
• Dette er en klasse som består av 15 elever. 1 jente og 14 gutter.
• 2 av elevene har dysleksi, men har ingen tilrettelegging på grunn av det (frivillig valgt).
• Spørreundersøkelse i forkant.</td></tr>
<tr><td>Rammebetingelser:</td><td colspan="2">• Arduino m/utstyr (komponenter, ledninger, og koplingsbrett)
• Legobil
• Undervisningslokale/klasserom
• Total tidsbruk: 4 skoletimer
• Vår egen kompetanse innen temaet det skal undervises i. Ettersom at undervisningen baserer seg på kompetanseheftet som vi har utviklet, føler vi selv at vi har</td></tr>
</table>

	god nok kunnskap for å undervise i disse timene. Utfordringen vil være hvis noen elever kommer med spørsmål utover det som et i kompetanseheftet.

Mål: Overordnet del (Kunnskapsdepartementet, 2017)	***Elektroniske kretser og nettverk, kompetansemål:*** • Bygge og programmere et selvvalgt produkt som består av mikrokontroller, analoge kretser, relevante sensorer og aktuatorer for å oppnå ønsket virkemåte. • Utforske sensorer knyttet til elektroniske kretser og nettverk og drøfte bruksområdet deres • Velge og bruke egnede instrumenter og programvare for å utføre målinger og feilsøking, og vurdere måleresultatet opp mot forventede verdier • 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 ***Grunnleggende ferdigheter:*** <u>Digitale ferdigheter</u> Bruke PC/IKT, bruke Arduino programmet <u>Lese ferdigheter</u> Lese oppgave og skrive oppgave/dokumentasjon <u>Muntlige ferdigheter</u> Diskutere faglige løsninger og vise forståelse. 2.1: Sosial læring og utvikling. Som lærere skal vi, støtte og bidra til elevenes sosiale læring og utvikling gjennom arbeid med fagene og i skolehverdagen for øvrig. 2.3: Grunnleggende ferdigheter. Skolen skal legge til rette for og støtte elevenes utvikling av de grunnleggende ferdighetene gjennom hele opplæringsløpet. 3.2: Undervisning og tilpasset opplæring.

	Skolen skal legge til rette for læring for alle elever og stimulere den enkeltes motivasjon, lærelyst og tro på egen mestring. **<u>Delmål:</u>** Elevene skal programmere en lysstyring til et valgt produkt. I denne sammenhengen er det valgte produktet en legobil.
Innhold	• Arduino, grunnleggende programmering, sosial utvikling gjennom gruppearbeid og dekning av læreplanmål ved bruk av egenprodusert kompetansehefte. • Innholdet i undervisningen er tilpasset elevenes læreforutsetninger og er en fin arbeidsoppgave for gruppearbeid • Kompetansehefte
Læreprosess: (Arbeidsmåter)	Induktiv læring: Elevene vil bli delt inn i grupper, og blir bedt om å løse en arbeidsoppgave ut ifra en arbeidsbeskrivelse. Det vil ikke bli gitt noen teorigjennomgang på forhånd, bare en gjennomgang av hva elevene skal arbeide med, og hensikten med oppgaven.

	Sosiokulturell læring: Ved å arbeide i grupper vil elevene oppleve å skape erfaringer gjennom refleksjon og samhandling. Fokuset i den sosiokulturelle læringsteorien er at arbeidet baserer seg på grupper, fremfor enkeltindivid. Teori og praksis: Samtaleundervisning. Arbeidsoppgaver/gruppearbeid. Praktisk arbeid med å innhente tekniske opplysninger for å kunne gjennomføre en programmering. Elevene skal til slutt reflektere. (Qr-kode ved undervisningsslutt.)
Vurdering:	• Vi vil vurdere elevenes læring og undervisning på bakgrunn av observasjoner av elevens deltagelse og aktiviteter underveis. • Denne undervisningsøkten baserer seg på gruppearbeid, og det skal ikke gis noen test/prøve på slutten av timen. Det eneste elevene trenger å gjøre i forhold til vurdering er at de skal svare på noen spørsmål etter endt undervisning. De vil få tilgang til spørsmålene via en QR-kode de vil få utdelt. Her skal de svare på spørsmål knyttet til undervisningsopplegget, arduino og kompetanseheftet. • Dette undervisningsopplegget er utarbeidet i forbindelse med vår aksjonering til FoU-arbeidet.

Relevans for yrkene:	• Arduino er en enkel inngang til programmering som de alle fleste vil møte i fremtidig yrke innen elektro. • En kan begrunne dette med utviklingen som skjer innenfor elektrofagene, de aller fleste har fagene og yrkene et økt fokus rundt programmering.

Vedlegg 2 – Kompetansehefte

SAMMENDRAG

Dette kompetanseheftet er ment for deg som er elev og lærer på Vg1 Elektro- og datateknologi, og Vg1 Teknologi- og industrifag.

Kompetanseheftet inneholder litteratur, øvingsoppgaver, tverrfaglige- og fordypningsoppgaver med løsningsforslag.

Helt i starten av heftet vil du finne kompetansemål fra Vg1 læreplanen som vi kan dekke ved å bruke Arduino. Deretter kommer pensum og litteratur, og sist i heftet finner du oppgaver med forskjellige vanskelighetsgrader.

Innhold

1.0 Innledning

Dette kompetanseheftet er ment for deg som går yrkesfagretningene Vg1 elektro- og datateknologi, eller Vg1 teknologi- og industrifag. Det er tatt utgangspunkt i kompetansemål fra læreplanene og pensum ut ifra lærebøkene som blir benyttet innenfor de forskjellige yrkesretningene.

Kompetanseheftet skal være med på å gi eleven en startpakke for hva de trenger av utstyr for å komme igjennom pensum og kompetansemålene for Vg1. Det blir gitt en innføring i hva Arduino er, og alt fra lette til mer avanserte arbeidsoppdrag som kan gjennomføres for å få øving i programmering.

Det er lagt opp til arbeidsoppgaver som er relevante for TIP elever, og arbeidsoppgaver som er relevante for EL elever. I og med at disse to yrkesretningene blir mer og mer lik har vi laget et arbeidsoppdrag som er tverrfaglig.

1.1 Kompetansemål fra læreplan til Vg1 elektro og datateknologi

Kompetansemål fra læreplanen som kan knyttes til mikrokontrolleren (Kunnskapsdepartementet, 2020a):

- Bygge og programmere et selvvalgt produkt som består av mikrokontroller, analoge kretser, relevante sensorer og aktuatorer for å oppnå ønsket virkemåte
- Utforske sensorer knyttet til elektroniske kretser og nettverk og drøfte bruksområdet deres
- Velge og bruke egnede instrumenter og programvare for å utføre målinger og feilsøking, og vurdere måleresultatet opp mot forventede verdier
- Styring av servo
- LCD-skjerm
- Voltmeter
- Temperaturmåling
- Fuktighetsmåling

1.2 Kompetansemål fra læreplan til Vg1 teknologi- og industrifag

Kompetansemål fra læreplanen som kan knyttes til mikrokontrolleren (Kunnskapsdepartementet, 2020b):

- Utvikle og beskrive skjemaer med grunnleggende komponenter som brukes i elektriske, hydrauliske og pneumatiske anlegg
- Utføre og bruke grunnleggende beregninger av relevant størrelser innenfor elektriske, hydrauliske og pneumatiske anlegg
- Bruke relevante simuleringsprogrammer for dokumentasjon, design og testing av styringssystemer innenfor elektro, hydraulikk, pneumatikk og kjemiske prosesser
- Koble opp, teste og feilsøke på automatiserte og manuelle styringssystemer basert på elektroniske, hydrauliske og pneumatiske anlegg og maskiner og utføre målinger som er relevante for fagområdet
- Anvende grunnleggende programmering av styringssystemer innenfor robotisering, automatisering og CNC

2.0 Generell introduksjon til Arduino og teori

2.1 Hva er Arduino?

Kildekode-elektronikkplattformen Arduino er basert på brukervennlige maskinvarer og programvarer. Et Arduino-kort er i stand til å lese innganger, lys på en sensor, en finger på en knapp eller en melding på twitter, gjøre det om til en utgang, aktivere en motor, slå på en LED eller publisere noe. Ved å sende et sett med instruksjoner til mikrokontrolleren på brettet kan du fortelle hva du vil skal skje. Det benyttes et Arduino programmeringsspråk som er basert på Wiring, og Arduino Software (IDE) som er basert på Processing.

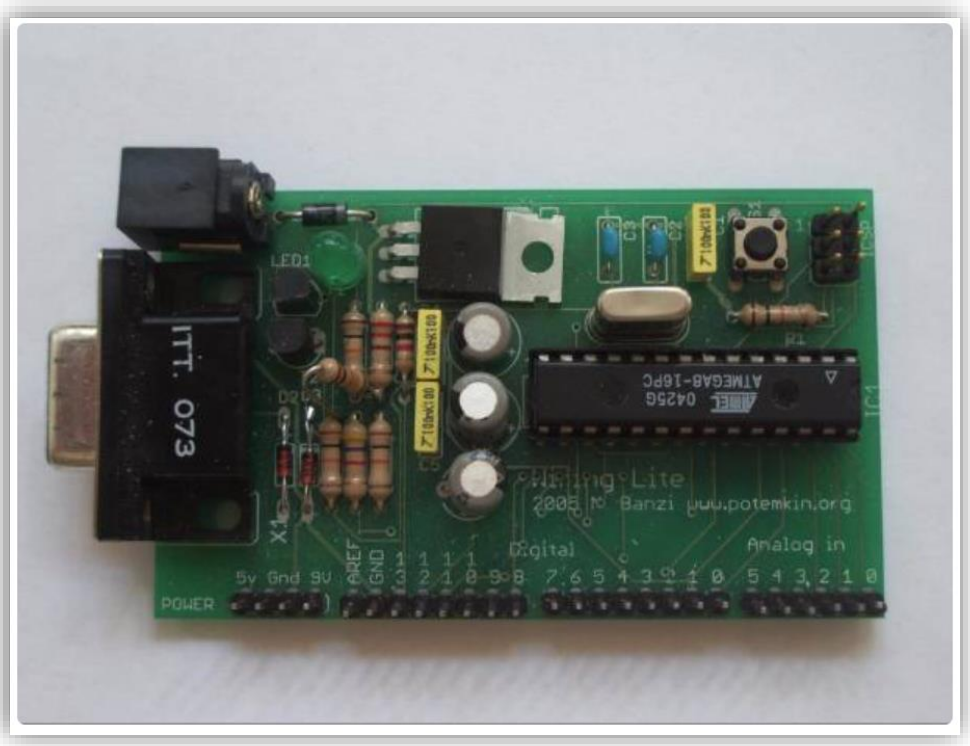
Arduino bruker programmeringsspråket C++ som er et forbedret programmeringsspråk videreutviklet fra språk C. Det ble utviklet for å skape et språk som hadde funksjonaliteten for høynivåspråk og hastigheten til lavnivåspråk (Almås & Rossen, 2021).

I flere år har Arduino hvert hjernen på flere tusen prosjekter. Alt fra hverdagslige gjenstander til komplekse vitenskapelige instrumenter.

2.1.1 Historien rundt Arduino

Vi skal tilbake til Ivera i Italia i 2005, hvor det det hele startet som et prosjekt. Målet var å lage en anordning for å kontrollere prototypene til interaksjonsdesignstudenter på en rimelig måte (Wikipedia, 2022).

Dette var et prosjekt som var bygget på Wiring-plattformen, som bygger på Processing og IDE-grensesnittet. Massimo Banzi, David Mellis (IDII-student) og David Curtielles startet et prosjekt hvor de kopierte Wiring koden, og startet et eget prosjekt som de kalte for Arduino. Arduino og Wiring hadde mye til felles av utviklingen som Nicholas Zambetti hadde gjort, og i en kort periode var også Zambetti ansett som et medlem av Arduino-teamet. Rundt samme tidsperiode ble også Gianluca Martino en del av Arduino-teamet, og hans oppgave var å hjelpe til med produksjonen og maskinvareutviklingen. Han utviklet sammen med David Cuartielles en billigere maskinvare, for å redusere kostnadene på brettene deres, ved å bruke Atmega8 (Barragàn, 2016).



Figur 20 Den første Arduino prototypen - Wiring Lite. https://arduinohistory.github.io/

2.2 Hvorfor Arduino?

Arduino har blitt brukt til tusenvis av prosjekter og applikasjoner, på grunn av sin enkle og tilgjengelige brukeropplevelse. Programvaren for Arduino er enkel å bruke for nybegynnere, men er likevel fleksible nok for avanserte brukere. Arduino kan brukes til å bygge alt fra vitenskapelige instrumenter, eller til å komme i gang med programmering og robotikk. Arduino er et nøkkelverktøy for å lære nye ting.

Det finnes flere forskjellige mikrokontrollere og mikrokontrollerplattformer tilgjengelig for fysisk databehandling. Ved å benytte seg av Arduino forenkler det prosessen med å jobbe med mikrokontrollere, i forhold til andre systemer på grunn av:

- Rimelig → Et Arduino kort er rimelig sammenlignet med andre mikrokontrollerplattformer.
- Cross-plattform → Arduino Software (IDE) kan brukes på flere operativsystemer.
- Enkelt, oversiktlig programmeringsmiljø → Programvaren til Arduino er enkel å bruke for nybegynnere, og fleksibel for avanserte brukere.

- Åpen kildekode og utvidbar programvare → Programvaren til Arduino er en åpen kilekodeverktøy, som er tilgjengelig for utvidelse av erfarne programmere. Språket utvides gjennom C++ biblioteket.

2.3 Grunnleggende om strukturen

2.3.1 Funksjoner og koding i Arduino

2.3.2 Strukturen til Arduinoprogrammet

```
1 //Her deklareres globale variabler og konstanter:
2
3
4 void setup() {
5   //Setup-koden kjører bare én gang, og den skrives her:
6   /* Her utføres alle operasjoner og innstillinger, som må være gjort før hovedprogrammet
7   kjøres */
8 }
9
10 void loop() {
11  //Her skrives hovedprogrammet, og kjøres etter void setup
12  /* Koden som skrives inne i void loop kjører om og om igjen,
13     helt til shut-down av mikrokontrolleren */
14 }
```

2.3.2 Setningsblokker

En setningsblokk eller en kodeblokk, består av en eller flere programsetninger gruppert sammen. Kompilatoren betrakter dem som sammenlenket, og de må starte og slutte med krøllparentes: { }.

Eksempel:

```
if(sensorValue < 500) {          //Dersom if-setningen slår til, utføres hele setningsblokken
                                 //mellom { og }.
digitalWrite(redLed, HIGH);  //Etter hver programlinje er det viktig å huske semikolon ( ; ),
                                 //som indikerer linjeslutt.
delay(500);                      //er ventetid eller delaytid før/etter operasjon
digitalWrite(redLed, LOW); // indikerer at LED er skrudd av (LOW/ingen strøm)
}
```

2.3.3 Variabler og konstanter

Det er viktig å huske at variabler og konstanter må deklareres/opprettes før vi kan bruke dem i programmet.

En variabel er et navn som tar vare på informasjon/verdier som behandles og endres i løpet av programkjøringen. Vi gir variabler verdier ved tilordning, ved å bruke tilordningsoperatoren. F.eks.: variabel = uttrykk. Her har vi først navnet på variabelen, og uttrykket er verdien på variabelen.

Eksempel:

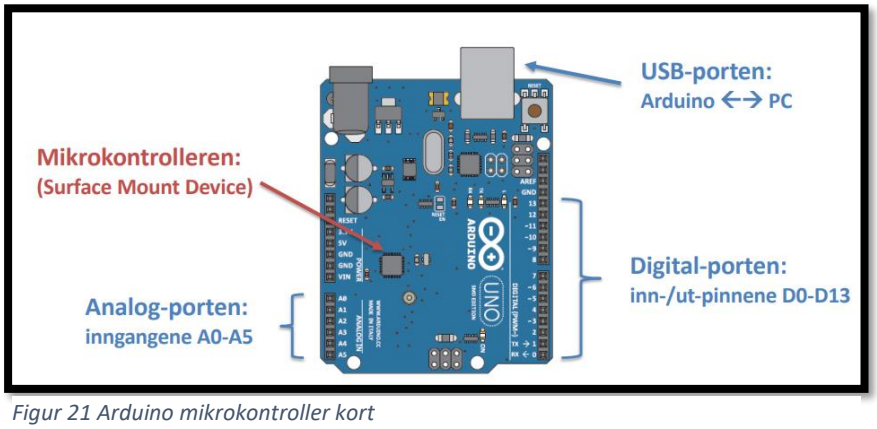
sensorValue = 500;

En konstant er et navn som tar vare på informasjon/verdier som ligger fast i løpet av programkjøringen. F.eks.: sensorPin = A2;

Variabler og konstanter kan ta forskjellige datatyper

- Heltall: int
- Desimaltall: float
- Tekststreng: String
- Bokstav/tegn: char
- Boolsk/sannhetsverdi: boolean

2.3.4 Arduino Uno



Figur 21 Arduino mikrokontroller kort

2.3.5 Viktige begreper i koding

- Programbyggeklosser

- Variabler og konstanter (deklarere, initialisere)
- Instruksjoner
- Valgstrukturer (if-else, switch case)
- Løkkestrukturer (while- og for-løkker)
- Betingelser (if-else kjøres)
- Funksjoner

2.3.6 I/O-instruksjoner

- Digitale

digitalWrite(pinne, verdi)

digitalRead(pinne)

pinMode(pinne, modus) (modus = INPUT eller OUTPUT)

- Analoge

analogRead(pinne)

analogWrite(pinne, verdi)

2.3.7 Navnekonvensjon

Når vi skal skrive koder er det viktig at de er tydelige og lette og lese. Selv om vi skriver koden kanskje bare en gang, skal den kunne leses mange ganger, og det finnes mange måter å skrive en kode på, så derfor skal vi her vise to eksempler på hvordan en kode kan skrives.

camelCase

Skriver vi kodene ved å benytte camelCase, deler vi ord med stor bokstav. Da kan vi skrive for eksempel: Int minVariabel;

Vi kan også dele ord med _. Da kan vi skrive for eksempel: Int min_variabel;

Det er også grei programmeringsskikk at vi skriver en forklarende tekst i starten av programmet. Det skriver vi etter «/*» og før */.

For eksempel:

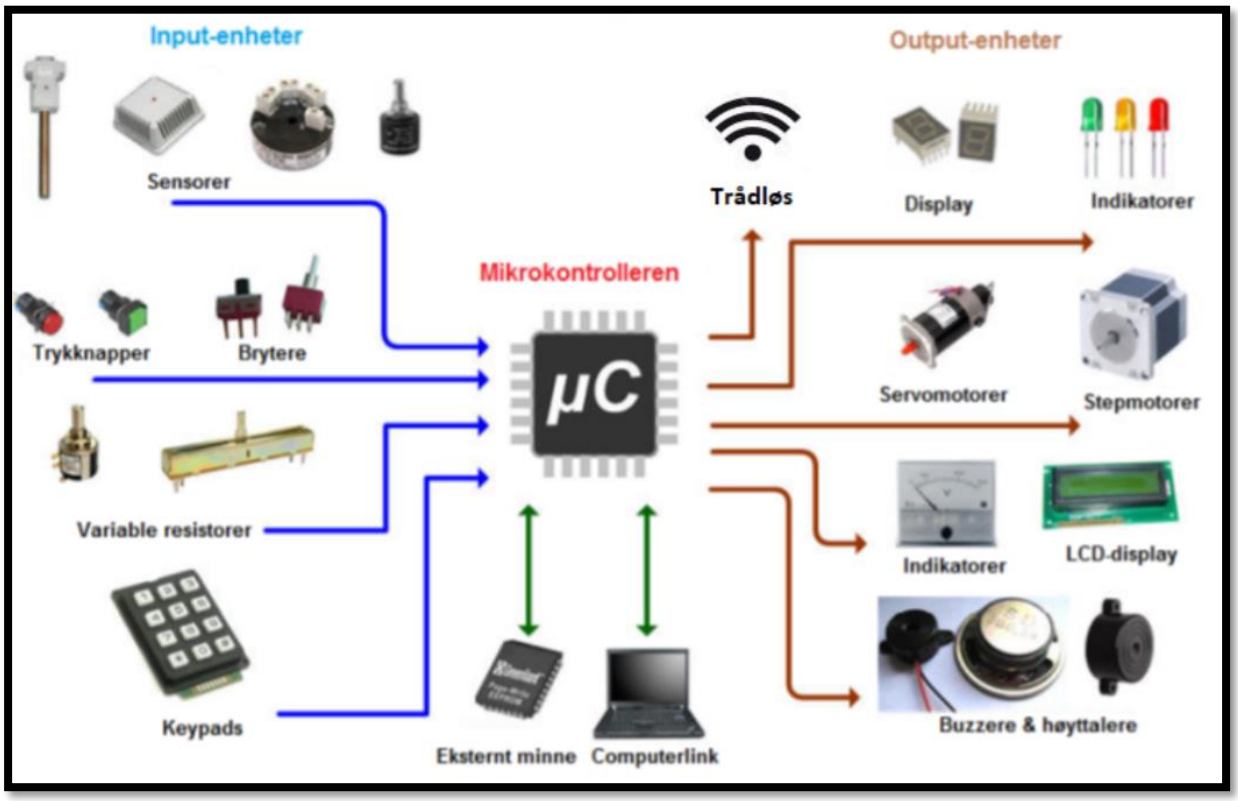
```
/*
Filnavn:    Mitt første prosjekt
```

Versjon: 1

Program: Styring av LED-diode

*/

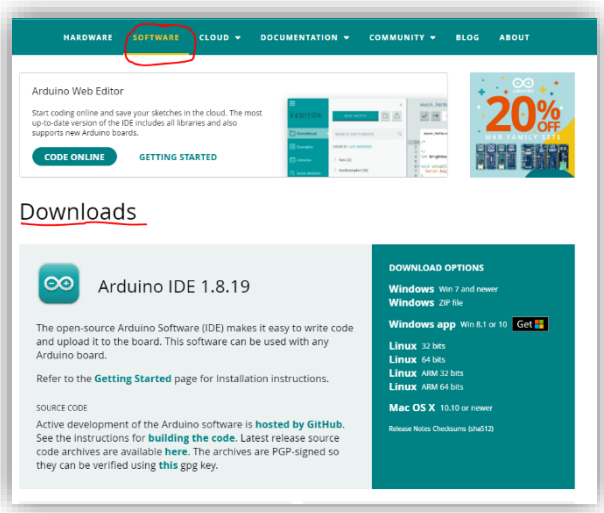
2.3.8 Nærmiljøet til en mikrokontroller



Figur 22 Bilder over komponenter i nærmiljøet til en mikrokontroller

2.3.9 Arduino Software (IDE), guide

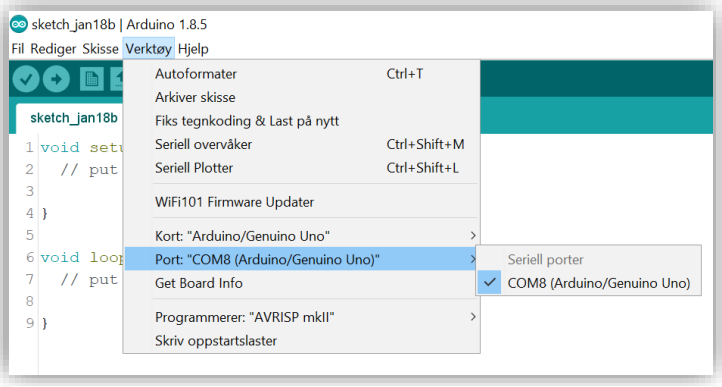
1. Gå inn på nettsiden arduino.cc
2. Øverst i menyen velger du software
3. Når du klikker deg inn på software, kan du laste ned Arduino IDE



Figur 23 nettsiden Arduino.cc

Når du har lastet ned programmet må det klargjøres. Det gjøres på denne måten:

1. Åpne Arduino-programmet du lastet ned
2. Koble Arduinoen til datamaskinen
3. Gå til verktøy og se at Arduinoen er satt til riktig port

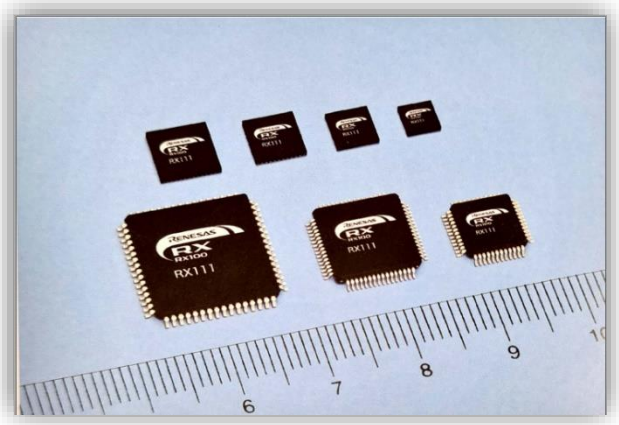


Figur 24 Programmet åpnet på datamaskinen

2.4 Mikrokontrolleren

Mikrokontrolleren er som en liten PC, som vi programmerer til å utføre det vi trenger den til. Mikrokontrolleren har en liten hjerne som vi kaller mikroprosessoren, et minne og I/O-enheter. Mikrokontrollere finnes i alle slags prisklasser og størrelser, og de kan utføre mange ulike oppgaver når vi programmerer dem. Nettopp på grunn av dette kan vi finne en mikrokontroller nesten overalt. Bare i hjemmet ditt vil vi kunne finne mange produkter som har en eller flere mikrokontrollere innebygd. Mobiltelefoner, biler, panelovner, klokker, datamaskiner, TV-apparater, termostater og alarmer for å nevne noen.

Det finnes mange forskjellige typer mikrokontrollere å velge imellom. Noen har mange pinner (bein), og andre har få. Avhengig av bruksområdet, velger du hvor mange innganger og utganger du trenger.



Figur 25 Forskjellige mikrokontrollere

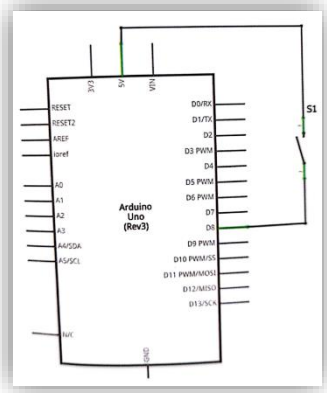
Igjennom arbeidet på skolen vil dere bli introdusert for Arduino, som dere skal programmere. Arduino er en elektronikkplattform og ikke et navn på en mikrokontroller (Venås & Vangsnes, 2020, s. 60).

2.5 Digitale innganger

D0-D13 er de digitale inngangene, og vi kan også bruke A0-A6 som analoge innganger og utganger. I koden vi lager bestemmer vi det. Hvis ikke noe annet blir spesifisert, er alle

pinnene i utgangspunktet innganger. Vi sier at en pinne er i høyimpedansmodus hvis den er satt til å være en inngang. Inngangen vil ha en impedans i størrelsesorden 100 MΩ.

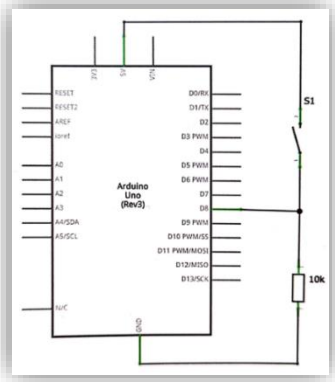
I figuren til høyre (figur 6) har vi en bryter som er koblet til 5V til digital pinne nummer 8 (D8). Her er D8 satt til å være inngangen. Når vi da trykker på bryteren, får vi 5V.



Figur 26 Arduino med bryter

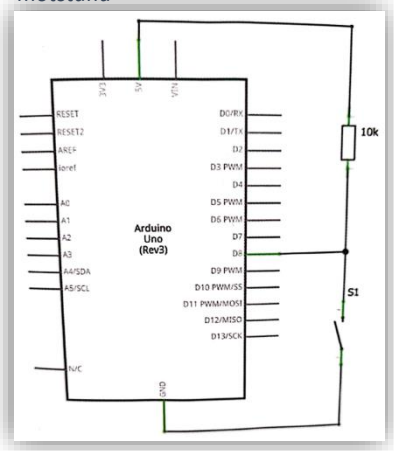
I og med at inngangen har en veldig høy impedans, vil spenningen på inngangen variere, og i praksis være ustabil når brytere ikke er trykt ned. I og med at pinnene har en veldig høy impedans, vil de være veldig følsomme for ytre påvirkninger. Dette vil føre til at pinnene oppfører seg som en antenne og kan plukke opp støy.

I bilde til høyre ser vi en krets, hvor det er blitt satt inn en motstand fra bryteren og ned til GND. Når vi trykker på bryteren nå, vil bryteren legge D8 til 5V, og blir det vi kaller aktivt høy. Ettersom at det ikke går strøm i inngangen, vil vi få noe spenningsfall over motstanden. Det vil si, vi måler 0V på begge sider og dermed 0V på inngangen, når bryteren ikke er trykket på. En krets som er koblet med en slik motstand, kaller vi for en pull-down-motstand.



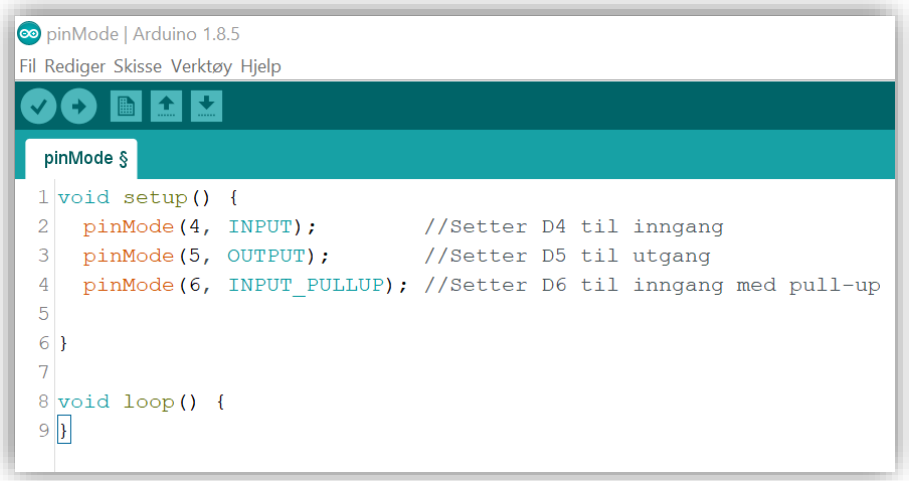
Figur 27 Bryter og pull-down-motstand

I bilde til høyre har vi nå gjort det motsatt. Her har vi koblet med en pull-up-motstand, og det vil si at ettersom det ikke går strøm i inngangen, blir spenningsfallet over motstanden 0V. Dermed har vi 5V på begge sider og inngangen vår blir lagt til 5V. Størrelsen som er anbefalt på slike motstander er 10kΩ. Vi kan slippe å bruke en ekstern motstand dersom vi velger å bruke den innebygde pull-up-motstanden som mikrokontrolleren har innebygd. Den kan vi aktivere med en kode. Velger vi å benytte oss av den innebygde pull-up-motstanden må vi huske at inngangen får 0V, blir aktivt lav, når vi trykker på bryteren.



Figur 28 Bryter med pull-up-motstand

Det er viktig at vi bestemmer om en pinne skal være inngang eller utgang. Det løser vi med å bruke funksjonen pinMode () (Venås & Vangsnes, 2020, s. 65).



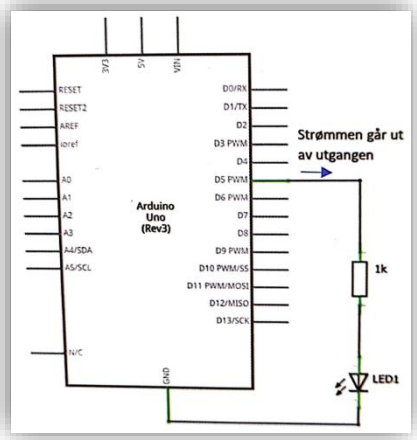
Figur 29 Hvordan bruke funksjonen

2.6 Digitale utganger

Når vi setter en pinne til å være utgang, vil det si at den er i en lav-impedansmodus, og skal levere strøm til andre kretser (Venås & Vangsnes, 2020, s. 68).

<u>Sourcing current</u>

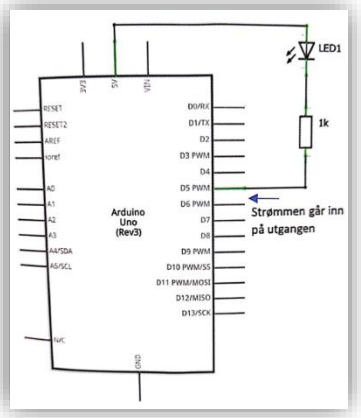
På skjemaet til høyre har vi en motstand og en lysdiode som er koblet fra D5 til GND. For at lysdioden skal lyse må D5 settes til 5V, ettersom at strømmen går fra + til -. Det vil si, strømmen går fra D5 gjennom motstanden og lysdioden før den kommer til GND. Navnet soucing current kommer av at strømmen går ut av utgangen.



Figur 30 Sourcing current

<u>Sinking current</u>

I sinking current kobles 5V inn på lysdioden og derfra går det videre til motstanden. For at lysdioden skal lyse må vi sette utgangen D5 til 0V, og da vil strømmen gå inn i utgangen på grunn av at strømmen går fra + til -.



Figur 31 Sinking current

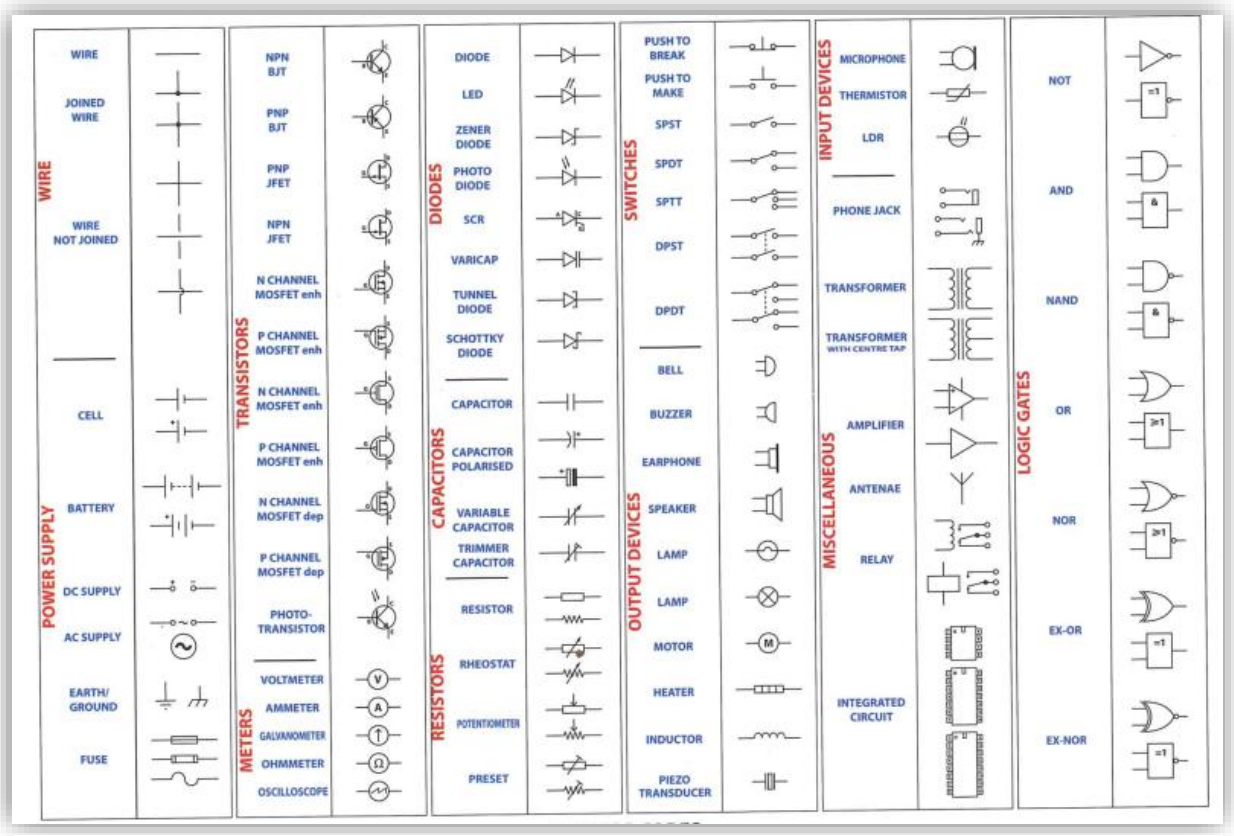
For å skrive til en digital utgang bruker vi en funksjon som heter digitalWrite(). For eksempel:

digitalWrite(5, HIGH);

digitalWrite(5, LOW);

2.7 Skjemategning

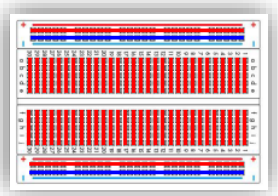
Innenfor elektronikk verden er det å utarbeide gode koblingsskjemaer viktig. Dette på grunn av at et slikt koblingsskjema er en del av dokumentasjonen av hvilke komponenter som er brukt, hvordan de er koblet sammen og det hjelper oss å holde kontroll på ulike versjoner av kretsen. Denne dokumentasjonen kan vi også bruke opp imot andre som skal arbeide med kretsen, underlag for å lage kretskort, eller planlegging når det skal gjøres innkjøp (Venås & Vangsnes, 2020, s. 72).



Figur 32 Symboler. Hentet fra boken elektroniske kretser og nettverk

2.8 Komponenter

Koblingsbrett og Arduino Uno



Figur 33 Koblingsbrett https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

Figur 34 Arduino UNO kort https://www.elfadistrelec.no/en/microcontroller-board-uno-arduino-a000066/p/11038919?ext_cid=shgooaqnoen-Shopping-campaign&gclid=CjwKCAiApfeQBhAUEiwA7K_UHwjGqcRmgn4TqUp6OJuY8U-gNasInXID3L6qUWvFA5JfwZCQi7au7hoCvBgQAvD_BwE

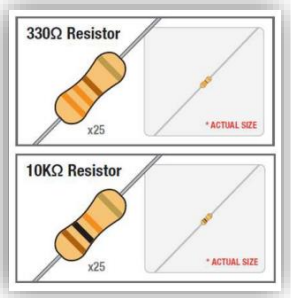
Bilde av koblingsbrettet viser hvilke koblingspunkter som er koblet sammen på undersiden. Vi bruker jumper wire 'e for å forbinde komponentene på koblingsbrettet med Arduino-en.

Figur 35 Jumper wire https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

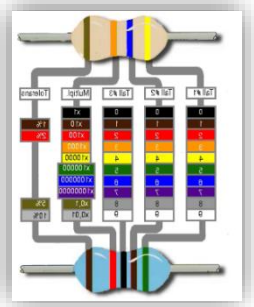
På sidene av koblingsbrettet har vi to langsgående sammenhengende hullrader. Disse er ment for å føre + og -. Den røde langsgående linjen er for pluss, og den blå langsgående linjen er for minus. Dette er for å indikere at alle kontaktpunktene langs en linje er koblet elektrisk sammen (Rossing & Stausland, 2021, s. 22).

Motstander

Motstander finnes i mange ulike verdier, og det er fargekoden på motstanden som bestemmer hvilken verdi motstanden har. Det er viktig å velge riktig verdi ut ifra hva du skal bruke det til.



Figur 20 Motstand https://www.ntnu.no/documents/2004699/12108297/Nordic+ESER



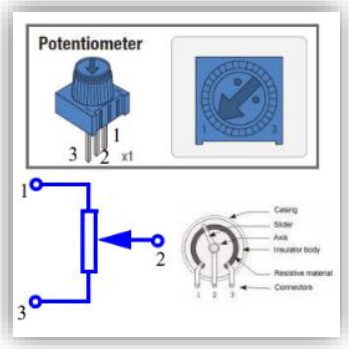
Figur 21 Motstandsverdier https://www.tek.no/nyheter/guide/i/MRpbRr/elektronikkens-verden-del-1?page=3

Grunnen til at motstander brukes er at en i enkelte tilfeller ønsker å begrense strømmen i en komponent, eller for å etablere et spenningsnivå, som oppnås med en spenningsdeler. Strømmen i en motstand vil ifølge ohms lov øke proporsjonalt med spenningen over den, fordi det for eksempel er ønskelig å omdanne en

varierende mostandsverdi hos mange sensorer for å få en varierende spenning. Hvilken vei motstanden plasseres har ingen betydning for dens virkemåte.

Det er fargene på ringene til mostanden som bestemmer verdien den har, og hver farge står for et av sifrene 0 til 9 (Rossing & Stausland, 2021, s. 23).

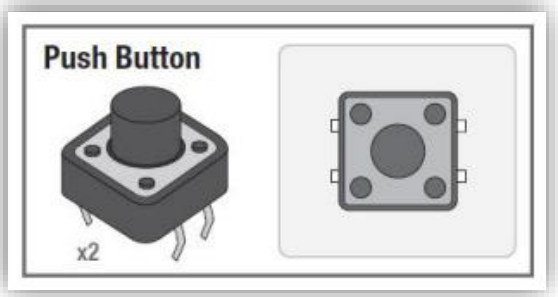
Potensiometer



Figur 22 Potensiometer. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

Et potensiometer er en motstand, men i motsetning til beskrivelsen av mostander i avsnittet ovenfor, er dette en motstand med et variabelt uttak. Potensiometer benyttes der hvor vi ønsker å etablere en variabel spenning mellom spenningen på ytterpunktene, eller bare ønsker at den skal fungere som en volumkontroll som f.eks. for et signal som sendes inn mellom pinne 1 og 3, og som tas ut mellom 2 og 3 (Rossing & Stausland, 2021, s. 23-24).

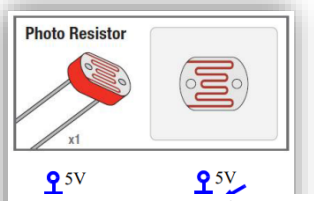
Bryter



Figur 23 Bryter. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

Bryteren som følger med arduino-en har fire bein, som er forbundet med hverandre to og to. Trykker man på knappen vil de to parene bli koblet sammen, og så lenge knappen er trykket inn opprettholdes forbindelsen helt til den slippes og brytes. Vi bruker bryterne til å gi kretsen en enkel informasjon om av og på, akkurat som en lysbryter. Trykket på bryteren må omdannes til en spenning for at kretsen skal forstå informasjonen (Rossing & Stausland, 2021, s. 24).

Fotomotstand

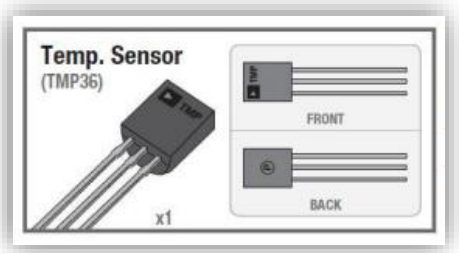


Figur 24 Fotomotstand. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

Fotomotstanden er rett og slett en motstand der hvor verdien blir bestemt av intensiteten på lyset som treffer den. Vi vil oppleve å få lavere motstandsverdi, desto mer den belyses. Sterk lys vil normalt gi 100 ohm, mens mørkt lys gir typisk 300 Kohm. Fotomotstanden består av to bein, og det har ingen betydning hvilken vei den kobles inn i kretsen.

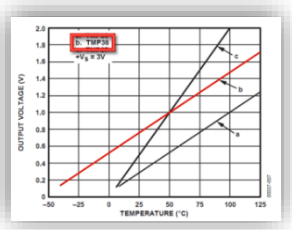
Ønsker man å styre en funksjon ved hjelp av lysstyrken, benyttes gjerne en fotomotstand. Det kan f.eks. være tenning av lys når det blir mørkt ute. Fotomotstanden kobles som oftest i en spenningsdeler, og plasseringen bestemmes av funksjonen (Rossing & Stausland, 2021, s. 25).

Temperatursensor



Figur 25 Temp. sensor. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

Temperatursensoren (TMP36) registrerer temperatur og gir ut en spenning, som er proporsjonal med temperaturen. Denne typen sensorer brukes gjerne i elektroniske termometre eller i termostater for å regulere en varmeovn. Vi kan også bruke temperatursensor for å beskytte elektronikk, og det gjøres ved at strømmen brytes når temperaturen overskrider et maksimalt nivå. Spenningen øker med 10 mV per grad. Ved 0 ℃ vil spenningen være 0,5 V.



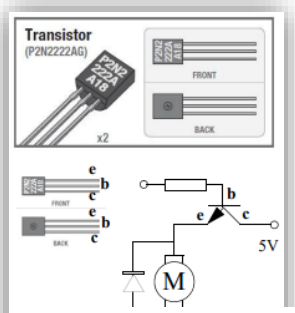
Figur 26 Diagram av spenning som funksjon. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

I bilde til høyre ser vi et diagram som viser spenningen som funksjon av temperaturen. Temperatursensoren (TMP36) er den røde kurven (Rossing & Stausland, 2021, s. 25).

Transistor

En transistor har som regel to bruksområder:

- Forsterker når strømmen/basespenningen varierer innen et lite område
- Bryter når strømmen/basespenningen er så stor at den slår av eller på transistoren



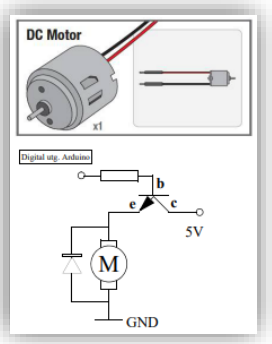
Figur 27 Transistor. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

I elektroniske forsterkere, radiosendere og mottakere, TV-er o.l. brukes transistoren som en signalforsterker. I styringssystemer og datamaskiner brukes den oftest som bryter. På bilde til høyre ser vi at en transistor har tre bein, eller terminaler som det heter. Bein «c» kalles for collector, bein «e» kalles for emitter og bein «b» for basen. Det kan gå en relativ

stor strøm fra collector beinet til emitter beinet, og størrelsen på denne collectorstrømmen kan styres av spenningen imellom basen og emitter, og når den blir stor nok kan det gå strøm i transistoren (Rossing & Stausland, 2021, s. 27).

Motor

Når en motor får tilført en spenning på over 3 V vil akslingen begynne å rotere. Arduinoer klarer ikke å styre store strømmer helt opp til 200 mA, derfor kan vi benytte en transistorbyter som vil tåle den store strømmen.



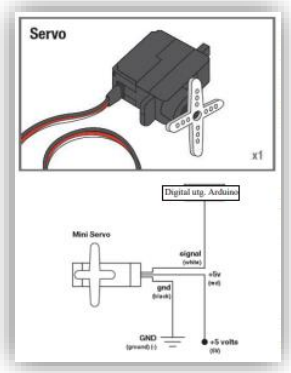
Figur 28 Motor. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

I dag er bruksområdet for motorer veldig kjent. Vi finner de igjen hvor det er noe som enten skal gå rundt eller bevege seg. Det kan være alt fra leketøy, datadisker til vifter, pumper, elektriske biler etc.

I bilde til høyre ser vi en krets hvor det er koblet inn en «fly back» diode over motoren. Den har som oppgave å hindre overspenninger som kan oppstå på transistoren idet strømmen brytes for å stoppe motoren (Rossing & Stausland, 2021, s. 27).

Servo

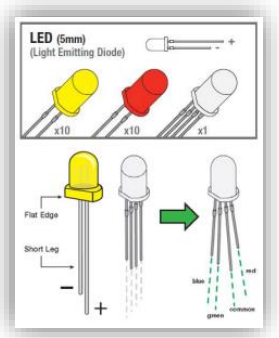
En annen motor vi blir kjent med igjennom arduino er servo motoren. Den kan dreie akslingen en bestemt vinkel eller rotere med en definert fart. På kommando fra mikrokontrolleren kan servo motoren dreie akslingen i en vinkel fra 0 – 180 º. Servoen mottar pulser, og det er lengden på en puls som bestemmer dreievinkelen. Har vi en pulslengde på 1,5 ms gir det oss en vinkel på 90 º. Disse pulsene må gjentas med jevne mellomrom, akkurat som man pulsbreddemodulerer et signal. Vi kan koble servo motoren til en utgang som har en slik utgang (pmw), f.eks. analog port nummer 9.



Figur 29 Servo. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

Servo motorene finner vi som oftest i modellfly, hvor det ønskes å styre side-, høyderor og flaps. Dette er også en komponent som er viktig i mange robot styringer, hvor målet kan være å bevege for eksempel en arm (Rossing & Stausland, 2021, s. 28).

Lysdioder



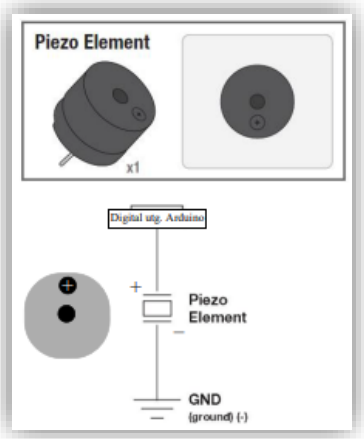
Figur 30 Lys dioder. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

Lysdioder leder strøm bare en vei, og det er fra anoden til katoden. Det vil si, for at den skal lyse er vi avhengige av å koble den riktig vei. Lysdioden begynner å lyse svakt når strømmen i den overstiger ca. 1,5 – 2 mA. Ettersom strømmen øker, øker også lysstyrken på dioden. Lysdioden kobles gjerne i serie med en motstand som har en verdi på 220 – 330 ohm, dette for å begrense strømmen, fordi uten denne seriemotstanden er det stor sannsynlighet for at dioden går i stykker.

Det finnes også lysdioder som har navnet RGB, og består av en rød-, en grønn og en blå lysdiode som er montert i samme kapsel.

Lysdioder eller LED, som står for light emitting diode, brukes til blant annet signallamper, displayer og i den senere tid til belysning (Rossing & Stausland, 2021, s. 28).

Buzzer



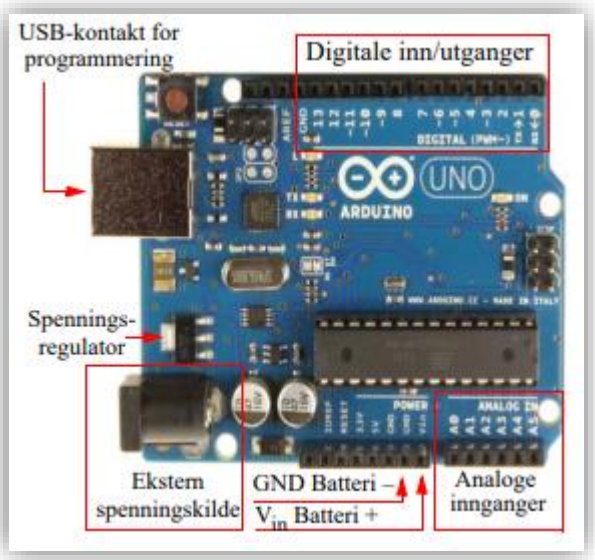
Figur 31 Piezo buzzer. https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-2aac474ac43e?t=1598292146218

En Buzzer, eller et piezo-elektrisk element, består av et krystall som trekker seg sammen når det påføres en spenning og et klikk kan høres. Når denne spenningen forsvinner, vil krystallet i buzzeren få tilbake sin opprinnelige form og et klikk vil høres. Ved å la spenningen variere hurtig høres lyd som i en høyttaler. For at buzzeren skal fungere korrekt må pluss og minus være riktig tilkoplet.

For å lage lyder og toner med forskjellig frekvens bruker man et piezo-elektrisk element. Buzzeren trenger bare en spenning for å gi en tone, på grunn av at buzzeren består av t piezo-element og en enhet som lager en varierende spenning (Rossing & Stausland, 2021, s. 29).

2.9 Mikrokontrollerkortet Arduino UNO

Mikrokontrollerkortet er det vi kaller for databehandlingsenheten, hvor vi finner inn- og utganger for sensorer og aktuatorer. I tillegg til inn- og utganger har den som nevnt i avsnitt 2.1.4 rammeverk, en intern timer og et internt lager for program og data. Kortet har også en USB-inngang, hvor det er mulig for oss å legge inn programvare og en plugg for batteritilkobling eller en batteriadapter.



Figur 37 https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-

Ett av de mest populære mikrokontrollerkortene innen arduino familien er UNO R3. Dette er et kort om er bygget opp på prinsipper fra Atmel mikrokontrolleren Atmega328P. Den har en klokkefrekvens på 16 MHz, og et flash lager i størrelsen 32 kbyte, SRAM 2 kbyte og en EEPROM 1 kbyte.

Mikrokontrollerkortet består av følgende inn- og utganger:

- **Digitale I/O-porter**

  Vi har 14 digitale inn- og ut porter på mikrokontrolleren. Dette er porter som kan programmeres til å enten være en inn- eller utgang. Port 3, 5, 6, 9, 10 og 11 er porter som kan pulsbreddemoduleres, såkalt pwm signal. Disse portene er merket med et sinus symbol (~) på kortet. Vi kan kun ha maksimalt 40 mA på hver av I/O portene.

- **Analoge innganger**

  Vi har 6 analoge innganger på kortet, som kan tilføres en spenning fra 0 – 5V. Tilfører vi spenninger høyere enn 5V vil det ødelegge mikrokontrolleren.

- **USB-kontakt**

  Mikrokontrollerkortet er også utstyrt med en US-kontakt, som brukes for direkte tilkobling av PC og for programmering av kortet og overføringer. Vi kan også overføre data mellom PC-ens monitor og kortet. Det tilføres en spenning fra USB-kontakten under programmeringen, og hvis denne belastes

med mer enn 500 mA vil strømforsyningen brytes inntil strømtrekket er redusert til under denne grensen. Bruker man en USB3 vil denne kunne levere en vesentlig høyere strøm.

- **Strømtilførsel**

  Anbefalt spenning for batterieliminator er på 7 – 12V, og grenseverdiene er mellom 6 – 20V. Man kan enten tilkoble batteriet via eliminatorpluggene, eller via V_{in} og GND. Datainnsamlingskortet får spenningen sin fra 5V utgangen, og enkelte komponenter må ha en lavere spenning som leveres fra 3.3V utgangen.

- **Reset**

  Mikrokontrollerkortet inneholder en RESET-knapp, som vi kan bruke til å resette programmet for så å starte det på nytt. Det er også montert kantkontakter som er for å montere til tilleggskort, shield card, rett på arduino-kortet.

- **Rx/Tx**

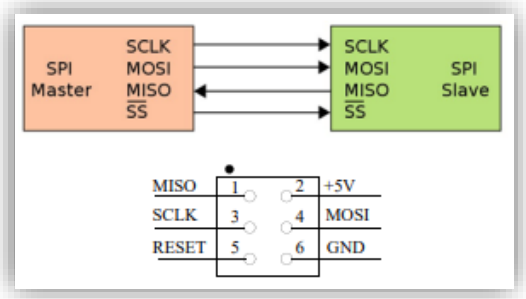
  Via portene Rx og Tx (I/O-port 0 og 1) støtter kortet UART, altså seriell datakommunikasjon. Vi benytter også disse portene for programmering av kortet. Rx og Tx er tilkoblet to lysdioder som blinker når kortet kommuniserer med USB/PC, og tilsvarende vil skje når vi overfører data til programeditoren for monitorering av data på PC-skjermen.

- **I^2C-databus**

  I^2C er en Inter IC-bus, og dette er en BUS som er svært enkelt å bruke med sine to linjer som består av klokke og datalinje. Hver krets har sin egen unike adresse som enten er satt av fabrikken, eller kan settes med strap'er på brikken. BUS'en er også utstyr med kollisjonsdeteksjon, som i starten var definert med en hastighet på 100 kb/s, men ettersom at vi stadig trenger raskere dataoverføring har vi nå Fast mode som er 40 kb/s og High speed som er på 3,4 Mb/s.

- **SPI-databus**

  SPI står for Seriel Peripheral Interface, og er en flerlinjers databuss som brukes for å overføre data til utstyr nedenfor mikroprosessoren, også kalt periferutstyr. Her fungerer den ene modulen som en master, sjef, og den andre modulen som slave, tjener. Master modulens oppgave er å ta initiativ til dataoverføring av SS-signallinjen, for så å sende sine data på en linje som heter MOSI, Master Output Slave Input. Slave modulens oppgave er å motta data på sin MOSI linje. Når data sendes mottar masteren svar på sin forespørsel til slaven som sender data tilbake på linjen MISO, Master Input Slave Output, som mottas av master på linjen MISO. Klokkesignalet har beskrivelsen SCLK og det er den som bestemmer takten til dataoverføringen. Slave Select signalet, SS-signalet, har som oppgave å varsle slaven om at master modulen ønsker kontakt. Er det flere slaver i et system trenger vi flere SS-linjer, en til hver slave. Figuren viser masteren på venstre side og slaven på høyre side.



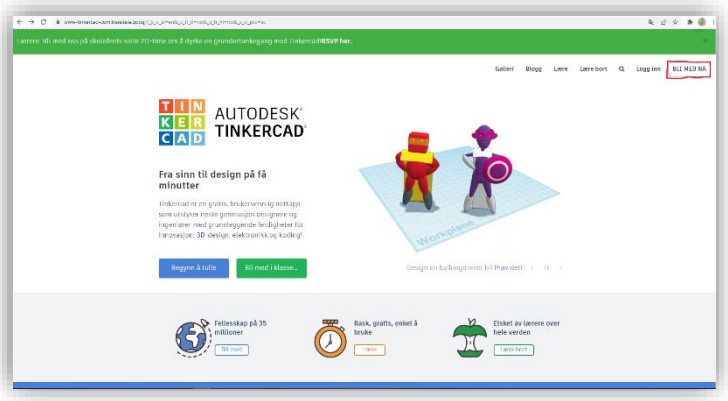
Figur 38https://www.ntnu.no/documents/2004699/12108297/Nordic+ESERO+Arduino+-+Grunnkurs+programemring+%28CanSat%29.pdf/6b388614-7aed-451e-92a9-

2.10 Tinkercad

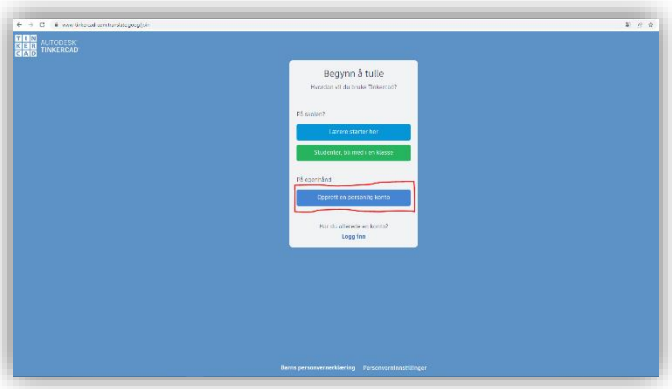
For at vi enkelt kan teste om kodene vi lager fungerer som tenkt, kan vi enkelt koble opp, lage og teste koden på en nettside som heter Tinkercad. Denne nettsiden er en nett-app hvor man kan opprette sin egen gratis bruker. Her finner vi utstyr og komponenter for å holde på med 3D-design, elektronikk og koding (Tinkercad, u.å.).

2.10.1 Hvordan opprette bruker?

1. Start med å gå inn på; www.tinkercad.com
2. Trykk på bli med nå som vist i bilde under.



3. Deretter oppretter du en personlig konto ved å trykke på der hvor det er markert på bilde under.



4. Gratulerer, du har opprettet din egen bruker på Tinkercad.

   Nå er det bare å begynne å leke seg med programmering og koding.
5. Søker du opp Tinkercad på youtube, kan du finne mange introduksjonsfilmer på hvordan man bruker Tinkercad.

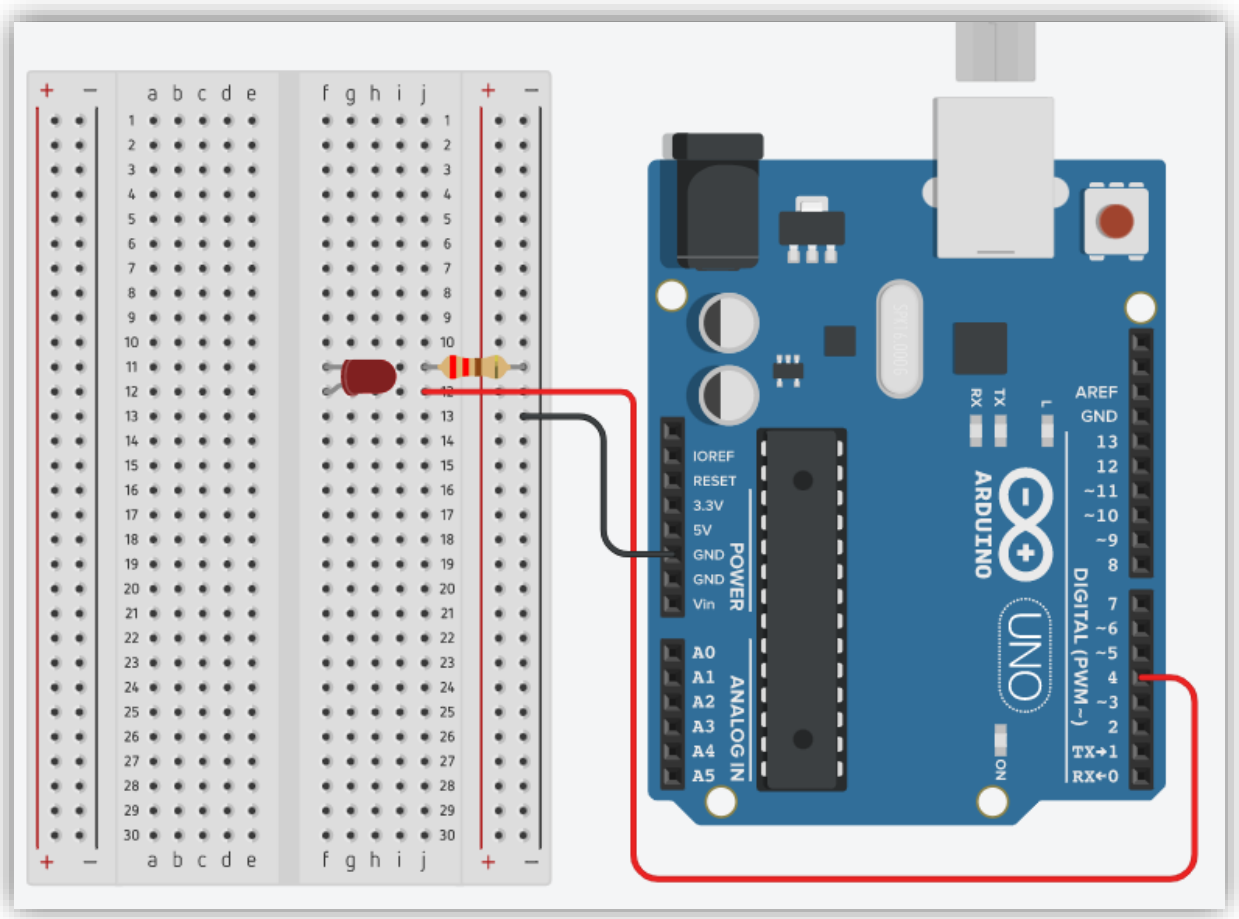
3.0 Praktiske oppgaver

Oppgave 1:

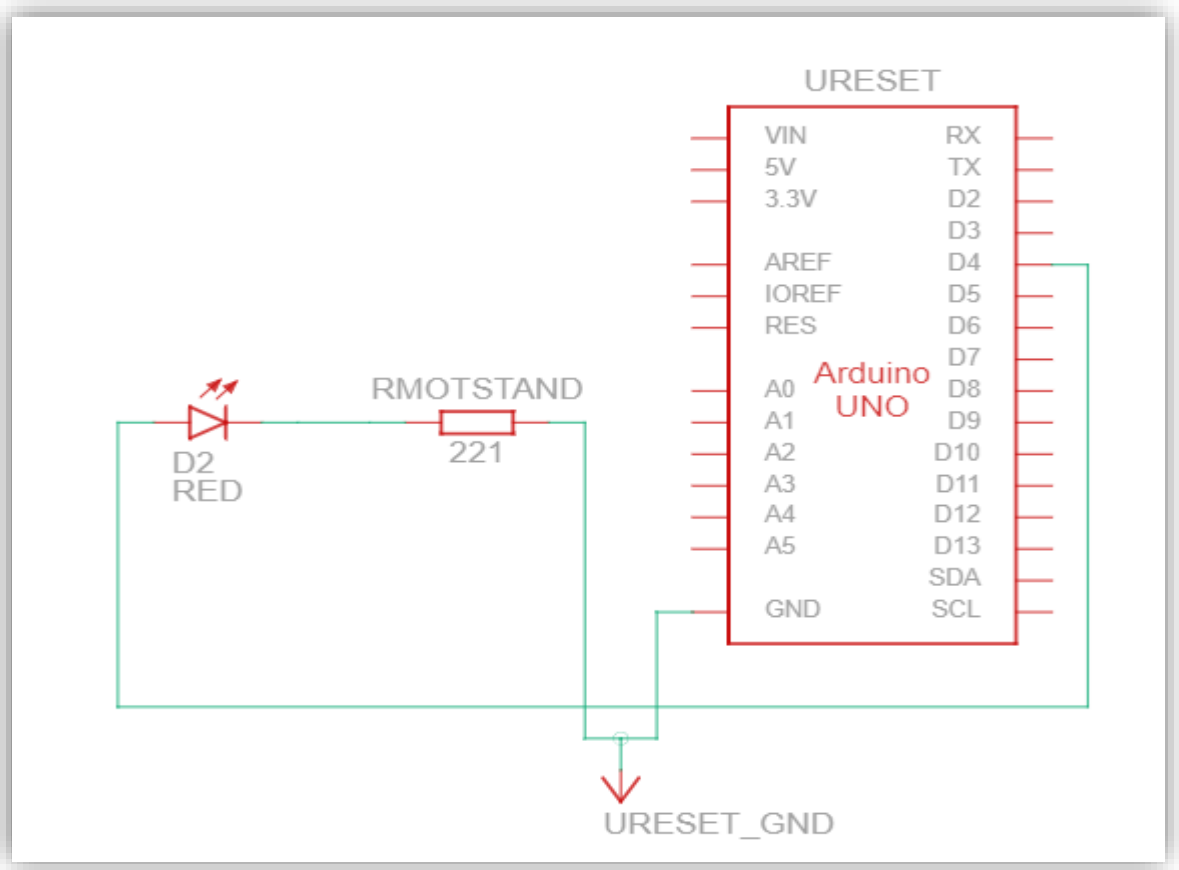
Her er målet å programmere de mest grunnleggende funksjonene som Arduino Uno innehar. I denne oppgaven skal det koples en diode mot strømkilden på Arduino-brettet, da vil den lyse og dette kalles for en «pull-down»-krets.

Her trenger du disse komponentene:

- Arduino Uno
- Koplingsbrett
- 1stk Diode i valgfri farge
- 1stk Motstand 220 Ohm (rød, rød, brun, gull)



Koblingsskjema:



Koden for å få denne til å fungere ser slik ut:

```
int redLed = 4;

void setup()
{
 pinMode(redLed, OUTPUT);
}

void loop()
{
  digitalWrite(redLed, HIGH);
  delay(1000);
  digitalWrite(redLed, LOW);
  delay(1000);
}

```

*Ved å endre verdien for «delay» endrer vi også takten LED 'n blinker.

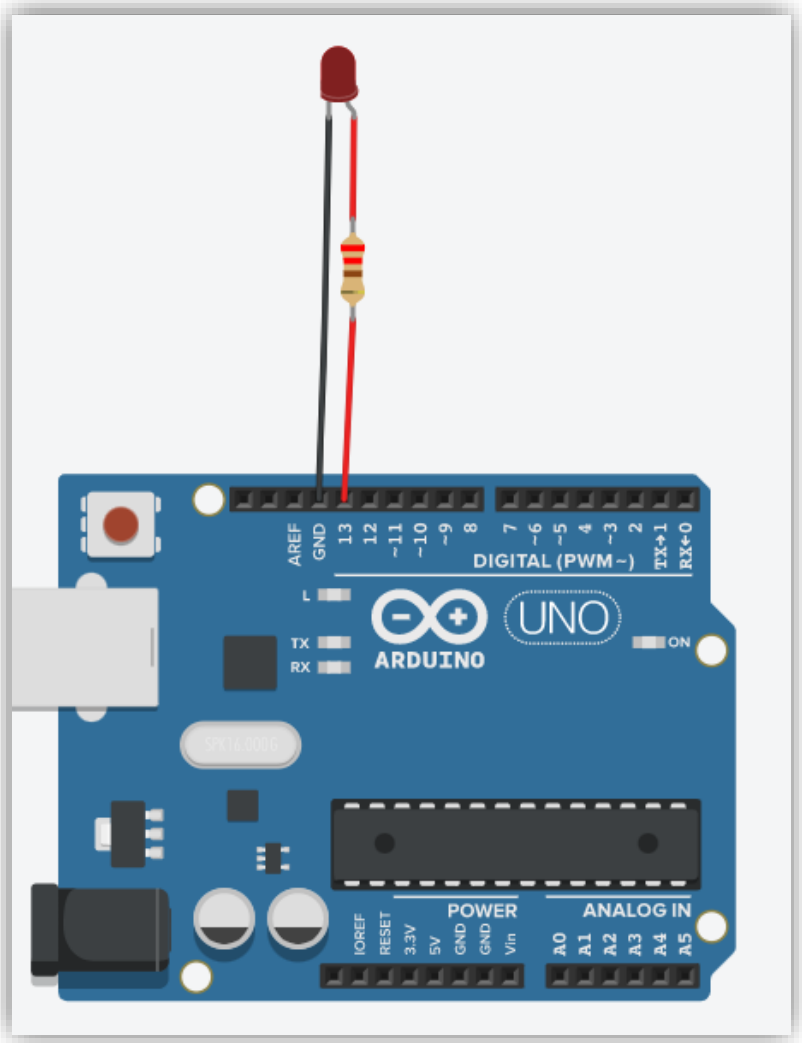
Oppgave 2: Blink med LED, og skriv God dag til Serial Monitor

Målet med denne oppgaven er at du skal koble opp en enkel LED, og få den til å blinke samtidig som det skal skriver «God dag» til Serial Monitor.

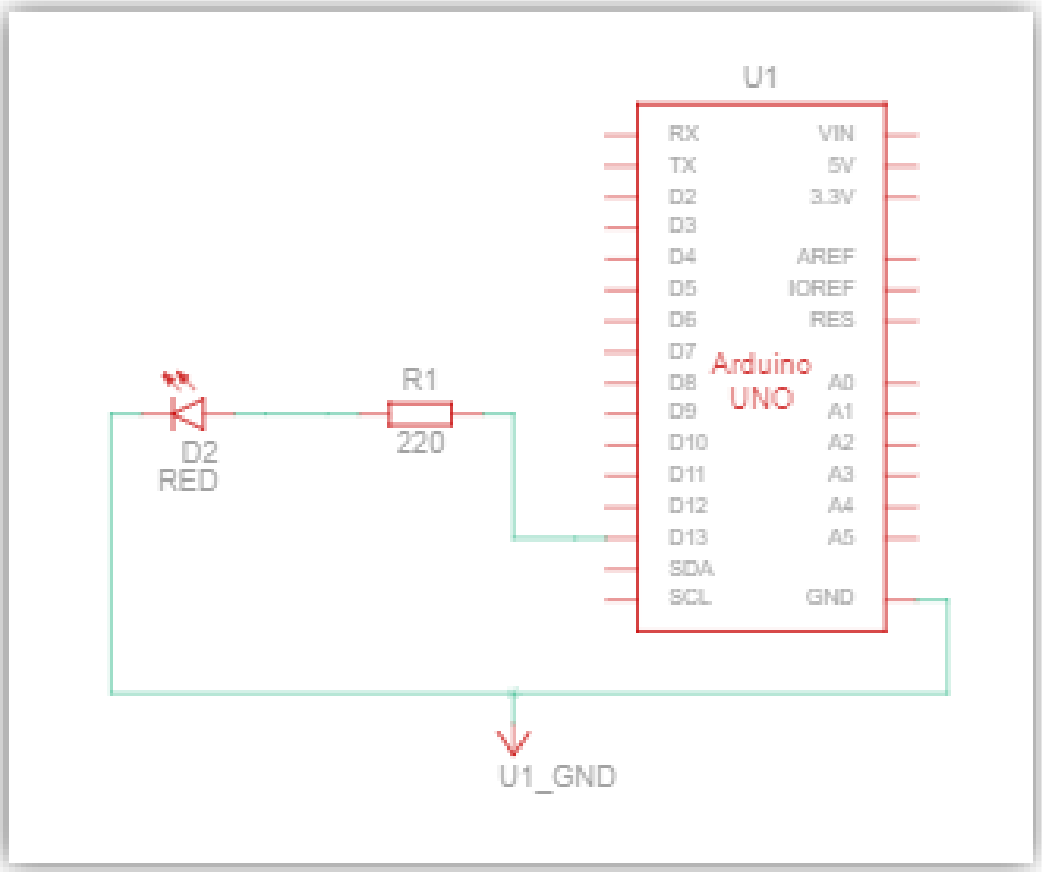
Komponenter til oppgaven:

- 1 stk diode rød
- 1 stk 220Ω motstand
- Arduino UNO

Koblinger på Arduino UNO



Skjemategning:



Kode:

```
int redLed = 13;  //setter redLed til digital pinne 13

void setup()//oppstartsblokken
{
  pinMode(redLed, OUTPUT);// setter redLed som utgang
  Serial.begin(9600);// Seriekommunikasjon med PCen
  Serial.println ("God dag!");// Skriver God dag til PCen
}

void loop()// Hovedblokken
{
  digitalWrite(redLed, HIGH);// Skrur på lysdioden
  delay(1000); // Venter 1000 millisekunder
  digitalWrite(redLed, LOW);// Skrur av lysdioden
  delay(1000); // Venter 1000 millisekunder
  Serial.println ("God dag!");// Skriver God dag til PCen
}
```

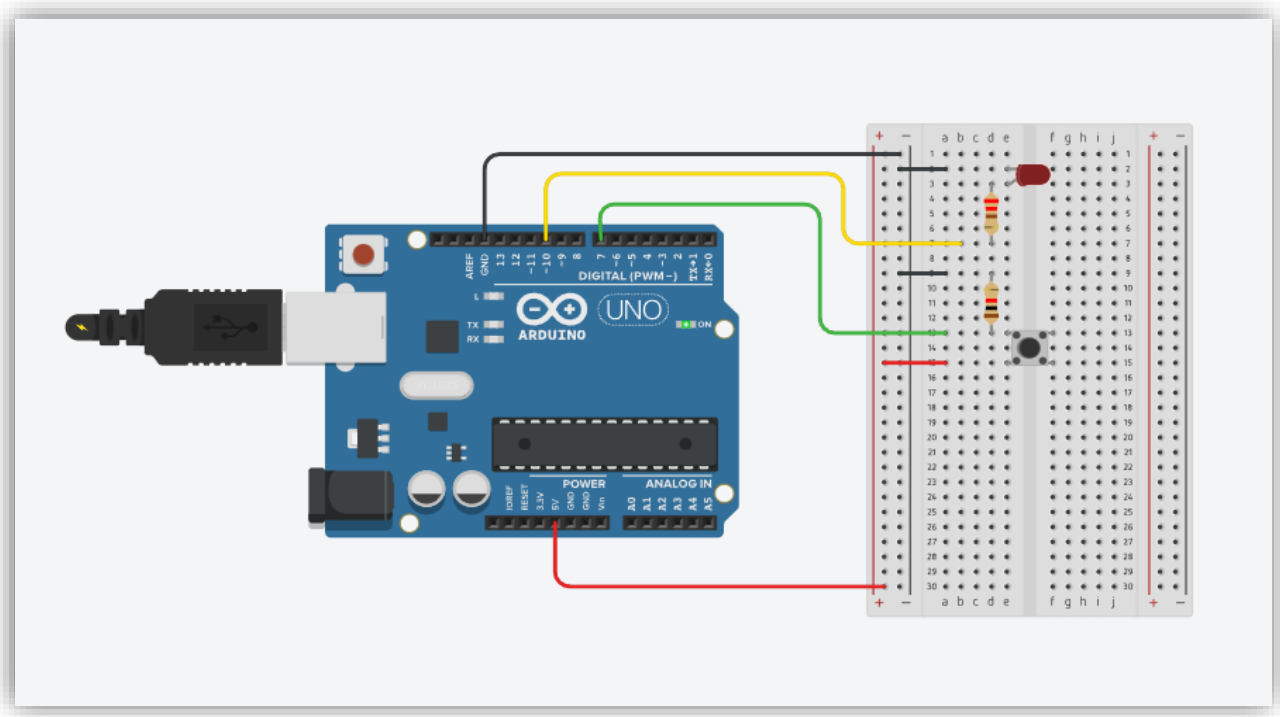
Oppgave 3: Led med bryter.

Oppgaven er å koble til en diode som skal lyse når vi holder inn bryteren. Bryteren skal ikke stå i serie med dioden, men brukes for å aktivere en utgang som får dioden til å lyse.

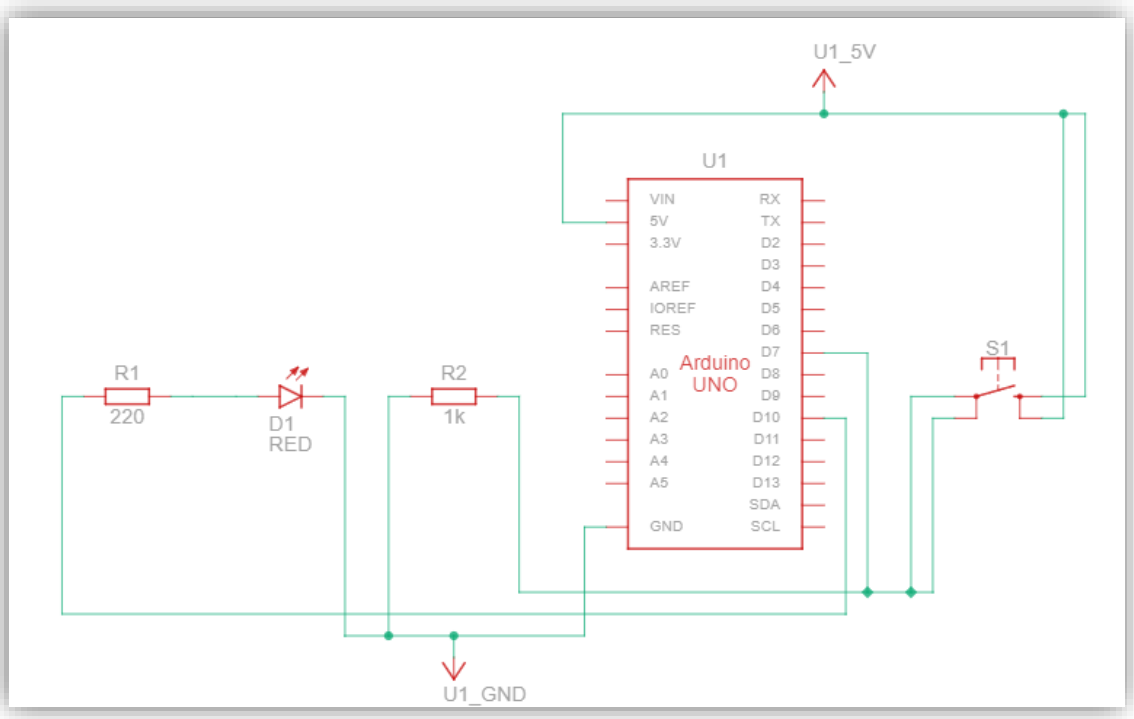
Komponenter til oppgaven:

- 1stk diode rød
- 1stk 220Ω motstand
- 1stk 1kΩ motstand
- 1stk trykkbryter
- Arduino Uno
- Koblingsbrett

Koblinger på Arduino UNO



Skjemategning:



Kode:

```
int ledPin = 10;
int buttonPin = 7;

void setup()
{
  pinMode(ledPin, OUTPUT);      //setter pinne 10 til utgang
  pinMode(buttonPin, INPUT);    //setter pinne 7 til inngang
}

Void loop()
{
  int value = digitalRead(buttonPin);
  if (value == LOW)                    //Hvis verdien er lav
  {
    digitalWrite(ledPin, LOW);         //Utgang 10 lav, led av
  }
  else                                 //hvis verdien er høy, bryter aktivert
  {
    digitalWrite(ledPin, HIGH);        //ledPin høy, led lyser
  }

```

Ekstra utfordring til oppgave 2, lag ett program som får dioden til å lyse ved å trykke en tur på knappen, trykker vi en tur til skal den slukke.

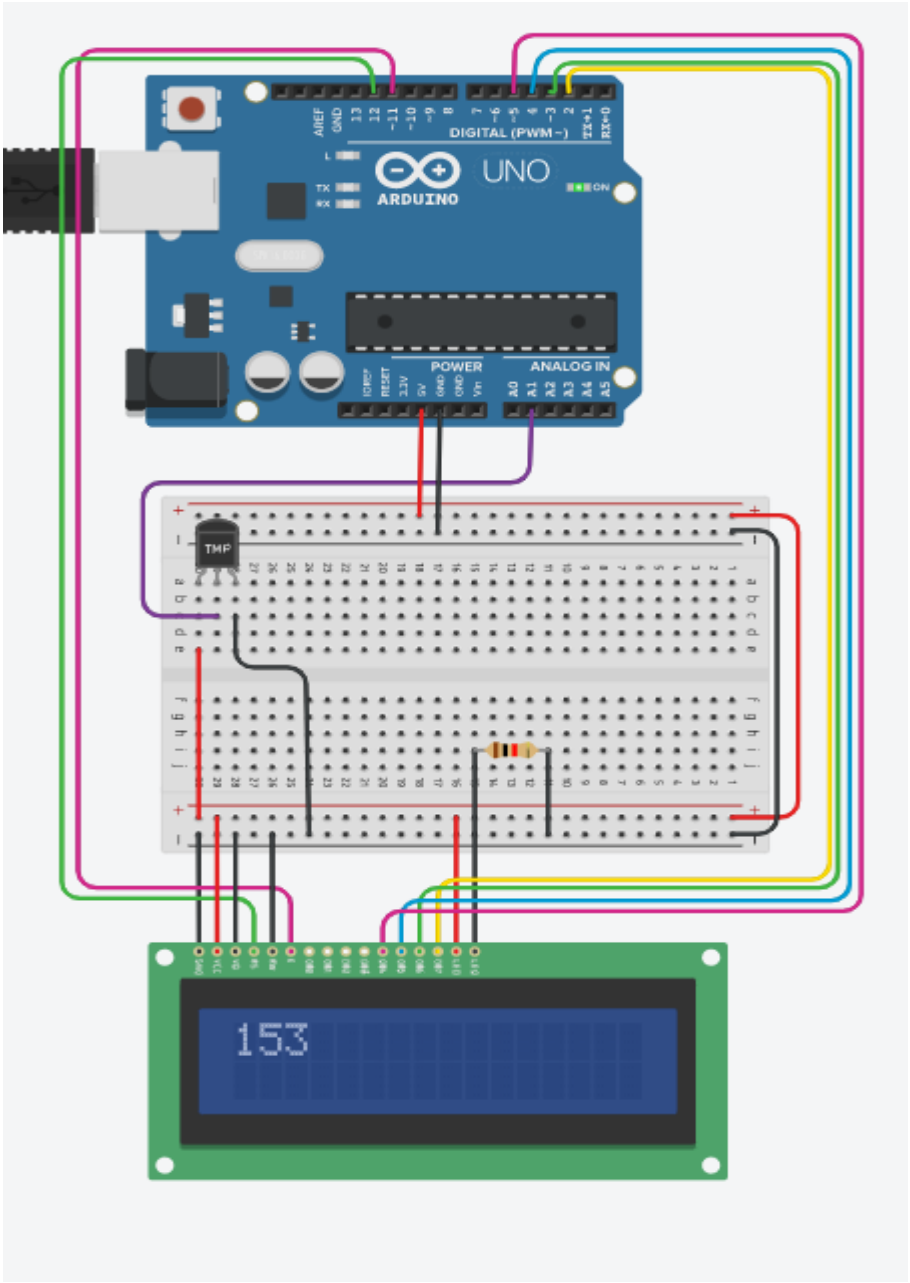
Oppgave 4:

Temperaturmåler med LCD-skjerm.

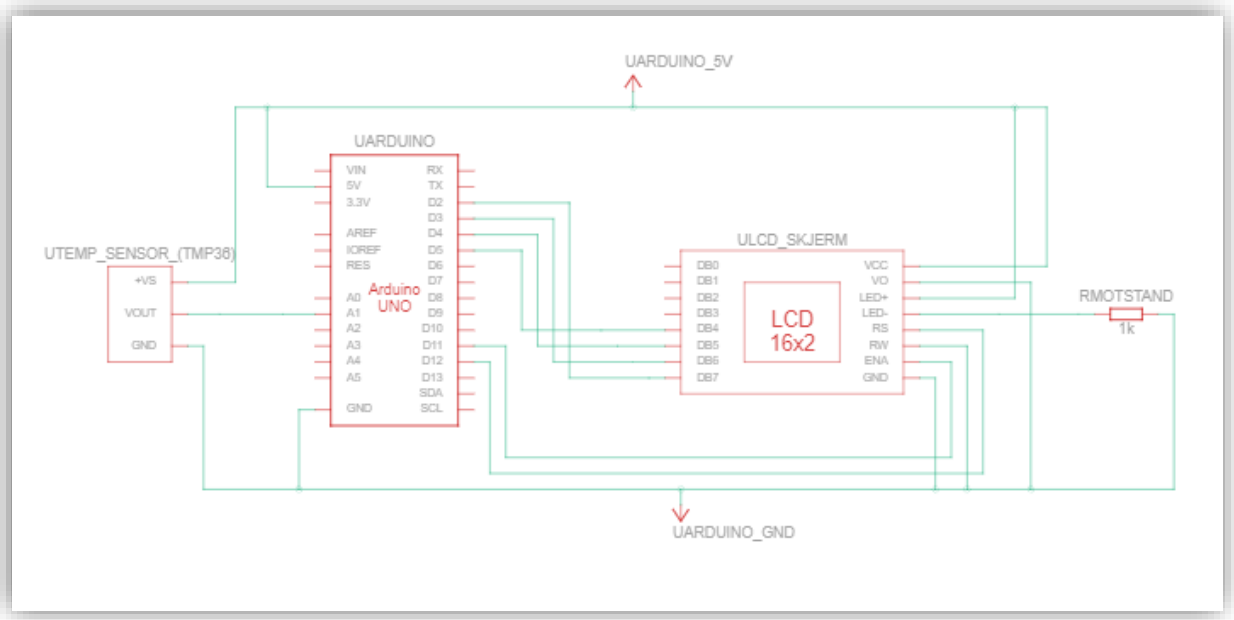
Komponenter til oppgaven:

- Arduino Uno
- Koblingsbrett
- LCD display 16:2
- 1stk Temp sensor
- 1stk 1kΩ motstand

Koplinger på Arduino Uno:



Skjemategning:



Kode:

```
#include <LiquidCrystal.h>
const int rs=12,en=11,d4=5,d5=4,d6=3,d7=2;
LiquidCrystal lcd(rs,en,d4,d5,d6,d7);
void setup()

{
Serial.begin(9600);
  pinMode(A1,INPUT);

}

void loop()
{
lcd.begin(16,2);
 lcd.setCursor(0,0);
  lcd.print(analogRead(A1));
 delay(500);
}
```

En kan på en oppkopling legge inn forskjellige koder. Den første koden som her blir presentert viser kun at tempsensoren viser ulike verdier i forhold til temperatur. Prøv gjerne å legge inn neste kode. Denne viser aktuell verdi i Celsius på LCD displayet.

```
const int sensorPin = A1;
const float romTemp = 20.0;

#include <LiquidCrystal.h>
LiquidCrystal lcd(12, 11, 5, 4, 3, 2);

void setup() {
  Serial.begin(9600);
  lcd.begin (16, 2);
  lcd.print("Temperatur");
  lcd.setCursor (0, 1);
  lcd.print(" vent litt....");
  delay(5000);
}

void loop() {
  int sensorVal = analogRead(sensorPin);

  Serial.print ("sensor Value: ");
  Serial.print (sensorVal);

  float voltage = (sensorVal / 1024.0) * 5.0;

  Serial.print(", Volts: ");
  Serial.print(voltage);

  Serial.print(", degrees C: ");
  float temperature = (voltage - .5) * 100;
  Serial.println(temperature);

  if (temperature < romTemp + 2) {
     lcd.begin (16, 2);
    lcd.print("Temperatur :");
    lcd.setCursor(7, 1);
    lcd.print("Celsius");
    lcd.setCursor(0, 1);
    lcd.print(temperature);
     }
 else if (temperature >= romTemp + 4 && temperature < romTemp + 6) {
    lcd.begin (16, 2);
    lcd.print("Temperatur:");
    lcd.setCursor(7, 1);
    lcd.print("Celsius");
    lcd.setCursor(0, 1);
    lcd.print(temperature);
  }
   else if (temperature >= romTemp + 2 && temperature < romTemp + 4) {
    lcd.begin (16, 2);
    lcd.print("Temperatur :");
    lcd.setCursor(7, 1);
    lcd.print("Celsius");
    lcd.setCursor(0, 1);
    lcd.print(temperature);
  }
  else if (temperature >= romTemp + 6) {
    lcd.begin (16, 2);
    lcd.print("Temperatur:");
    lcd.setCursor(7, 1);
    lcd.print("Celsius");
    lcd.setCursor(0, 1);
    lcd.print(temperature);

  }
  delay(2000);
}
```

Oppgave 5: Switch case

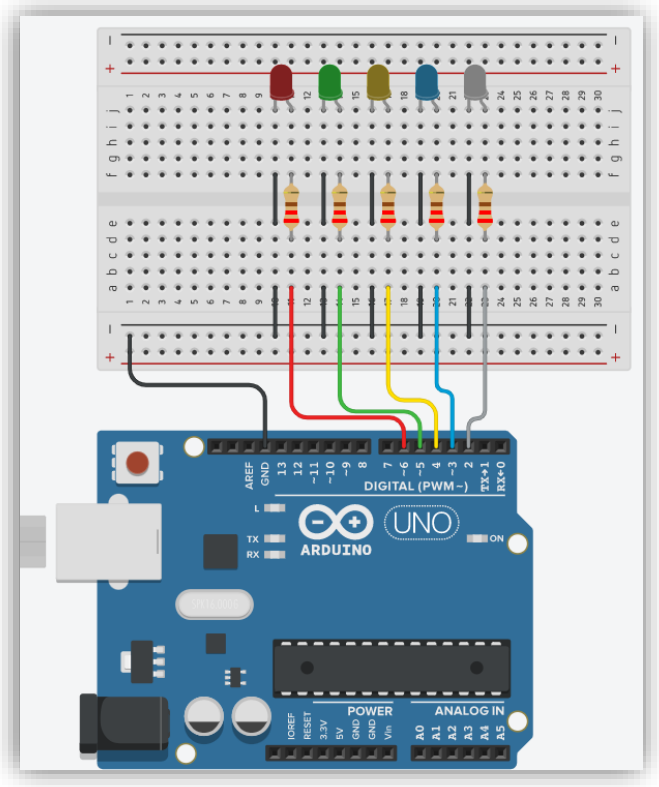
I denne oppgaven skal vi skru av å på LED-dioder ved å skrive til Serial Monitor.

Programkoden vår skal vi lage ut ifra en switch case.

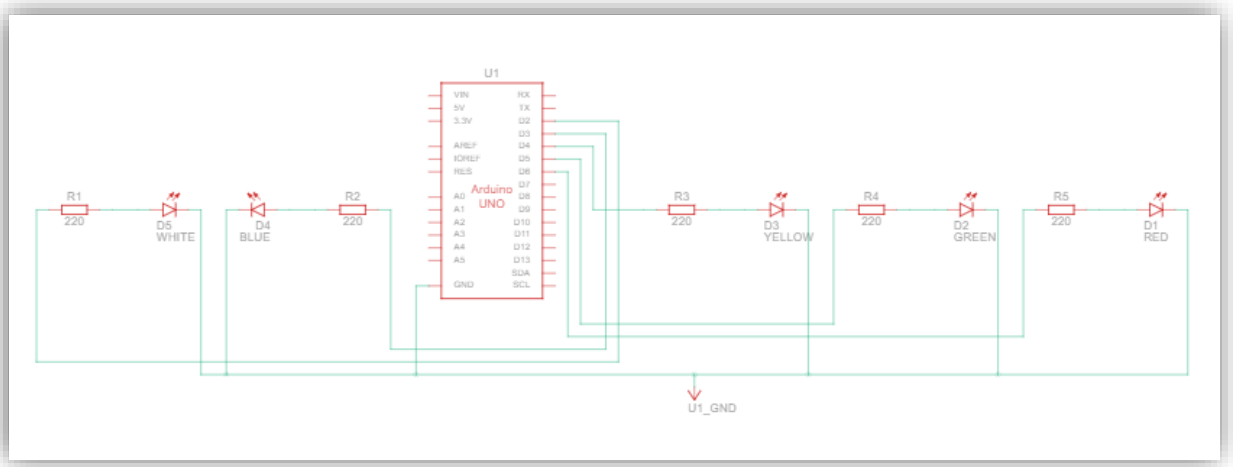
Komponenter til oppgaven:

- 5 LED-diode. Valgfri farge, men artig å velge forskjellige.
- 5 x 220Ω motstander.
- Koblingsbrett
- Arduino Uno

Koblinger på Arduino Uno



Skjemategning



Programkode

```
const int whiteLed = 2;   //setter whiteLed til digitalpinne 2
const int blueLed = 3;    //setter blueLed til digitalpinne 3
const int yellowLed = 4;  //setter yellowLed til digitalpinne 4
const int greenLed = 5;   //setter greenLed til digitalpinne 5
const int redLed = 6;     //setter redLed til digitalpinne 6

void setup()
{                                                  //void setup start
  Serial.begin(9600);                              //starter serieovervåking
  for(int thisPin = whiteLed; thisPin<redLed; thisPin++)  //bruker ei for-løkke for å telle mellom whiteLed og redLed
  {                                                //for-løkke start
    pinMode(thisPin, OUTPUT);                      //setter thisPin som OUTPUT
 }                                                 //for-løkke slutt
}                                                  //void setup slutt



void loop()
{
  if(Serial.available()>0)
  {
    int inbyte=Serial.read();
    switch (inbyte)                                //switch case for styringen
    {
      case 'w':                                    //case w = whiteLed
      digitalWrite(whiteLed, HIGH);                //Dersom det blir skrevet w til Serial Monitor, settes digital pinnen whiteLed HIGH
      break;

      case 'b':                                    //case b
      digitalWrite(blueLed, HIGH);                 //Dersom det blir skrevet b til Serial Monitor, settes digital pinnen blueLed HIGH
      break;

      case 'y':                                    //case y
      digitalWrite(yellowLed, HIGH);               //Dersom det blir skrevet y til Serial Monitor, settes digital pinnen yellowLed HIGH
      break;

      case 'g':                                    //case g
      digitalWrite(greenLed, HIGH);                //Dersom det blir skrevet g til Serial Monitor, settes digital pinnen greenLed HIGH
      break;

      case 'r':                                    //case r
      digitalWrite(redLed, HIGH);                  //Dersom det blir skrevet r til Serial Monitor, settes digital pinnen redLed HIGH
      break;

       case 'o':                                   //case o
      digitalWrite(whiteLed, LOW);                 //Dersom det blir skrevet o til Serial Monitor, vil vi skri av alle LED-diodene,
      digitalWrite(blueLed, LOW);                  //uavhengig av hvor mange du har skrudd på
      digitalWrite(yellowLed, LOW);
      digitalWrite(greenLed, LOW);
      digitalWrite(redLed, LOW);
      break;
      default:
      for(int thisPin=whiteLed;thisPin<redLed;thisPin++)
      {
        digitalWrite(thisPin, LOW);
      }
    }
  }
}
```

4.0 Arduino for Vg1 elektro og datateknologi

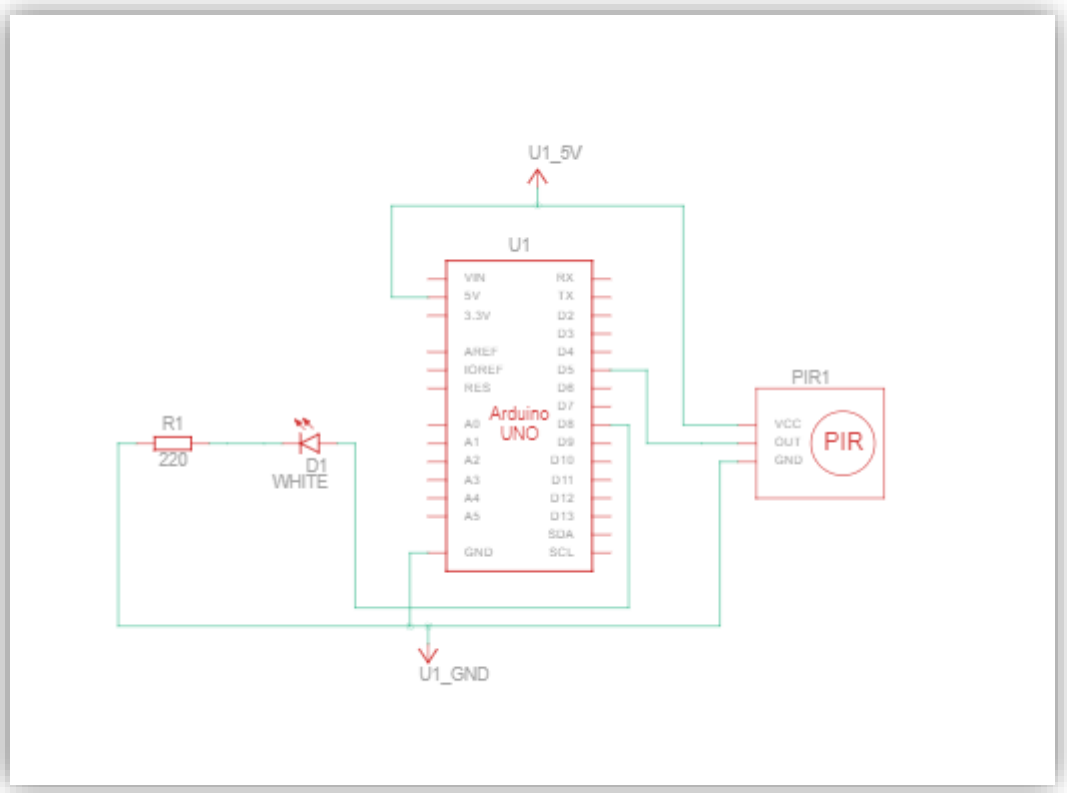
Oppgave 1

I denne oppgaven skal vi lage en programkode for automatisk tenning og slukking av et lys. Dette løser vi ved å benytte oss av en PIR detektor, som registrerer bevegelser. Dette er en styring som er veldig aktuell å benytte i korridorer, møterom, klasserom etc., på grunn av at det er en energi-økonomisk løsning.

Fremgangsmåte:

- Koble opp etter vedlagt koblingsskjema

- Lag koden slik at styringen fungerer etter forutsatt bruk



Løsningsforslag

https://www.tinkercad.com/things/hsyb2UBdwBp

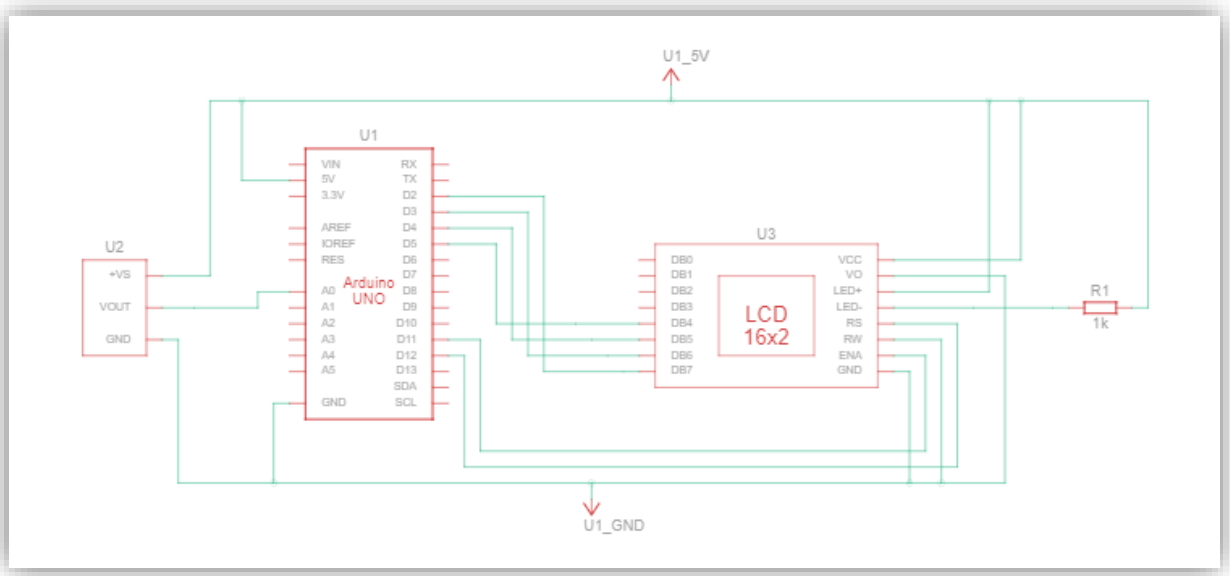
Oppgave 2

I denne oppgaven skal vi benytte oss av en temperatur sensor for å måle temperaturen, i for eksempel et rom, eller i en værstasjon. Temperaturen skal vises på en LCD-skjerm, og vi skal i Serial Monitor kunne se hva sensor verdien er, spenningen og hvor mange grader det er.

Fremgangsmåte:

- Koble opp etter vedlagt koblingsskjema

- Lag koden slik at styringen fungerer etter forutsatt bruk.



Løsningsforslag

https://www.tinkercad.com/things/jvqNalcKf2N

Fordypningsoppgave – Smart Home

I fremtiden vil stadig flere hus være såkalte smarthus. Du vil kunne ha muligheten til å styre alt via en mobiltelefon. I denne fordypningsoppgaven skal vi bygge oss et smarthus. Denne oppgaven er hentet ifra fra Keystudio.com (Keyestudio, 2021).

Figur 39
https://wiki.keyestudio.com/KS0085_Keyestudio_Smart_Home_Kit_for_Arduino#Instruction:

Inne på denne linken;
https://wiki.keyestudio.com/KS0085_Keyestudio_Smart_Home_Kit_for_Arduino#Instruction
finner du alt du trenger av informasjon for å gjennomføre oppgaven. Her har du:

- Instruksjon
- Utstyrsliste
- Nedlastning av software og installering av driver
- Hvordan legge til bibliotek
- Prosjekt
- Sett-sammen guide
- Relaterte ressurser

Denne oppgaven er basert på et ferdig kit man kan få kjøpt, men det er ingen problem med å løse oppgaven ved å lage huse selv om man har alle komponentene som trengs (Keyestudio, 2021).

Figur 40
https://www.keyestudio.com/products/keyestudio-smart-home-kit-withlus-board-for-arduino-diy-stem

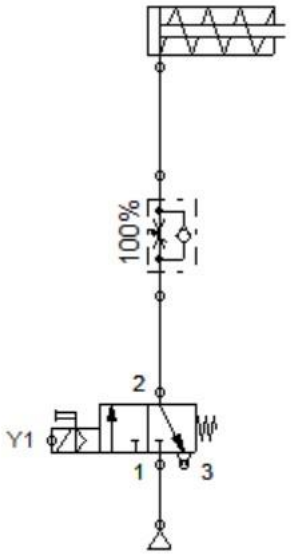
5.0 Arduino for Vg1 teknologi- og industrifag

Oppgave 1 elektropneumatikk

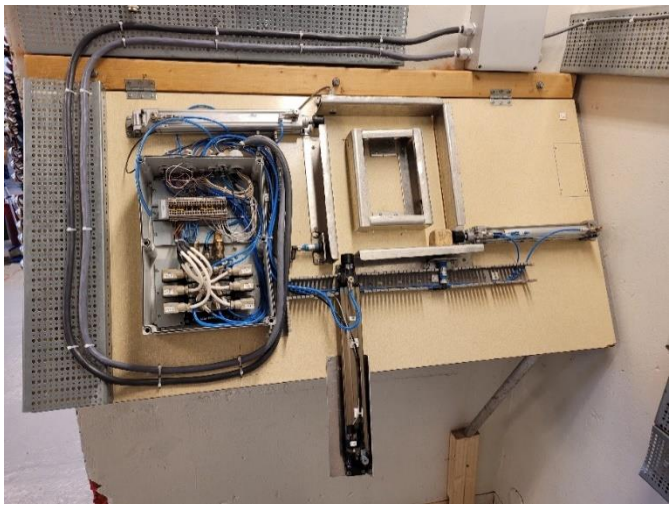
I denne oppgaven skal det monteres en giver på et transportbånd som registrere om det passere metallgjenstander. Kommer det metall, skal en sylinder skyve objektet av bandet og opp i en beholder. Dette er små deler, max 0,5kg. Når beholderen er ¾ full skal en lampe begynne å lyse. Blir den helt full skal en alarm gå. Dette kan løses på flere måter, men for å få mere innsikt i programmeringen og bruk av arduino skal dette styres med et arduinobrett.

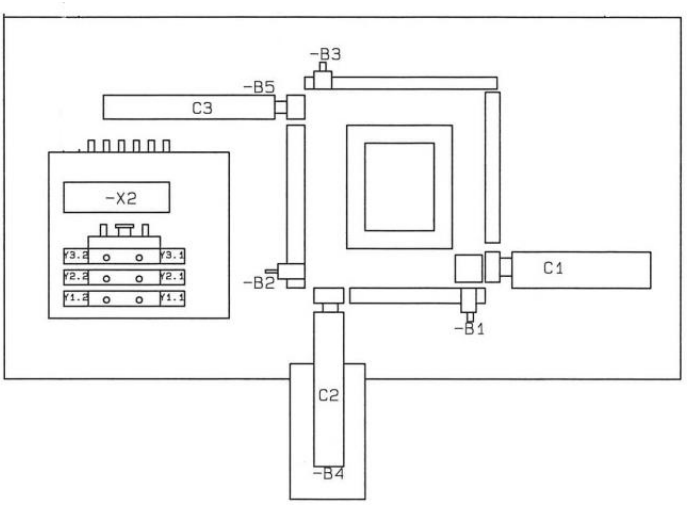
Utfordringer som må løses er:

- Hvordan skal metallet registreres?
- Hvordan forsikre seg at sylinderen går hele veien i +retning?
- Hvordan få koblet til arduino til å styre 24V?
- Det skal være komplett dokumentasjon, elektro og pneumatikk, med en beskrivelse av de forskjellige komponentene og hvordan de fungerer. Figuren under er et forslag på hvordan selve pneumatikkdelen kan løses.



Oppgave 2 elektropneumatikk





I oppgaven er det 3 sylindere som beveger en kloss rundt i rammen som vist på bildet.

Oppgaven skal løses med en arduinostyring og nødvendige sensorer. Her er det behov for releer og ekstern strømforsyning, 24V spoler på ventilene.

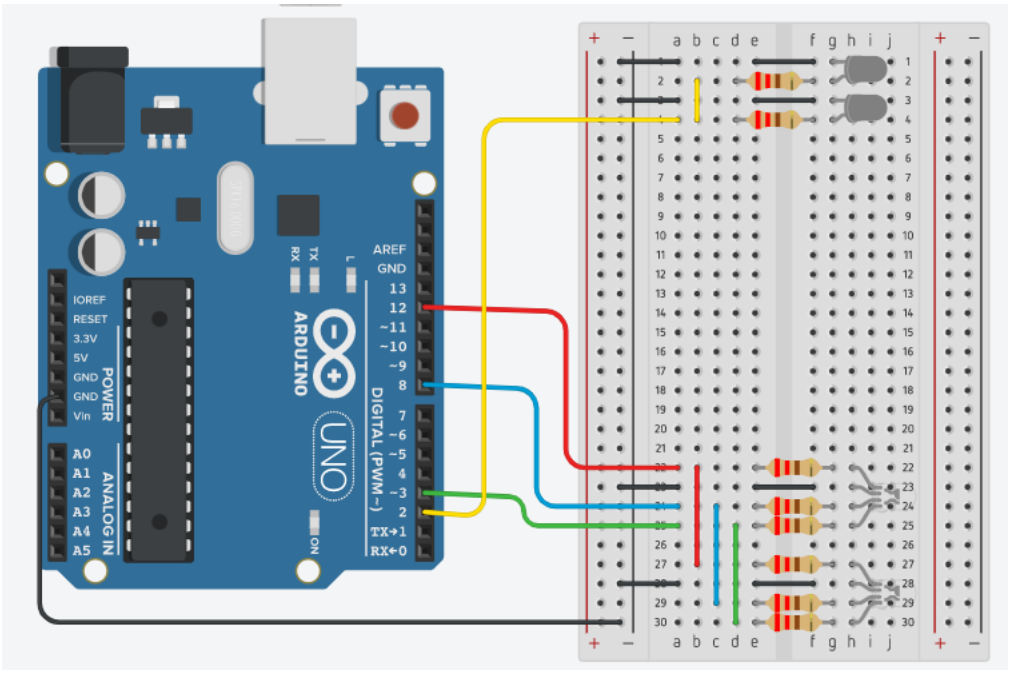
Sekvensen er: S1+, S2+ og S1-, S3+ og S2-, S3-

- Lag nødvendig dokumentasjon
    - Pneumatikkskjema i fluidsim.
    - El skjema og tilkoblinger til arduino i Fritzing
    - Programmet til arduino.

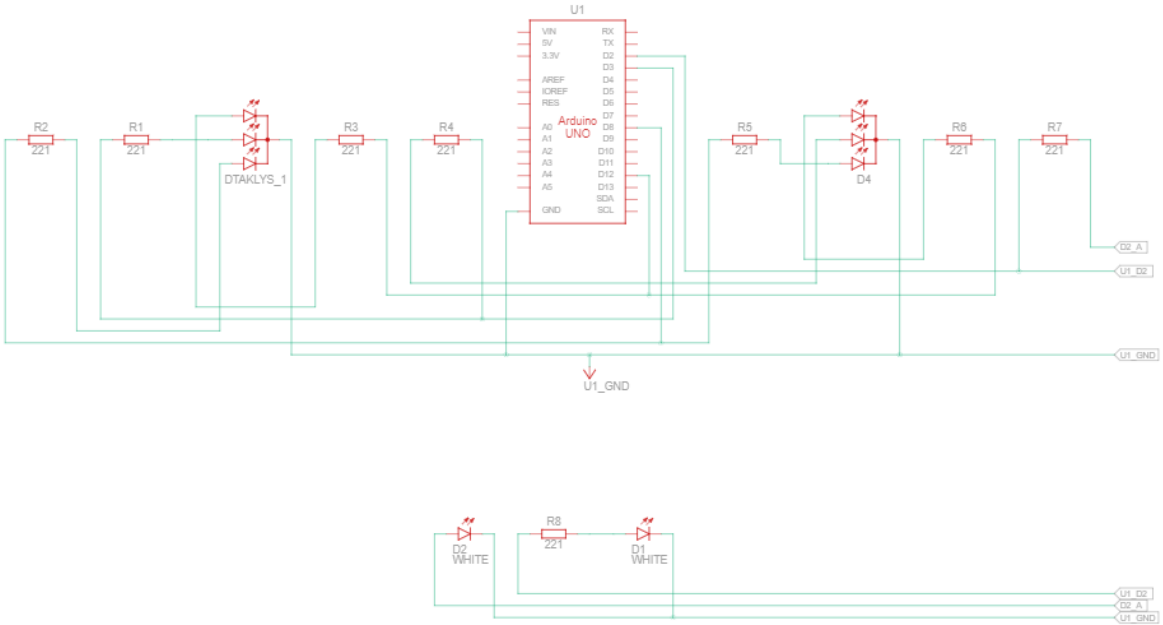
6.0 Tverrfaglige arbeidsoppgaver

Oppgave 1: Rubicon Jeep med forskjellige lys. Denne oppgaven kan brukes i klasserom og er forhåndskoplet. Dette fordi læringen skal ha søkelys på programmering og mindre på komponentene samt oppkoplingen. Bilen er utstyrt med hvite LED dioder som frontlys og RGB dioder på tak.

Oppkoplingen på bilen ser slik ut om den ble koplet på brett.



Koblingsskjema:



Programkode:

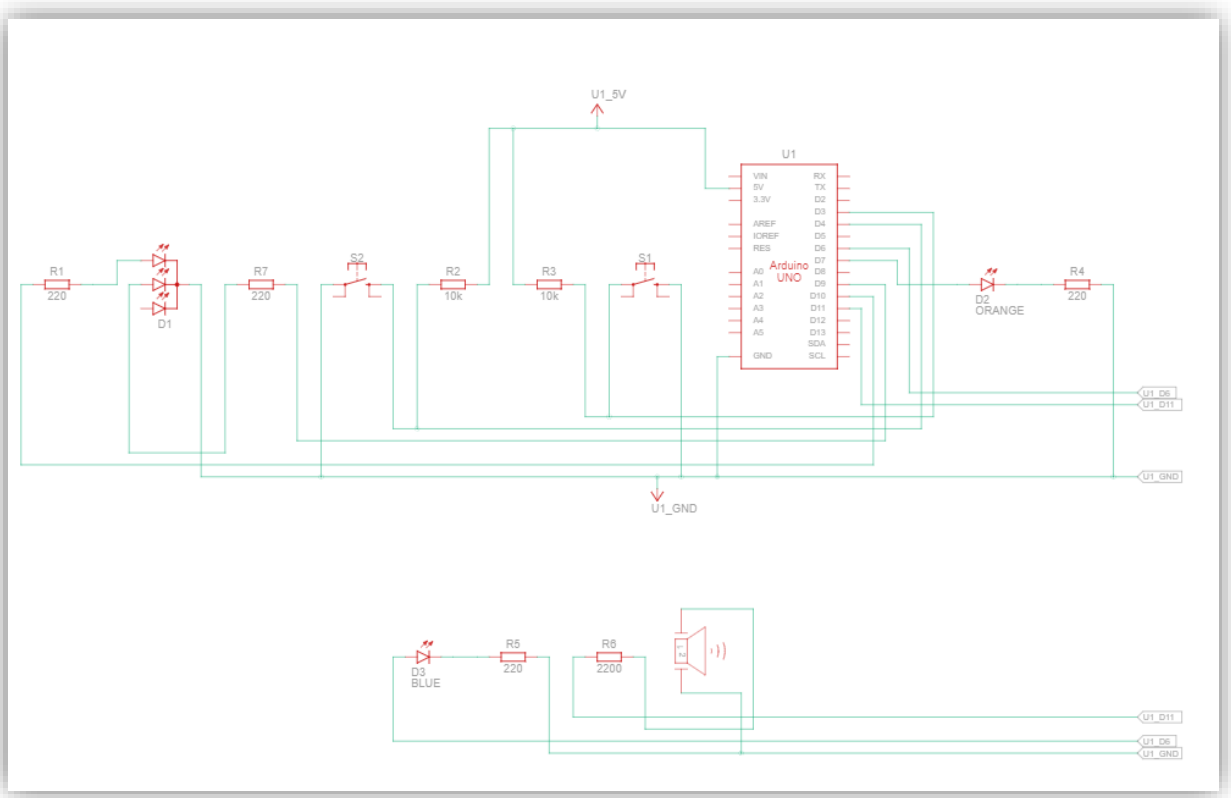
```
// intigrerer de forksjellige diodene og rgb fargene
int red;
int green;
int blue;
int whiteLed =2;


const int GREEN = 3;  // setter en greenLed til digitalpinne 3
const int BLUE = 8;  // setter en blueLed til digitalpinne 8
const int RED = 12; // setter en redLed til digitalpinne 12

void setup(){
 pinMode(whiteLed, OUTPUT) ; // setter whiteLed som en OUTPUT Frontlys

}
void loop (){


digitalWrite(whiteLed, HIGH);   // Får Fremlysene til å lyse)

  red = 255; green = 0; blue = 0; // redLed Lyser
  for(green = 0; green <= 255; green++) // greenLed Lyser
  setRGB(red, green, blue);
  for(red = 255; red >= 0; red--) // rødLed slukkes
  setRGB(red, green, blue);
  for(blue = 0; blue <= 255; blue++) // blueLed Lyser
  setRGB(red, green, blue);
  for(green = 255; green >= 0; green--) // greenLed slukkes
  setRGB(red, green, blue);
  for(red = 0; red <= 255; red++) // redLed Lyser
  setRGB(red, green, blue);

}
// RGB - lysdiode fargeinstallasjonsfunksjon
void setRGB(int r, int g, int b){
  analogWrite(RED, r);
  analogWrite(GREEN, g);
  analogWrite(BLUE, b);
  delay(10) ; // blir pause på 10millisekund i loopen
}
```

Oppgave 2 – reaksjonsspill

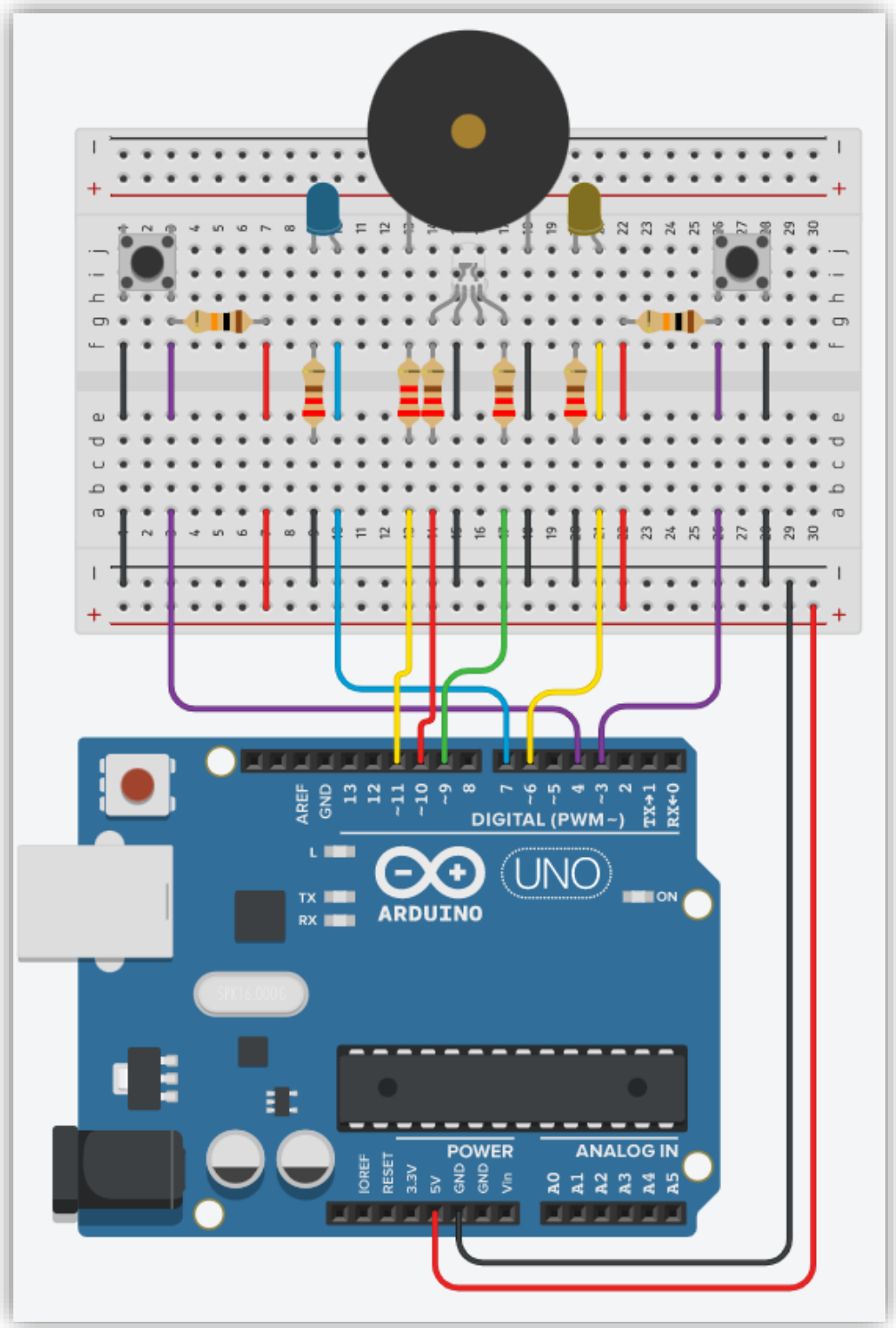
I denne oppgaven skal vi lage oss vårt eget reaksjonsspill for 2 spillere. Hver spiller har sin egen trykknappbryter, og når lys dioden skifter farge fra rødt til grønt, er det første mann til å trykke på sin knapp. Den som trykker først, får 10 poeng.

For at vi skal kunne starte spillet må vi skrive til Serial Monitor s (start). Vi skal også legge til en reset (r). Når vi skriver r til Serial Monitor, skal vi reset 'e spillet og poengene nullstilles. For å avslutte spillet skal vi skrive q (quit) til Serial Monitor, og da skal det skrives til Serial monitor hvem som er vinneren av spillet og hvor mange poeng vinneren har.

<u>Koblingsskjema:</u>



Oppkobling i Tinkercad:



Utstyrsliste:

- 1 stk gul LED
- 1 stk Blå LED
- 1 stk RGB
- 1 stk Buzzer
- 2 stk trykknapp
- 4 stk 220Ω motstand
- 2 stk 10kΩ motstand
- 1 stk 2,2kΩ motstand
- Arduino UNO
- Koblingsbrett

Kode:

Her vil jeg at dere skal prøve å lage koden uten at dere bruker løsningsforslaget i første omgang. Her kommer noen hint til hva koden bør inneholde:

- Før void setup: Deklarere globale variabler og konstanter
- Void setup: Kjøres bare en gang, og her utføres alle operasjoner og innstillinger som må være gjort før hovedprogrammet kjøres
- I void setup må vi få ordnet med en programlinje som skriver en liten introduksjon til spillet i seriemonitoren
- Void loop: Her skal vi skrive selve hovedprogrammet
- Det første vi vil skal skje i void loop er å lese om det har kommet noen kommando fra seriemonitoren
- Hvis det har kommet en kommando, skal vi lese den
- Deretter skal vi deklarere variabelen inbyte, og lese kommandoen
- Når vi har fått inn en kommando blir neste steg å avgjøre hva den betyr
- I dette tilfelle skal vi benytte oss av en switch state, på grunn av at vi skal kunne velge imellom flere alternativer

- Vi starter med å opprette en case s (start) som vi må skrive til serimonitoren for å få startet spillet
- Etter at vi har gjort det skal vi sette opp en tilfeldig forsinkelse for rødt til grønt lys i RGB 'n
- Vi må også kunne finne ut om vi har en taper, dvs. en spiller har trykket ned knappen før lyset har endret farge fra rødt til grønt
- Vi må også lage en programlinje for at lyset skifter fra rødt til grønt uten at vi har feiltrykk
- Nå skal vi lage en kode for lys- og lydfanfare til vinneren, og opprette en else-løkke for feiltrykk, dvs fault ikke == 0, og vi har en taper
- Vi oppretter en egen case for q (quit) hvor vi ønsker å oppsummere poeng og kåre en vinner
- Til slutt skal vi lage en siste case som heter r (reset) som skal reset' e spillet, og poengsummen nullstilles

Løsningsforslag:

```
/*
  - I denne koden har vi laget et reaksjonsspill, som er beregnet for to spillere.
  - Hver spiller har hver sin trykknapp, og hvert sitt lys.
  - Når RGB'en skifter fra rødt til grønt lys, er det om og gjøre å trykke først på sin knapp.
  - Den som trykker først vinner runden, og får 10 poeng.
*/

int buzzerPin = 11;  //Setter buzzerPin til digital pinne 11
int redLed = 10;     //Setter redLed til digital pinne 10
int greenLed = 9;    //Setter greenLed til digital pinne 9

int blueLed = 7;     //Setter blueled til digital pinne 7 (Spiller 1)
int yellowLed = 6;   //Setter yellowLed til digital pinne 6 (Spiller 2)
int buttonPin1 = 4;  //setter buttonPin til digital pinne 4 (Spiller 1)
int buttonPin2 = 3;  //Setter buttonPin2 til digital pinne 3 (Spiller 2)

int winner = 0;       //Initialiserer vinnerindikatoren
int winnerBeep = 750; //Buzzer-pitch --> Fanfare for vinneren
int fault = 0;        //Initialiserer feilindikatoren
int faultBeep = 200;  //Buzzer-pitch --> Feillyden

/*
  - Nå skal vi sette inn variabler for poengene.
*/

int pointsPlayerA = 0;  //Poeng spiller A
int pointsPlayerB = 0;  //Poeng spiller B

unsigned long wait = 0;
unsigned long now = 0;


void setup()
{
 Serial.begin(9600);   //Starter seriemonitoren
 Serial.println("Reaksjonsspill for 2 spillere!");
 Serial.println("Hver spiller har hver sin trykknapp");
 Serial.println("Tilgjengelige kommandoer: s (start), q (quit) og r (reset)");

 pinMode(redLed, OUTPUT);       //Setter redLed som OUTPUT
 pinMode(greenLed, OUTPUT);     //Setter greenLed som OUTPUT
 pinMode(blueLed, OUTPUT);     //Setter blueLed som OUTPUT
 pinMode(yellowLed, OUTPUT);    //Setter yellowLed som OUTPUT
 pinMode(buttonPin1, INPUT);    //Setter buttonPin1 som INPUT
 pinMode(buttonPin2, INPUT);    //Setter buttonPin2 som INPUT
}

void loop()
{

  // Det første vi vil skal skje i void loop er å lese om det har kommet noen kommando fra seriemonitoren
  // Hvis det har kommet en kommando, skal vi lese den

  if(Serial.available () > 0)
  {
   char inbyte=Serial.read ();  //Her deklarerer vi variabelen inbyte, og leser kommandoen inn i den

   // Nå har vi fått inn en kommando, og da blir neste steg å avgjøre hva den betyr
   // Vi skal benytte en switch state her, ettersom at vi skal kunne velge i mellom flere alternativer

   switch(inbyte)
   {
    case 's':
    digitalWrite(redLed, HIGH);  //Slår på rød RGB

    // Nå skal vi sette opp en tilfeldig forsinkelse for rød -> grønn i RGB'en

    fault = 0;
    now = millis ();
    wait = now + random (300, 600);
    while(millis () < wait && digitalRead (buttonPin1) == HIGH && digitalRead (buttonPin2) == HIGH)
    {
     }

```

```
    // Nå skal vi finne ut om vi har en taper: Dvs. knapp trykket ned før forsinkelsen vår har løpt ut

    if(digitalRead (buttonPin1) == LOW)fault = blueLed;
    if(digitalRead (buttonPin2) == LOW)fault = yellowLed;
    digitalWrite(redLed, LOW);                          //Slår av RGB

    // Hvis forsinkelsen utløper helt, uten at vi har feiltrykk:

    if(fault == 0)
    {
     digitalWrite(greenLed, HIGH);                  //Slår på grønn RGB. Spillet har startet

     //Bruker en while-løkke for å vente på at en spiller trykker på knappen sin
     while(digitalRead(buttonPin1) ==HIGH && digitalRead(buttonPin2) ==HIGH)
      }

     if(digitalRead(buttonPin1) ==LOW)  //Hoppet ut av while-løkka fordi en spiller har trykket ned sin knapp
     {
      winner = blueLed;

      pointsPlayerA = pointsPlayerA + 10;  //Teller 10 poeng hvis spiller A vinner
      pointsPlayerB = pointsPlayerB + 0;   //Spiller B får 0 poeng hvis spiller A vinner
      Serial.print("Poeng spiller 1 = ");
      Serial.print(pointsPlayerA);

      Serial.print("\t Poeng spiller 2 = ");
      Serial.println(pointsPlayerB);
     }
```

```
       else
       {
        winner = yellowLed;

        pointsPlayerA = pointsPlayerA = 0;   //Poeng hvis spiller B vinner
        pointsPlayerB = pointsPlayerB = 10;  //Spiller B vinner, og får 10 poeng
        Serial.print("Poeng spiller 1 = ");
        Serial.println(pointsPlayerA);

        Serial.print("\t Poeng spiller 2 = ");
        Serial.println(pointsPlayerB);
       }

       //Kode for lys- og lydfanfare til vinneren

       for(int k = 0; k < 5; k++)
       {
        tone(buzzerPin, (winnerBeep+(k*20)));
        digitalWrite(greenLed, HIGH);
        digitalWrite(winner, HIGH);
        delay(50);
        digitalWrite(winner, LOW);
        digitalWrite(greenLed, LOW);
        delay(50);
       }

       noTone(buzzerPin);
      }

```

```
    //Oppretter en else-løkke for feiltrykk, dvs fault ikke == 0, og vi har en taper:

    else
    {
     tone(buzzerPin, faultBeep, 500);   // Lager en feillyd på 5 sekunder og blink for taperen
     for(int k = 0; k < 10; k++)
     {
      digitalWrite(redLed, HIGH);
      digitalWrite(fault, HIGH);
      delay(50);
      digitalWrite(redLed, LOW);
      digitalWrite(fault, LOW);
      delay(50);
     }
    }

     break;  //Avslutter case s

    case 'q':

    if(pointsPlayerA > pointsPlayerB)
    {
     Serial.print("Spiller 1 vinner med: ");
     Serial.println(pointsPlayerA - pointsPlayerB);
    }

```

```
159      else if(pointsPlayerA = pointsPlayerB)
160      {
161       Serial.print("Vi har ingen vinner     ");
162       Serial.println("Uavgjort. Dere fikk like mange poeng!");
163      }
164
165      else
166      {
167       Serial.print("Spiller 2 vinner med: ");
168       Serial.println(pointsPlayerB - pointsPlayerA);
169      }
170
171      break;
172
173      case 'r':
174
175      Serial.println("Reset, poengene nullstilles");
176      pointsPlayerA = 0;
177      pointsPlayerB = 0;
178      break;
179     }
180    }
181 }
```

7.0 Forberedende øvingsoppgave til fordypningsoppgave – iskremmaskin

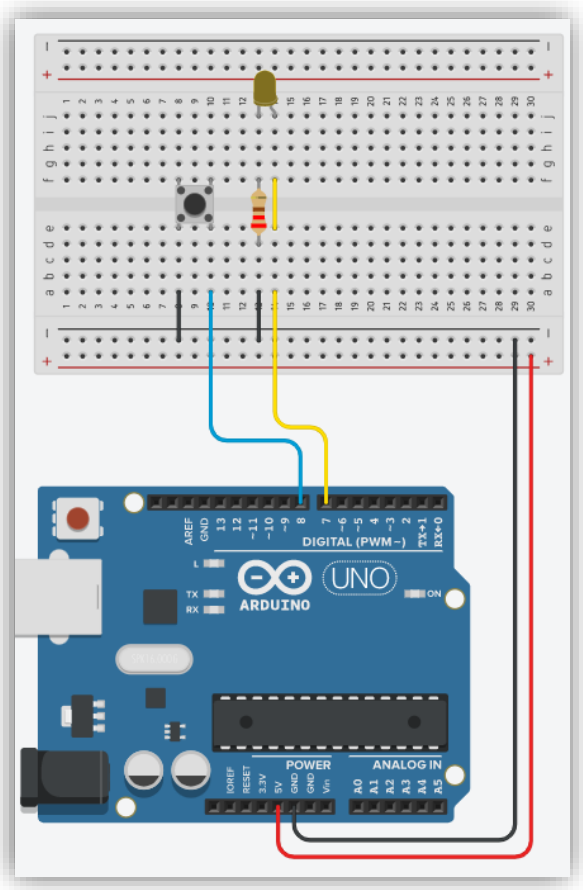
I denne forberedende oppgaven skal du få arbeide med en oppgave som vil være til stor hjelp når du skal starte på fordypningsoppgaven. Ved å bruke en switch-case skal du lage programkoden for tre enkle sekvensielle styringer. Det vil være til stor hjelp for deg å bruke teorien fra iskremmaskinen i denne øvingsoppgaven.

Oppgave 1

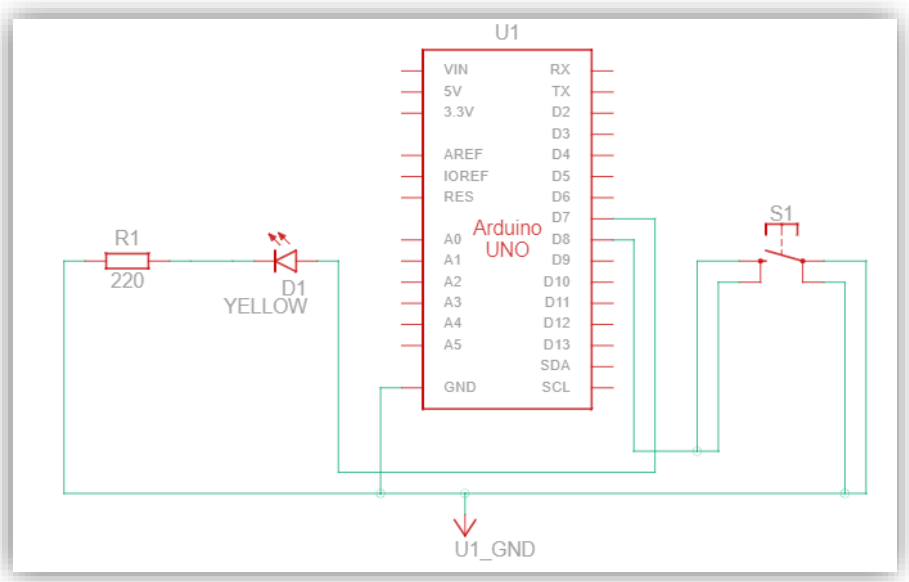
Her skal vi lage en kode som får et lys til å lyse når bryteren er trykt ned.

Du skal opprette en egen funksjon som heter light(). Inne i void loop bruker du en while-løkke som skal vente på knappetrykk, og henter inn funksjonen light().

Oppkobling på brettet:



Koblingsskjema:



Programkode:

```
const int yellowLed = 7;  //yellowLed til digitalpinne 7
const int buttonPin = 8;  //buttonPin til digitalpinne 8

int buttonValue;          //Startverdien til testvariabelen

void setup() {
  pinMode(yellowLed, OUTPUT);          //Setter yellowLed som OUTPUT
  pinMode(buttonPin, INPUT_PULLUP);    //Setter buttonPin som INPUT_PULLUP
}

void loop() {
  while(digitalRead(buttonPin)){}     //Venter på knappetrykk fra buttonPin
  light();                            //buttonPin er trykt, og funksjonen light kjøres
}

void light(){                         //Oppretter en funksjon som heter yellowLed
  if(!buttonValue) {                  //If-setningen kjøres hvis buttonPin er trykt ned
    digitalWrite(yellowLed, HIGH);    //setter yellowLed HIGH
    }
  }

```

Oppgave 2

I oppgave to skal vi ta i bruk serieovervåkningen. Vi skal bruke samme programkode og oppkobling som i oppgave 1, men nå skal det også skrives til seriel monitor at yellowLed lyser.

Programkode:

```
const int yellowLed = 7;  //yellowLed til digitalpinne 7
const int buttonPin = 8;  //buttonPin til digitalpinne 8

int buttonValue;          //Startverdien til testvariabelen

void setup() {
  Serial.begin(9600);                 //Åpner serieovervåkningen
  pinMode(yellowLed, OUTPUT);         //Setter yellowLed som OUTPUT
  pinMode(buttonPin, INPUT_PULLUP);   //Setter buttonPin som INPUT_PULLUP
}

void loop() {
  while(digitalRead(buttonPin)){}    //Venter på knappetrykk fra buttonPin
  light();                           //buttonPin er trykt, og funksjonen light kjøres
}

void light(){                                //Oppretter en funksjon som heter yellowLed
  if(!buttonValue) {                         //If-setningen kjøres hvis buttonPin er trykt ned
    Serial.println("yellowLed lyser");   //skriver til Serial Monitor
    digitalWrite(yellowLed, HIGH);           //setter yellowLed HIGH
    }
  }

```

oppgave 3

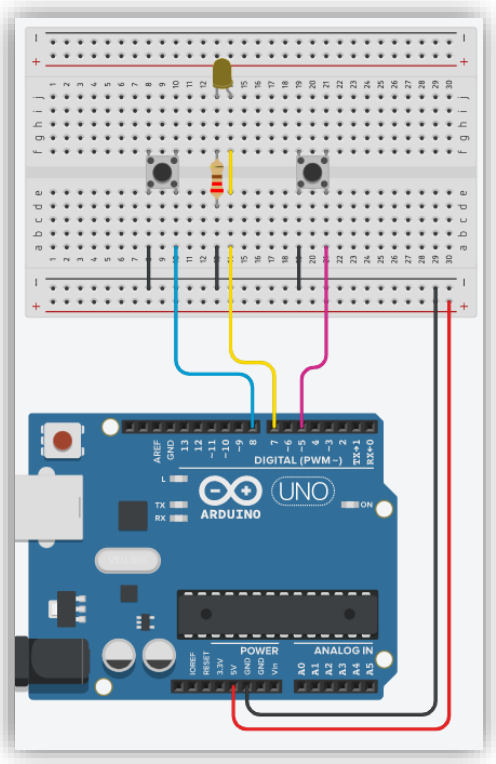
Nå skal vi sette sammen oppgave 1 og 2 i en switch-case. Vi skal koble opp en trykknapp til, buttonPin2, som skal hjelpe oss å hoppe fra en case til neste case.

Case 1: Skrive til serial monitor at maskinen starter opp og det skal rapporteres status for case'n. Når dette er utført skal vi vente i 1000 ms før det kommer en ny beskjed i serial monitor om at vi skal trykke på buttonPin2.

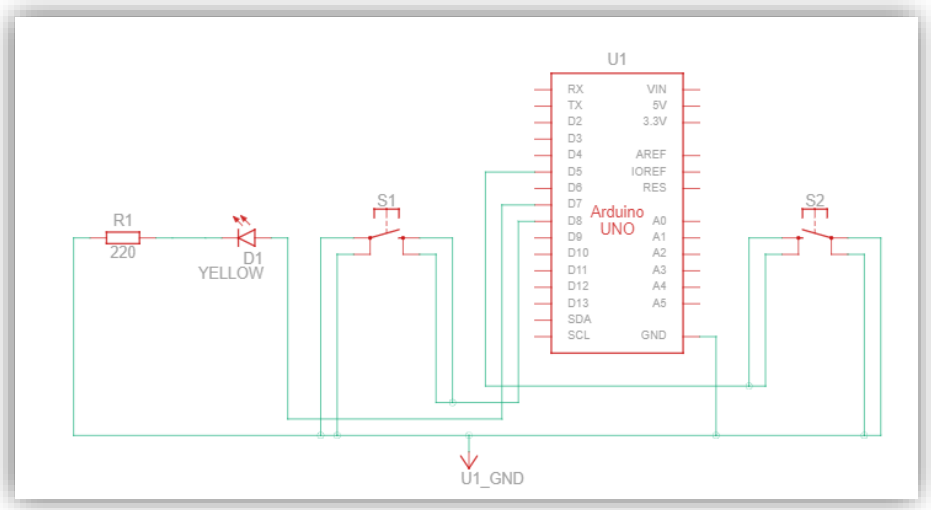
Case 2: Her får vi beskjed om at vi venter på knappetrykk fra buttonPin og at vi må trykke på buttonPin2 for å gå videre til case 3. Når buttonPin er trykt ned henter vi inn funksjonen light (). Når buttonPin2 er trykt ned kjører vi videre til case 3.

Case3: Her skal det skrives til serial monitor at yellowLed is on.

Oppkobling på brettet:



Koblingsskjema:



Programkode:

```
const int yellowLed = 7;  //yellowLed til digitalpinne 7
const int buttonPin = 8;  //buttonPin til digitalpinne 8
const int buttonPin2 = 5; //buttonPin2 til digitalpinne 9

const int trinn_1 = 1;
const int trinn_2 = 2;
const int trinn_3 = 3;

int buttonValue;          //Startverdien til testvariabelen
int buttonValuePin5;
int var = trinn_1;

void setup() {
  Serial.begin(9600);                 //Åpner serieovervåkningen
  pinMode(yellowLed, OUTPUT);         //Setter yellowLed som OUTPUT
  pinMode(buttonPin, INPUT_PULLUP);   //Setter buttonPin som INPUT_PULLUP
  pinMode(5, INPUT_PULLUP);
}

void loop()
{                                          //start void loop
  switch (var) {                           //start switch case
    case trinn_1:                          //case trinn_1
    statusRapport(1, 1);                   //Rapporterer inngående status for trinn 1

    while(digitalRead(buttonPin2)){}       //Venter på knappetrykk fra buttonPin2
    delay(500);
    statusRapport(0, 1);                   //Rapporterer utgående status for trinn 1
    var = trinn_2;
    break;

    case trinn_2:                          //case trinn_2
    statusRapport(1, 2);                   //Rapporterer inngående status for trinn 2

    while(digitalRead(buttonPin)){}        //Venter på knappetrykk fra buttonPin
    light();                               //hvis while buttonPin slår til, kalles funksjonen light

    while(digitalRead(buttonPin2)){}       //Venter på knappetrykk fra buttonPin2
    delay(500);
    statusRapport(0, 2);                   //Rapporterer utgående status for trinn 2
    var = trinn_3;
    break;

    case trinn_3:                          //case trinn_3
    statusRapport(1, 3);                   //Rapporterer inngående status for trinn 3

    while(digitalRead(buttonPin2)){}       //Venter på knappetrykk fra buttonPin2
    delay(500);
    statusRapport(0, 3);                   //Rapporterer utgående status for trinn 3
    var = trinn_1;
    break;
  }//end switch case
}//end void loop

void light(){                              //Oppretter en funksjon som heter yellowLed
  if(!buttonValue) {                       //If-setningen kjøres hvis buttonPin er trykt ned
    digitalWrite(yellowLed, HIGH);         //setter yellowLed HIGH
    }
}
```

```
62 void statusRapport(int i_0, int trinn_nr) {//start void statusRapport
63
64   if(i_0 == 1) {//start if
65     Serial.print("label: ");
66     Serial.print(trinn_nr);
67     Serial.println(" is active");
68     Serial.println(" ");
69
70       switch(trinn_nr) {//start switch case
71       case trinn_1:
72       Serial.println("machine start");
73       delay(1000);
74       Serial.println("push buttonPin2");
75       break;
76
77       case trinn_2:
78       Serial.println("push buttonPin. Then you push buttonPin2 for case 3");
79       break;
80
81       case trinn_3:
82       Serial.println("yellowLed is on");
83       break;
84     }//end if
85   }//end switch case
86 }
```

8.0 Tverrfaglig fordypningsoppgave - Iskremmaskin

I denne oppgaven skal vi lage programmet til en iskremmaskin. Ved å gjennomføre 3 lab oppgaver, skal vi til slutt i lab oppgave 4, ferdigstille hele programmet i en sekvensstyring.

Innhold

Bakgrunn

Denne oppgaven er basert på kompetansemålet (Kunnskapsdepartementet, 2020a):

- *Bygge og programmere et selvvalgt produkt som består av mikrokontroller, analoge kretser, relevante sensorer og aktuatorer for å oppnå ønsket virkemåte*

Vi skal ikke bygge et produkt i denne oppgaven, men vi skal lage et program for en iskremmaskin. Igjennom 3 lab oppgaver skal vi komme frem til lab oppgave 4, som skal være det ferdige programmet for iskremmaskinen vår. I lab oppgave 4 skal vi lage en switch-case på 3 case 'er, hvor lab oppgave 2 er case 1, lab oppgave 1 er case 2 og lab oppgave 3 er case 3.

Grunnen til at lab oppgave 1 er case 2, er fordi at den er litt enklere å gjennomføre enn lab oppgave 2, som er case 1. Så det kan være fint å starte med en litt lettere oppgave først! 😊

Dette heftet inneholder mange sider, og det ser nok ved første øyekast litt uoverkommelig ut. Men slapp av. Den inneholder en god del pensum, som skal hjelpe deg igjennom hver lab oppgave, så det er ikke så «ille» som det ser ut til.

Vi kommer til å bruke rikelig med tid på dette prosjektet, og dere vil få veiledning av faglærer underveis og hvis dere får spørsmål eller trenger enda mer forklaring, utover det som er forklart i dette heftet.

Lab oppgave 1 → 1,5 uke. Pensum: pull-down krets, while-løkke, for-løkke, tone og noTone, egendefinerte funksjoner og if-setningen. Innlevering: link til programmet på Tinkercad!

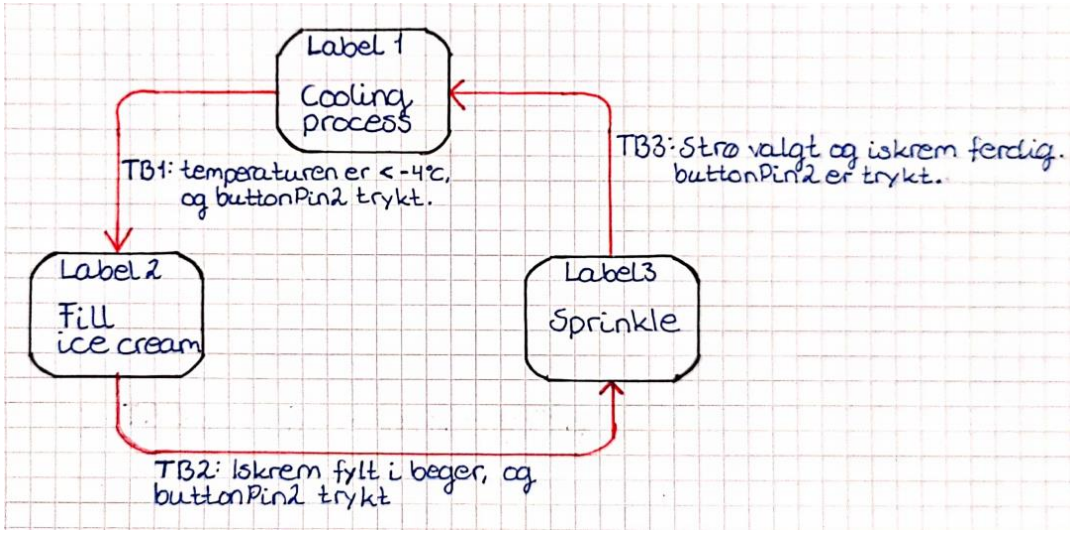
Lab oppgave 2 → 2 uker. Pensum: millis, while-løkke (fortsettelse), if- og if-else-setningen, unsigned long og egendefinerte funksjoner. Innlevering: link til programmet på Tinkercad!

Lab oppgave 3 → 2 uker. Pensum: switch-case, if-setningen, for-løkke, tone og egendefinerte funksjoner. Innlevering: link til programmet på Tinkercad!

Lab oppgave 4 → 2-3 uker. Pensum: I siste lab oppgave setter vi alt vi har brukt i de tre tidligere lab oppgavene i fokus. Innlevering: link til programmet på Tinkercad!

For hver lab oppgave finner dere relevant pensum i innledningen, fra side 9.

Prosessen



I figuren over ser vi et prosessflytskjema over prosessen som vi skal lage programmet til.

1. label1: Her må iskremen kjøles ned til -4°C før vi kan trykke på buttonPin2 som tar oss med videre til label2.
2. label2: Her skal vi fylle iskrem i begeret. Vi kan ikke gå videre før begeret er fylt, og buttonPin2 er trykt. Når dette er oppfylt, kan vi gå videre til label3.
3. label3: Her skal vi velge hvilken smak vi vil ha på strøet på iskremen. Når vi har valgt strø, og det er strødd over iskremen, er iskremen ferdig. Trykker vi ned buttonPin2 nå så starter vi på label1 igjen, og er klar for å lage en ny iskrem!

Grunnleggende om strukturen

Strukturen til arduinoprogrammet

```
1 //Her deklareres globale variabler og konstanter:
2
3
4 void setup() {
5   //Setup-koden kjører bare én gang, og den skrives her:
6   /* Her utføres alle operasjoner og innstillinger, som må være gjort før hovedprogrammet
7   kjøres */
8 }
9
10 void fillIceCream() {
```

```
11  //Her skrives hovedprogrammet, og kjøres etter void setup
12  /* Koden som skrives inne i void loop kjører om og om igjen,
13     helt til shut-down av mikrokontrolleren */
14 }
```

Setningsblokker

En setningsblokk eller en kodeblokk, består av en eller flere programsetninger gruppert sammen. Kompilatoren betrakter dem som sammenlenket, og de må starte og slutte med krøllparentes: { }.

Eksempel:

```
if(sensorValue < 500) {         //Dersom if-setningen slår til, utføres hele setningsblokken
                                //mellom { og }.
digitalWrite(redLed, HIGH);  //Etter hver programlinje er det viktig å huske semikolon ( ; ),
                             //som indikerer linjeslutt.
delay(500);
digitalWrite(redLed, LOW);
}
```

Variabler og konstanter

Det er viktig å huske at variabler og konstanter må deklareres/opprettes før vi kan bruke dem i programmet!!

En variabel er et navn som tar vare på informasjon/verdier som behandles og endres i løpet av programkjøringen. Vi gir variabler verdier ved tilordning, ved å bruke tilordningsoperatoren. F.eks.: variabel = uttrykk. Her har vi først navnet på variabelen, og uttrykket er verdien på variabelen.

Eksempel:

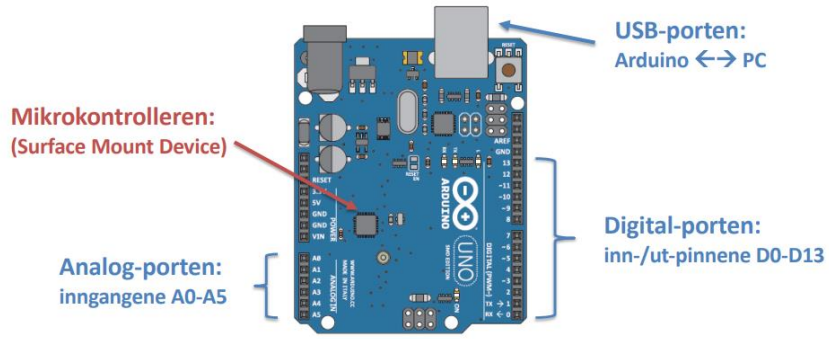
```
sensorValue = 500;
```

En konstant er et navn som tar vare på informasjon/verdier som ligger fast i løpet av programkjøringen. F.eks.: sensorPin = A2;

Variabler og konstanter kan ta forskjellige datatyper

- Heltall: int
- Desimaltall: float
- Tekststreng: String
- Bokstav/tegn: char
- Boolsk/sannhetsverdi: boolean

Arduino Uno



Viktige begreper i koding

- Programbyggeklosser
- Variabler og konstanter (deklarere, initialisere)
- Instruksjoner
- Valgstrukturer (if-else, switch case)
- Løkkestrukturer (while- og for-løkker)
- Betingelser (if-else kjøres?)
- Funksjoner

I/O-instruksjoner

- Digitale

digitalWrite(pinne, verdi)

digitalRead(pinne)

pinMode(pinne, modus) (modus = INPUT eller OUTPUT)

- Analoge

analogRead(pinne)

analogWrite(pinne, verdi)

Komponenter som kreves for å gjennomføre denne øvingen

Piezo-summer

Grønn led

Gul led

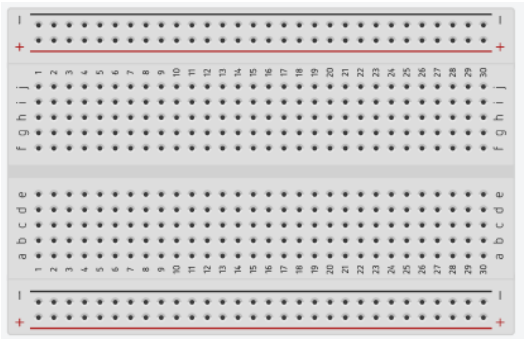
Temperatur sensor

Trykknapp

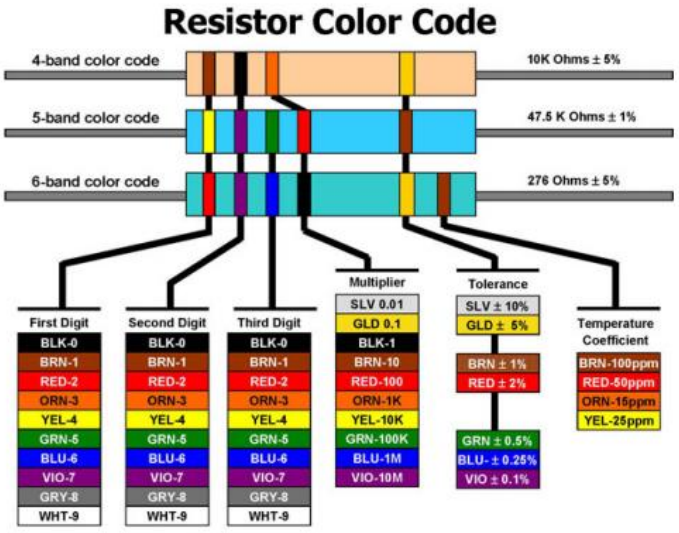
RGB-led

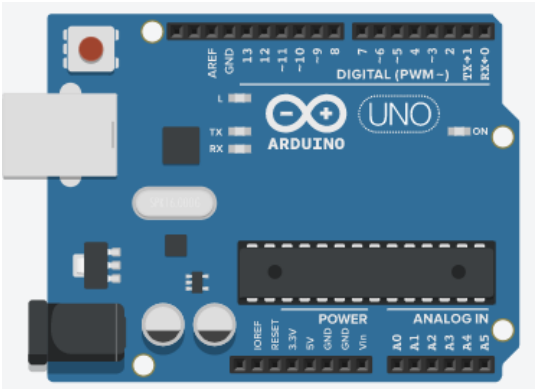
Motstand 220Ohm

Motstand 10KOhm



Koblingsbrett



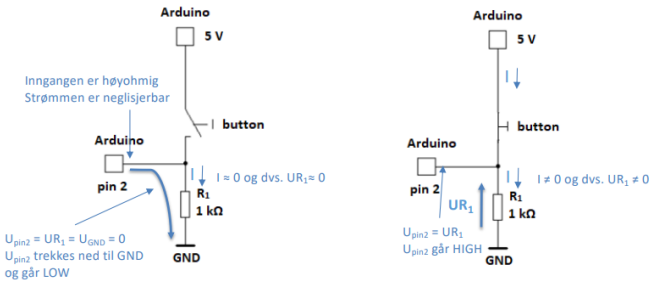


Arduino UNO

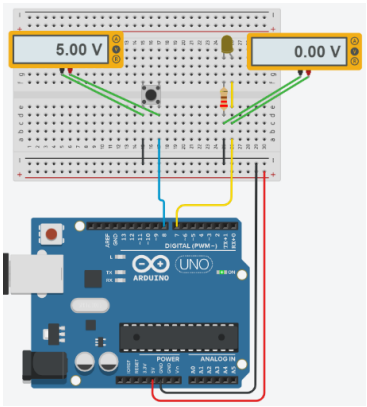
Relevant pensum til oppgavene

Pull-down krets

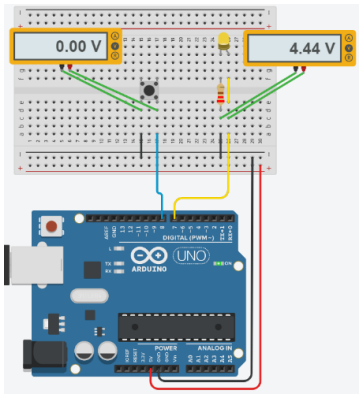
I denne oppgaven skal vi benytte oss av en PULL-DOWN krets. Da bruker vi den interne motstanden i arduino kortet, og bryteren vil legge pinnen(e) som lysdiodene er tilkoblet, til 5V. Bryteren er aktivt høy (Arduino.cc, 2021).



Eksempel:

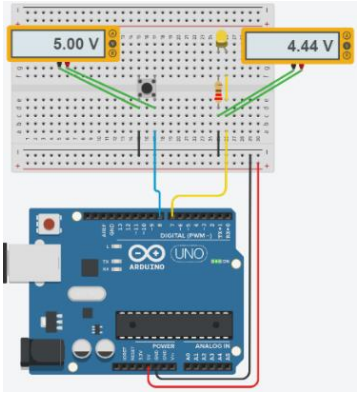


Her ser vi at bryteren har 5V når den ikke er trykket ned, altså aktivt høy.
Lysdioden har 0V, på grunn at av knappen ikke har blitt trykt ned enda.



Når knappen trykkes ned, legges bryteren lav (0V), og lysdioden får 5V og vil lyse.
Trykknappen er kun lav i et lite øyeblikk, før den går tilbake til høy (5V).

Dette bilde illustrerer hvordan spenningen er i det du trykker ned knappen. Trykknappen er lav (0V), og lysdioden har 5V.

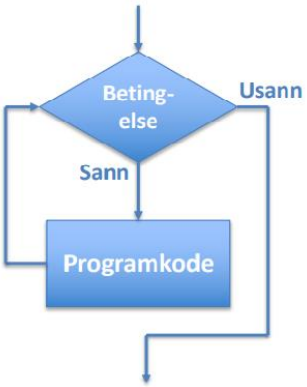


Her ser vi hvordan spenningen er etter at du har trykt på knappen. Både bryteren og lysdioden har 5V.

while-løkke()

Ved å benytte en while-løkke kan man få samme resultat som med en for-løkke, men while-løkken gir oss litt utvidet funksjonalitet. While-løkken kan brukes til å kjøre noe så lenge en knapp er trykket inn for eksempel (Arduino.cc, 2020d).

```
while(betingelse) {
    //Programkode
}
```



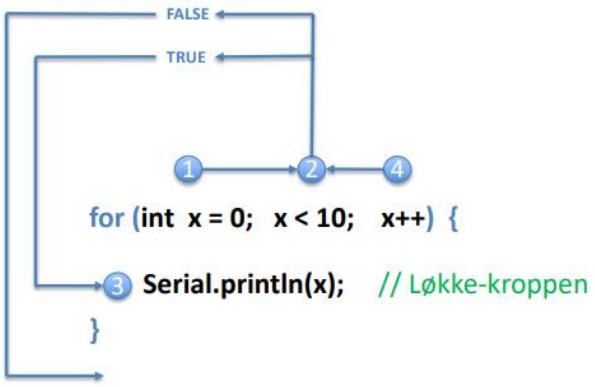
While-løkka er også en sekvens av instruksjoner som repeteres flere ganger. I denne oppgaven skal vi bruke while-løkka til å sjekke om trykknappen har blitt trykt. Så lenge trykknappen IKKE har blitt trykt ned, kjøres while-løkka om og om igjen. Når trykknappen blir trykt ned bryter vi ut av while-løkka.

for-løkke()

For-løkken (Arduino.cc, 2019c) styres av en betingelse og gjør det samme om og om igjen, helt til betingelsen ikke lenger er oppfylt. Inne i for-parentesen skriver vi inn tre ting:

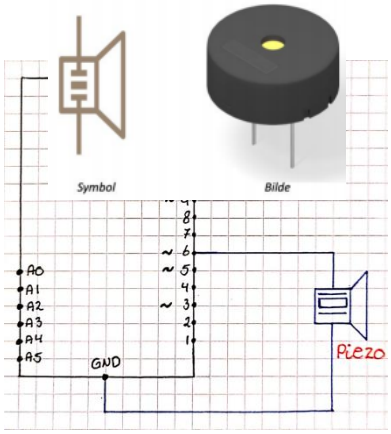
1. Deklarasjonen og initieringen av int-variabelen.
2. Her setter vi betingelsen for hvor mange ganger løkka skal kjøres
3. Det siste er hva for-løkka skal gjøre med variabelen hver gang løkken har kjørt.

```
for(initialisering; betingelse; inkrement;) {
    //Kode
}
```



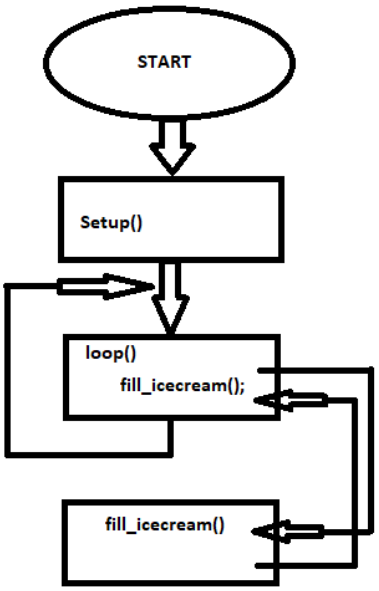
1 – Registrer startverdien på tellevariabelen
2 – Er betingelsen sann:
False = ferdig med for-løkka
True = 3
3 – Kjør løkka en gang til, og deretter 4
4 – Øk tellevariabelen med 1, og gå til 2

tone og noTone ()



Vi kan skille imellom to typer piezo-summere: analoge og digitale. De digitale summerene, som også blir kaldt aktive summere, gir en bestemt tone og styres ved å slå på spenningen av og på. Ved å bruke funksjonen tone() (Arduino.cc, 2020c) eller noTone() (Arduino.cc, 2019d) sender vi ut en ønsket tone (frekvens), for en ønsket tid. Piezo-summeren kan gi en litt høy lyd i TinkerCad, så derfor kan det være greit å sette inn en motstand på 10KΩ for å dempe lyden.

Egendefinerte funksjoner



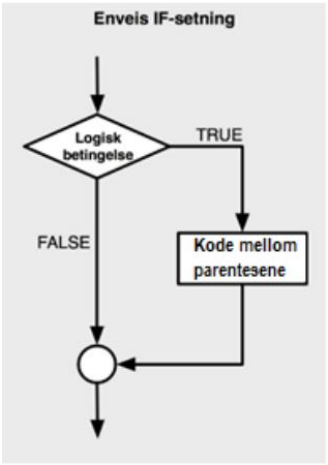
En egendefinert funksjon er greit å kunne opprette dersom samme oppgave skal utføres mange steder i programmet, og det allerede ikke eksisterer en innebygd funksjon som kan gjøre jobbe. Slik som i denne oppgaven skal vi kalle inn funksjonen lyd i hver case i switch-case'n vår. Husk at det du deklarerer inne i parameterlisten kun skal være synlig lokalt i funksjonen du lager. Det du deklarerer globalt, utenfor funksjonen, skal ikke brukes inne i funksjonen.

```
void funksjonsnavn(parameterliste) {
    //Kode
}
```

Eksempel på hvordan vi kan definere funksjonen som vi har kalt fill_icecream():

```
void fillIceCream() {   //void betyr tom, og returnerer ingen verdi når den kalles.
                        //fill_icecream er navnet vi har valgt å gi funksjonen
  //Skriv hva funksjonen skal gjøre her
}
```

if-setningen ()



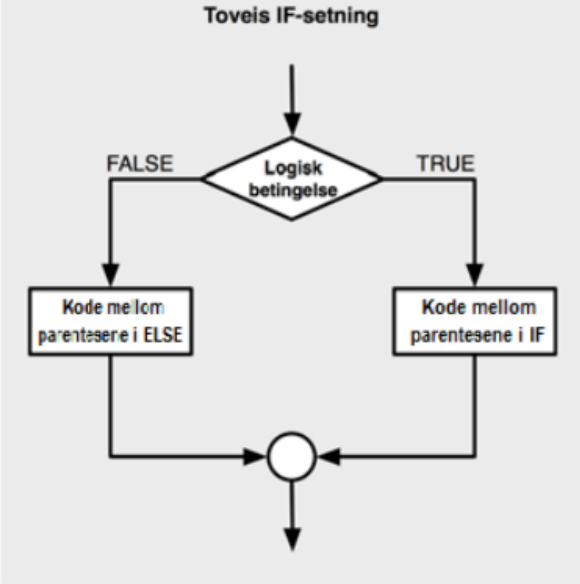
If-setningen (Arduino.cc, 2020a) lar den som programmerer bestemme hva som skal skje hvis en bestemt betingelse er oppfylt. Vi kan si det slik at if-setningen hjelper oss å ta valg i programkjøringen. Eksempler på hva som ofte sammenlignes ved hjelp av if-setninger:

- Om en int-variabel er høyere eller lavere enn en bestemt verdi
- Om en boolean-variabel er sann eller usann
- Om en char-variabel er et bestemt tegn

```
if(betingelse) {          //Betingelsen er en logisk betingelse
    //Kode                //Her kan vi ha en eller flere programlinjer
}
```

if-else-setningen ()

Virkemåte:



Uansett om betingelsen vår er sann eller usann, vil en av handlingene (programkodene) utføres.

For eksempel kan vi si at hvis trykknappen er trykt ned, er betingelsen vår sann, og lysdioden lyser. Hvis den ikke er trykt ned, er betingelsen vår usann, og lysdioden lyser ikke (Arduino.cc, 2020a) (Arduino.cc, 2019b).

```
if (betingelse 1)
{
     // programkode 1
}

else if (betingelse 2)
{
     // programkode 2
}

  -
  -                          } Kan være multiple else if
  -

else
{
     // programkode N
}
```

millis()

For å starte en tidtaking bruker vi millis() (Arduino.cc, 2022a) funksjonen, og prinsippet for tidtaking med millis() er å sjekke om det har gått en viss tid (timingPeriod). Det første vi må gjøre er å etablere en referansevariabel som har startverdien startMillis = millis();. Deretter fanger vi den løpende verdien til millis(), og den blir stadig større enn startMillis'n vår, fordi tiden (millisekundene) har løpt siden vi fanget millis() verdien for den, runningMillis = millis();. Etter det må vi teste om avstanden mellom startMillis og nyeste oppdatering av runningMillis har blitt større eller lik timePeriod. Det er viktig å huske at vi må legge et slikt oppsett til en løkkestruktur, enten void loop eller while, slik at vi får repeterende oppdatering av runningMillis og testingen i if-setningen.

f.eks.:

```
if((runningMillis – startMillis) >= (timingPeriod)) {
        //kode
}
```

String()

String (Arduino.cc, 2019h) er en tekststreng og kan vises på to forskjellige måter, og brukes til å lagre tekst. Teksten kan vises på en LCD eller i arduino Serial monitor-vinduet. String er en sammensatt datatype, med tilhørende funksjoner. Den har flere char på rekke og rad, og må angis med doble hermetegn.

bool

bool (Arduino.cc, 2019a) er en variabel som bare kan lagre én av to verdier, sann eller usann. Disse to verdiene kan enten skrives som true eller false, eller 1 (true) og 0 (false). Å bruke en slik variabel er velegnet når vi skal sammenligne noe. Som i denne lab oppgaven skriver vi:

```
bool timeLapsed = false; //her deklarerer timeLapsed til å være false.
```

switch-case ()

Hvis man skal ha mange ulike alternativer blir det raskt uoversiktlig å bruke if- og else-setningen, derfor kan det i mange sammenhenger være mer praktisk å bruke en switch-case (Arduino.cc, 2019i). Switch-case'n baserer seg på en variabel, og kan utføre mange forskjellige instruksjoner avhengig av hvilken verdi variabelen har.

En switch-case styrer programflyten basert på betingelser:

```
switch (var) {    //Nøkkelordet er switch, og argumentet = testvariabelen
case label1:      //Tester om label1 er lik verdien til testvariabelen
    //Kode        //Hvis verdien er lik testvariabelen, kjøres denne koden
break;            //Deretter bryter vi ut av switch-case setningen
}
```

Vi bruker break for å hoppe ut fra løkker eller if og switch-case'n, og du kan ha så mange case'r du måtte ønske i switch-case setningen.

Serial biblioteket

Serial.begin() (Arduino.cc, 2020b): Setter opp seriell kommunikasjon mellom Arduino og PC.

Serial.print() (Arduino.cc, 2022b): Skriver tekst, verdier, eller variablers verdier til Seriemonitoren.

Serial.println() (Arduino.cc, 2019f): Samme som Serial.print(), men utfører linjeskift på slutten. "println" uttales som om det sto printline (på engelsk).

Serial.available() (Arduino.cc, 2019e): Gir deg ant. tegn (byte) som er mottatt i Arduinoens seriell-buffer

Serial.read() (Arduino.cc, 2019g): Gir deg det første tilgjengelige av innkomne tegn. Gir deg -1 hvis seriell-bufferet er tomt.

Lab oppgave 1 – egendefinerte funksjoner, LED og piezo-summer

I denne øvingenes første lab oppgave skal vi få et led-lys til å lyse i 5 sekunder når knappen har blitt trykt ned, ved å lage en funksjon som vi skal kalle fillIceCream(). Etter 5 sekunder skal lyset slukkes. Vi blir også nødt til å lage en trudelutt med en buzzer i det lyset starter og lyse, og en trudelutt når lyset slukker. For å få til dette skal vi benytte oss av ei for-løkke inne i funksjonen fillIceCream().

I selve void loop'en skal vi bruke en while-løkke som skal vente på knappetrykk fra trykknappen, buttonPin. Når den blir trykt, hopper vi videre i programmet og funksjonen void fillIceCream() kjøres.

Når knappen er trykt ned, og funksjonen void fillIceCream() kjøres, skal det i det lyset starter å lyse skrives i serial monitor; «Filling up ice cream». Når lyset slukkes, skal det skrives; «Filling up ice cream finished» i serial monitor. Dette for at vi hele tiden skal kunne ha kontroll på hva som skjer, ved å lese det som skrives i serial monitor.

Alt vi lager i denne lab oppgaven skal vi bruke igjen senere i lab oppgave 4, hvor alle de 3 lab oppgavene du skal igjennom skal sammensettes til et program.

Arbeidsoppdrag:

1. Først må du opprette en ny oppgave i TinkerCad, som du kaller for lab oppgave 1.
2. Koble opp komponentene på utstyrsbrettet etter koblingsskjemaet, og koble det sammen med arduinokortet til rett pinne. HUSK! 5V og GND til koblingsbrettet.
3. Deklarer globale variabler og konstanter i headeren:
    - gult lys = yellowLed
    - trykknapp = buttonPin
    - piezo-summer = buzzerPin
    - buttonValue som forteller oss noe om startverdien til tellevariabelen
4. void setup: Her skriver du setup-koden som kun kjøres en gang. Alle operasjoner og innstillinger som må være gjort før hovedprogrammet kjøres, utføres her.

    Eksempel:

    ```
    void setup() {
    Serial.begin(9600);          //setter opp seriell kommunikasjon mellom arduino og PC
    pinMode(pinne, modus);   //modus = INPUT eller OUTPUT
    ```

```
}
```

5. void loop: Her kommer hovedprogrammet, og det kjøres etter void setup. Inne i krøllparentesene i void loop skal vi bruke en while-løkke som sjekker for knappetrykk, og funksjonen fillIceCream().

   Funksjonen fillIceCream() kommer utenfor krøllparentesene til void loop. Inne i funksjonen vår må vi bruke en if-setning for å sjekke om knappen er trykt ned eller ikke, og vi må ha 2 for-løkker. Den første for-løkken er får trudelutten når lyset starter å lyse, og den 2. for-løkken er når lyset slukker etter 5 sekunder.

   Hvis if-setningen slår til, det vil si knappen er trykt ned, skal gul led lyse i 5 sekunder og vi får en liten trudelutt fra buzzeren. Når knappen er trykt, skal vi også skrive til monitor: «Filling up ice cream».

   Når 5 sekunder har gått skal vi ha en ny trudelutt, og vi skal skrive til serie monitor: «Filling up ice cream finished» og lyset slukker.

```
void setup() {
//Setup-koden kjører bare én gang, og den skrives her:
/* Her utføres alle operasjoner og innstillinger, som må være gjort før
hovedprogrammet
    kjøres */
/* Vi skal også skrive til serial monitor at iskremmaskinen starter opp,
   og det skal skrives bare en gang ved oppstart! */
}


void loop() {
while(betingelse);       //venter på knappetrykk fra buttonPin
fillIceCream();          //Hvis buttonPin blir trykt ned, kjøres funksjonen fill_icecream()
}
void fillIceCream() {    //lager en egendefinert funksjon
if(betingelse) {    // HUSK! Betingelsen vår er å sjekke om knappen er trykt ned
   //Kode           //her skal vi skrive til serial monitor, og gult lys skal starte å lyse
for( initialisering; betingelse; inkrement;) { //Her skal vi skrive programmet for
                                               //trudelutten. Husk at den skal gå fra en
                                               //mørk til en lysere tone
tone(pinne, betingelse);       //skriver inn pinne og betingelsen i for-løkken
```

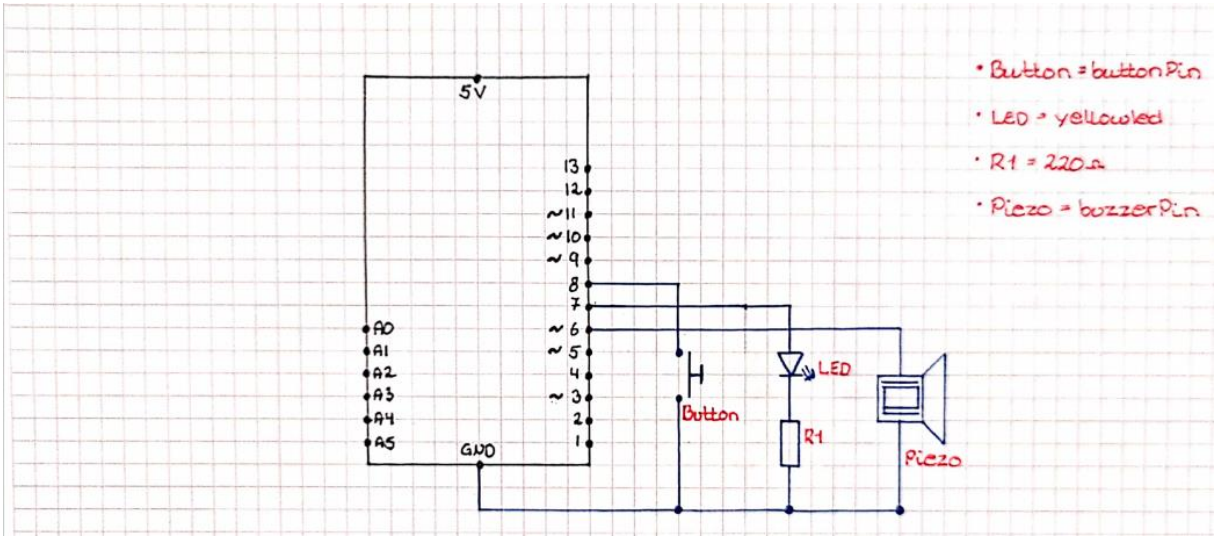
```
delay();                    //legger inn en liten pause f.eks. 50ms.
noTone(pinne);              //skriver inn pinnen til buzzeren
}

delay();                    // den tiden gult lys skal lyse
Serial.print();             //skriver inn det som skal skrives i serial monitor når lyset
                            //starter å lyse
digitalWrite(pinne, modus);  //Hva skulle skje med gult lys når knappen var trykt ned?

for(initiallisering; betingelse; inkrement;) {     //Her skal vi skrive programmet for
                                                   //trudelutten. Husk at den skal gå fra en
                                                   //lys til en mørkere tone
tone(pinne, betingelse);        //skriver inn pinne og betingelsen i for-løkka
delay();                    //legger inn en liten pause f.eks. 50 ms.
 noTone(pinne);             //skriver inn pinnen til buzzeren
  }
 }
}
```

Utstyrsliste

For å gjennomføre lab oppgave 1 trenger du følgende:

- Trykknapp
- Gul led
- 1x Motstand (220Ω)
- Buzzer (summer)
- 1x Motstand (10KΩ) – for å bruke på buzzeren, slik at vi får en mer komfortabel lyd
- Koblingsbrett
- Arduino kortet

Koblingsskjema



Lab oppgave 2 – Nedkjøling av iskremen ved hjelp av millis()

I lab oppgave 2 skal vi opprette en egendefinert funksjon som skal hete startCooling. Formålet med denne funksjonen er at vi skal kjøle ned iskremen til -4°C, før vi kan fylle iskremen i begeret.

Selve nedkjølingsprosessen tar 30 sekunder, og den skal vi løse ved å bruke millis funksjonen. Vi skal lage programmet slik at vi starter på 0, og teller oss opp til 30 sekunder, ved at vi legger til 10 sekunder for hver telling.

Arbeidsoppdrag:

1. Starter med å deklarere globale variabler og konstanter i headeren
   →hva skal deklareres: greenLed, tempPin, buttonPin og idealTemp.
   idealTemp deklarerer vi til -4, ettersom det er den temperaturen vi vil iskremen skal ha.
2. void setup: her skriver vi koden som bare kjører én gang, og som må være gjort før hovedprogrammet kjøres.

```
void setup() {
//Setup-koden kjører bare én gang, og den skrives her:
/* Her utføres alle operasjoner og innstillinger, som må være gjort før hovedprogrammet
```

```
    kjøres */
  /* Vi skal også skrive til serial monitor at iskremmaskinen starter opp,
     og det skal skrives bare en gang ved oppstart! */
  }
```

3. void loop: Her skriver vi hovedprogrammet som kjøres etter void setup. Det programmet som skrives her kjører om og om igjen, helt til vi skrur av arduinoen.

```
void loop() {
startCooling(betingelse);  //husk at her skal betingelsen være det vi deklarerte globalt!
                           //bruk f.eks. idealTemp, som vi deklarerte globalt
}

bool startCooling(betingelse) {  //oppretter en funksjon vi kaller startCooling.
                                 //betingelsen skal være lokalt inne i funksjonen, så bruk f.eks.
                                 //prodTemp, som er produksjonstemperaturen vår, -4℃
Serial.print();              //her skal vi skrive til serial monitor at kjølingen har startet.
                             //30 sekunder til:
Serial.print(betingelse); //betingelsen vår må være det vi deklarerte globalt, for det er
                             //den temperaturen vi vil ha: prodTemp, altså -4℃
Serial.println();            //her skriver vi til serial monitor degrees, for å få med grader bak -4℃
Serial.println();            //her skal vi legge inn et opperom ved å skrive: « » til serial monitor

unsigned long startMillis = millis();  // etablerer en referansevariabel med startverdi
bool timeLapsed = false;                //deklarerer timeLapsed til false
int sek = 0;                            //starter tellingen på 0
String txt;                             //Bruker String for å lagre tekst

//Benytter en while-løkke for å ta tiden
while(!(timeLapsed)) {   //her må vi huske !, fordi vi har satt timeLapsed til å være false
                         //
```

UTFORDRING:

Herfra vil jeg at du skal prøve mer på egenhånd, med litt mindre hjelp i kodene. Starten på koden står her, men du må fullføre hver linje! Se på kommentarene bak hver linje, der står det beskrevet hva som skal skje.

```
unsigned long runningMillis = millis(); //fanger den løpende verdien for millis()
if(betingelse) {                        //Nedkjølingen tar 30 sekunder
kode;                                   //Det har gått 10 sekunder
kode;                                   //starter timingen på nytt
Serial.println(txt);
}
if(betingelse) {                        //sjekker om det har gått 30 sekunder eller mer
Kode;                                   //kjøleprosessen har gitt -4℃ . HINT: timeLapsed
Kode;                                   //Setter pinne 9, greenLed HIGH
Kode;                                   //skriver til serial monitor: Temperature reach -4 degrees
Kode;                                   //opperom
Kode;                                   //skriver til serial monitor: Ready for label 2
}
else if(betingelse) { Kode; }   //sjekker om det har gått 20 sekunder eller mer
                                //skriver til serial monitor: cooling – temperature has reached
                                // -4 ℃
                                //HUSK: txt -strengen som vi opprettet
else if(betingelse) { kode; }   //sjekker om tellingen er mindre enn 20 sekunder,
                                //men større enn 10 sekunder
                                //skriver til serial monitor: cooling-temperature has reached
                                // -2 ℃
                                //HUSK: txt-strengen som vi opprettet
else { kode; }                  //Starten av kjøleprosessen
                                //hvis ingen av de tidligere else if 'ene slår til, slår denne til
                                //koden her skal skrive cooling-temperature has reached 0 ℃
 }
```

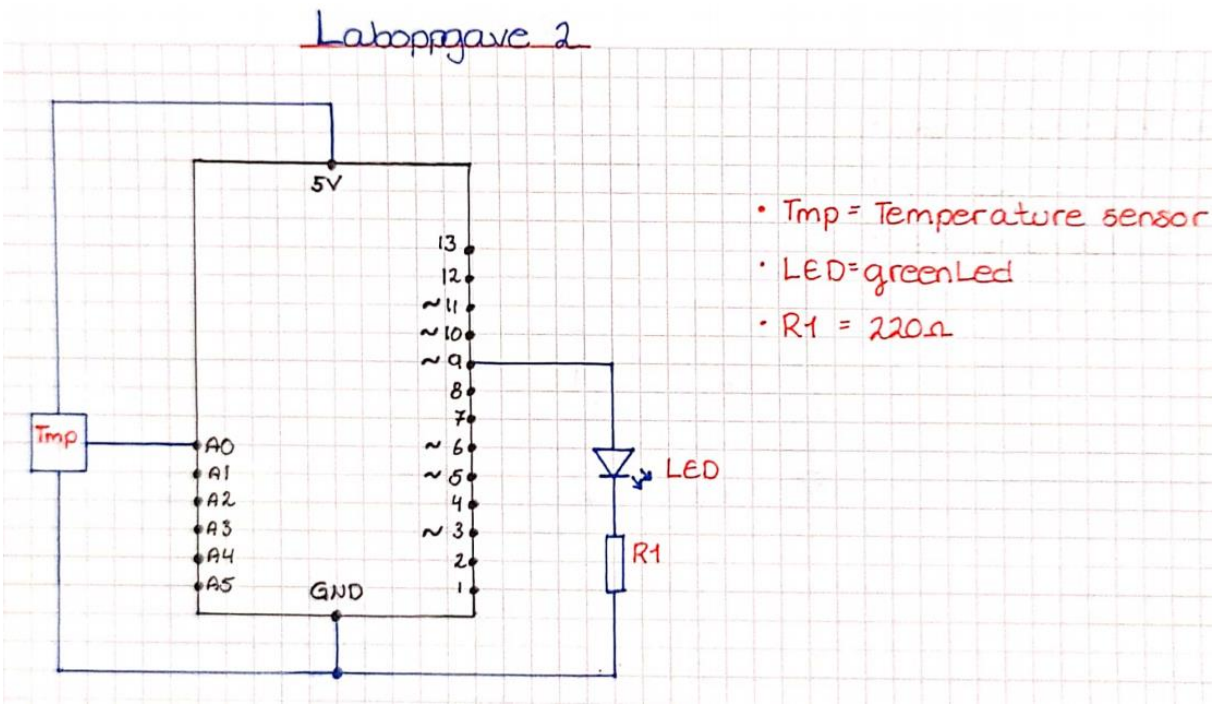
```
}
```

Utstyrsliste

For å gjennomføre lab oppgave 3 trenger du følgende:

- Grønn led
- 1x Motstand (220Ω)
- Temperatur sensor (TMP36)
- Koblingsbrett
- Arduino kortet

Koblingsskjema



Lab oppgave 3 – switch-case med RGB

I denne lab oppgaven skal vi lage en egendefinert funksjon med en switch-case ved å benytte en RGB-diode, som skal simulere hvilket strø man kan velge på iskremen. Man skal kunne velge imellom 3 forskjellige sorter strø, jordbær,- tutti-frutti og blåbærsmak.

For å gjøre det enklest mulig for de som arbeider der skal vi lage en switch-case hvor vi kan skrive inn en bokstav i serial monitor, som senere skal gi oss det strøet som vi har valgt.

- R = Strawberry → Rødt lys som blinker 5 ganger og bipper på hvert blink med en buzzer.
- G = Tutti-frutti → Grønt lys som blinker 5 ganger og bipper på hvert blink med en buzzer.
- B = Blueberry → Blått lys som blinker 5 ganger og bipper på hvert blink med en buzzer.

Arbeidsoppdrag:

Når man starter programmet, skal det komme opp en meny, over hvilket strø vi kan velge imellom. Menyen skal komme opp i serial monitor: «Menu: R=Strawberry, G=Tutti-frutti, B=Blueberry». Når den som betjener maskinen skriver inn for eksempel R, skal serial monitor skrive «Strawberry».

Vi blir nødt til å lage 3 case'r som skal hete R, G og B, og vi må lage det slik at vi har en if-setning med Serial.available i void loop'en, som skal «se/overvåke» etter om noen av case 'ene slår til. Seriel available'n vår er satt til 0, så når vi skriver en av de tre kommandoene, vil den overvåke at en av casen 'ene har slått til og case'n utføres.

Vi skal opprette to egendefinerte funksjoner i dette programmet, som skal brukes senere i lab oppgave 4:

- Funksjon 1: void sprinkle();
- Funksjon 2: void sound(int varighet);

Litt starthjelp med koden i void loop!

```
void setup() {
//Setup-koden kjører bare én gang, og den skrives her:
/* Her utføres alle operasjoner og innstillinger, som må være gjort før hovedprogrammet
    kjøres */
/* Vi skal også skrive til serial monitor at iskremmaskinen starter opp,
   og det skal skrives bare en gang ved oppstart! */
}


void loop() {
//henter inn den egendefinerte funksjonen sprinkle
}

void sprinkle() {    //oppretter en egendefinert funksjon, som vi kaller sprinKle
Kode;                //her serial monitor skrive menyen over strøet vi kan velge imellom
if(betingelse) {     //her skal vi sjekke om Serial.available er større enn 0
Kode;               //deklarerer inbyte hvor Serial.read leser innkommende data

switch(betingelse) {   //her starter switch-case 'en vår. Husk å skrive inn betingelsen
case1:                  //her kommer vår første case, som vi har kalt R
kode;             //her skal serial monitor skrive Strawberry
                  //og RGB 'en skal blinke med rødt lys 5 ganger
                  //vi må også kall inn funksjonen sound, som skal bippe hver gang det blinker
}
break;            //bryter ut av switch-case setningen

case2:           //her kommer vår andre case
```

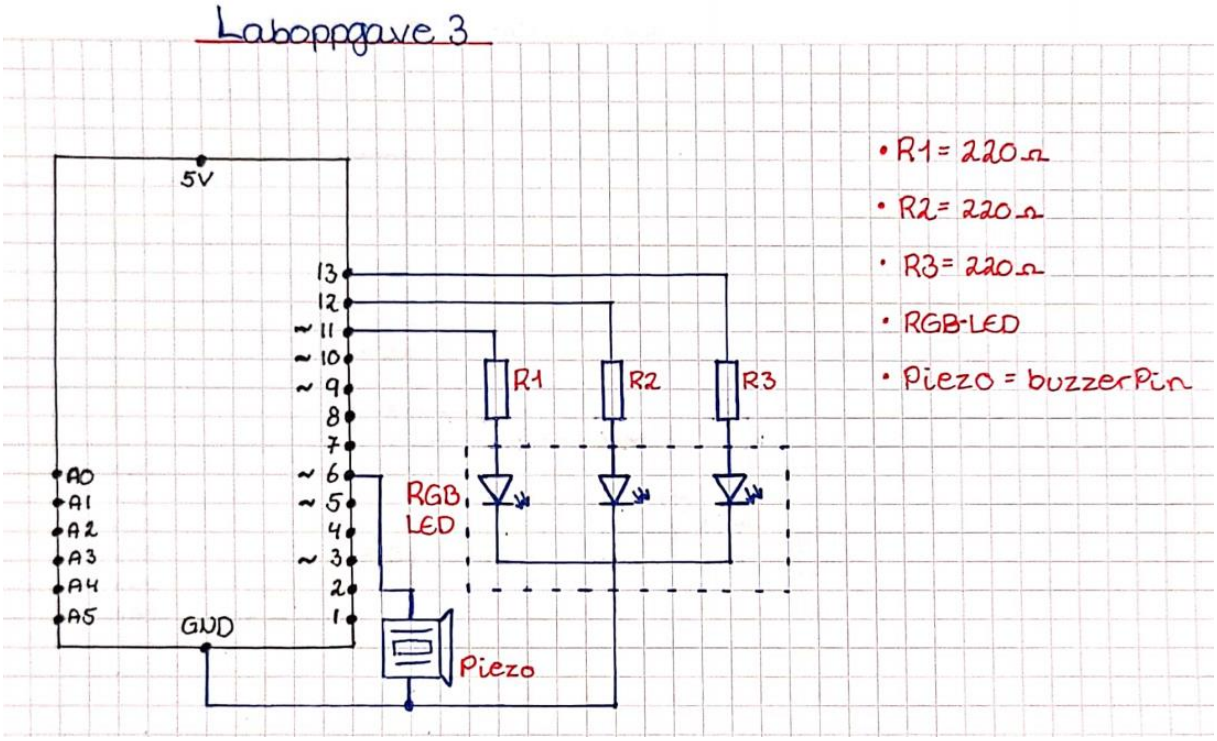
```
Kode;              //det samme som i case 1, bare at nå skal grønt lys blinke,
                   //og serial monitor skal skrive tutti-frutti
}
break;           //bryter ut av switch-case setningen

case3:           //her kommer vår tredje case
Kode;            //samme som i case 1 og 2, bare at nå skal blått lys blinke,
                 //og serial monitor skal skrive Blueberry
}
break;           //bryter ut av switch-case setningen
   }
  }
}
void sound() {   //oppretter en egendefinert funksjon som vi kaller sound
tone(pinne, frekvens, varighet);   //her skriver vi bippet som vi skal ha i hver case
                                   //HUSK: pinne nummeret for buzzeren, frekvensen f.eks.
                                   //1500, og skrive inn betingelsen til funksjonen
}
```

Utstyrsliste

For å gjennomføre denne oppgaven trenger du følgende:

- 3x Motstand (220Ω)
- 1x motstand (10KΩ)
- RGB-LED
- Buzzer (summer)
- Koblingsbrett
- Arduino kortet

Koblingsskjema



Lab oppgave 4 – Ferdig program for iskremmaskinen

I fjerde og siste lab oppgave skal vi sy sammen all lab oppgavene til en oppgave, hvor vi vil få hele prosessen i en helhet. Dere skal nå være godt forberedt i og med at vi skal bruke alle kodene som vi allerede har laget.

Her skal vi lage en egendefinerte funksjoner i tillegg til de vi har utarbeidet i de tidligere oppgavene. Vi må ha en funksjon som sjekker inngående og utgående status for hvert trinn i switch case 'en, og denne funksjonen skal vi lage i en switch case.

Arbeidsoppdrag:

Sett sammen de tidligere oppgavene. Vi må opprette en switch case inne i void loop, som skal kjøre hovedprogrammet vårt. De funksjonene vi har opprettet i de andre lab oppgavene skal vi bruke i hovedprogrammet, men de skal kun kalles opp. Selve koden må vi skrive utenfor void loop. Som litt ekstra hjelp har jeg skrevet strukturen til programmet!

Strukturen på programmet:

```
//Før void setup må vi deklarere globale variabler og konstanter.

void setup() {
//Setup-koden kjører bare én gang, og den skrives her:
/* Her utføres alle operasjoner og innstillinger, som må være gjort før hovedprogrammet
    kjøres */
/* Vi skal også skrive til serial monitor at iskremmaskinen starter opp,
   og det skal skrives bare en gang ved oppstart! */
}
/* switch-case 'en for hovedprogrammet er skrevet inne i void loop. Husk at funksjoner du
lager selv skal skrives utenom void loop. De funksjonene vi definerer selv utenom void loop,
er de vi skal kalle inn i switch case 'en som er i void loop. */


void loop() {
switch(betingelse) {
case label1:
// Her skal koden for trinn 1 skrives:
```

```
/* I case 1 skal prosessen for nedkjølingen av iskremen finne sted. Vi må også legg inn en trykknapp ekstra som vi skal betjene for å gå videre til trinn 2. Vi skal benytte en while-løkke for å sjekke om buttonPin2 blir betjent.*/

statusReport(betingelse);      //Rapporterer inngående status for trinn 1

startCooling(betingelse);      //funksjonen for nedkjølingen av iskremen


while(betingelse) {}           //Venter på knappetrykk fra buttonPin2, for å bytte til neste case

delay();                       //f.eks. 500 ms

statusReport(betingelse);      //Rapporterer utgående status for trinn 1

var = label2;

break;

case label2:

// Her skal koden for trinn 2 skrives:

/* I case 2 skal vi fylle iskrem i begeret. Vi må benytte en while-løkke som venter på knappetrykk fra buttonPin. Når den trykkes ned, bryter vi ut av while-løkka og kjører funksjonen fillIcecream. Når vi har kjørt ferdig fillIcecream har vi ei ny while-løkke som venter på at buttonPin2 blir trykt, slik at vi kan gå videre til trinn 3! */

statusReport(betingelse);      //Rapporterer inngående status for trinn 2


while(betingelse);             //funksjonen for nedkjølingen av iskremen

fillIceCream();                //funksjonen fillIcecream


while(betingelse) {}           //Venter på knappetrykk fra buttonPin2, for å bytte til neste case

delay();                       //f.eks. 500 ms

statusReport(betingelse);      //Rapporterer utgående status for trinn 2

var = label3;

break;

case label3:

// Her skal koden for trinn 3 skrives:

/* I case 3 skal vi bruke funksjonen som gjorde det mulig for oss å velge hvilket strø vi ville ha på iskremen vår. Kaller inn funksjonen sparKle, og når den er kjørt venter while-løkka på at buttonPin2 blir betjent, slik at vi kan gå videre til start igjen! */

statusReport(betingelse);      //Rapporterer inngående status for trinn 3
```

```
sprinkle();                    //Her kjøres funksjonen sparKle

while(betingelse) {}           //Venter på knappetrykk fra buttonPin2, for å bytte til neste case
delay();                       //f.eks. 500 ms
statusReport(betingelse);      //Rapporterer utgående status for trinn 3
var = label1;
break;
  }
}
/* Etter void lopp starter vi med å skrive inn programmet for de funksjonene vi har opprettet
i de tidligere lab oppgavene, samt at vi skal opprette en ny switch case, som skal rapportere
inngående og utgående status for hvert trinn. */
bool startCooling(betingelse) {
      kode;
}

void fillIcecream() {
     kode;
}

void sprinkle() {
     kode;
}

// Her kommer den nye funksjonen void statusReport()
 void statusReport(betingelse) {  // Betingelse: vi ønsker at den skal sjekke for inngående,
                                  // og utgående status for hver case.
                                  //den skal også sjekke hvilken case som slår til
/* Vi skal benytte oss av en if-else-setning for å sjekke om det er for inngående eller
utgående status. Hvis den er sann (1) skal den sjekke status for inngående. Er den usann (0),
skal den sjekke for utgående. Nedenfor er koden ferdig skrevet, men i feil rekkefølge. Klarer
du å endre det slik at programlinjene kommer i rett rekkefølge? */
if(betingelse) {i_0 == 1) {
```

```
Serial.print(labelNr);
Serial.println(« is active»);
Serial.println(«label: »);
Serial.println(« »);

switch(labelNr) {
case label2:
break;
case label1:
break;
case label3:
Serial.println(«Menu: R=Strawberry, G=Tutti-frutti, B=Blueberry»);
break;
  }
}

else {
Serial.println(« »);
Serial.print(«label: »);
Serial.println(« finished»);
Serial.print(labelNr»);

switch(labelNr) {
case label3:
break;
case label1:
digitalWrite(pinne, modus);
break;
case label2:
digitalWrite(pinne, modus);
break;
```

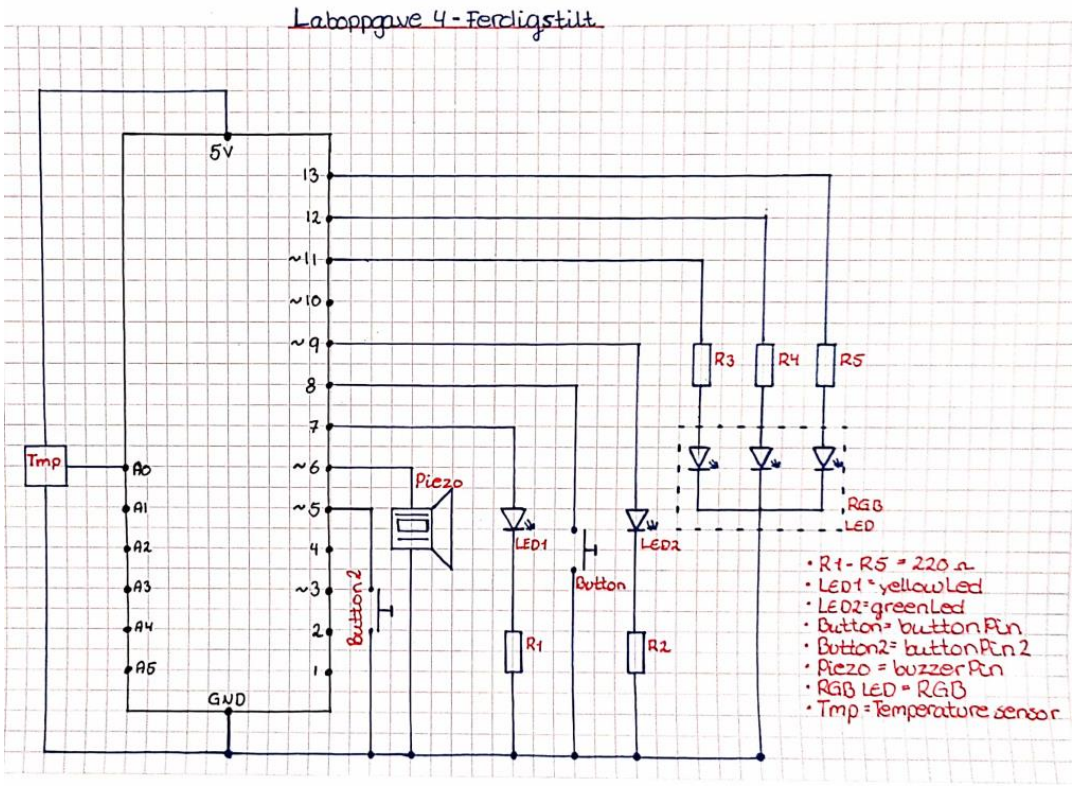
```
    }
  }
}
```

Utstyrsliste

For å gjennomføre denne oppgaven trenger du følgende:

- 5x Motstand (220Ω)
- 1x motstand (10KΩ) (for å bruke på buzzeren hvis man vil)
- Gul led
- Grønn led
- 2x trykknapp (push button)
- Temperatur sensor (TMP36)
- RGB-LED
- Buzzer (summer)
- Koblingsbrett
- Arduino kortet

Koblingsskjema



Referanser

Arduino.cc. (2019a). *bool*. Arduino.cc. https://www.arduino.cc/reference/en/language/variables/data-types/bool/

Arduino.cc. (2019b). *else*. Arduino.cc. https://www.arduino.cc/reference/en/language/structure/control-structure/else/

Arduino.cc. (2019c). *for*. Arduino.cc. https://www.arduino.cc/reference/en/language/structure/control-structure/for/

Arduino.cc. (2019d). *noTone()*. Arduino.cc. https://www.arduino.cc/reference/en/language/functions/advanced-io/notone/

Arduino.cc. (2019e). *Serial.available()*. Arduino.cc. https://www.arduino.cc/reference/en/language/functions/communication/serial/available/

Arduino.cc. (2019f). *Serial.println()*. Arduino.cc. https://www.arduino.cc/reference/en/language/functions/communication/serial/println/

Arduino.cc. (2019g). *Serial.read()*. Arduino.cc. https://www.arduino.cc/reference/en/language/functions/communication/serial/read/

Arduino.cc. (2019h). *String*. Arduino.cc. https://www.arduino.cc/reference/en/language/variables/data-types/string/

Arduino.cc. (2019i). *switch...case*. Arduino.cc. https://www.arduino.cc/reference/en/language/structure/control-structure/switchcase/

Arduino.cc. (2020a). *if*. Arduino.cc. https://www.arduino.cc/reference/en/language/structure/control-structure/if/

Arduino.cc. (2020b). *Serial.begin()*. Arduino.cc. https://www.arduino.cc/reference/en/language/functions/communication/serial/begin/

Arduino.cc. (2020c). *tone()*. Arduino.cc. https://www.arduino.cc/reference/en/language/functions/advanced-io/tone/

Arduino.cc. (2020d). *while*. Arduino.cc. https://www.arduino.cc/reference/en/language/structure/control-structure/while/

Arduino.cc. (2021). *constants*. Arduino.cc. https://www.arduino.cc/reference/en/language/variables/constants/constants/

Arduino.cc. (2022a). *millis()*.
https://www.arduino.cc/reference/en/language/functions/time/millis/

Arduino.cc. (2022b). *Serial.print()*. Arduino.cc.
https://www.arduino.cc/reference/en/language/functions/communication/serial/print/

Almås, S. & Rossen, E. (2021, 8 Februar 2021). *C++*. Store Norske Leksikon. https://snl.no/C+

Barragàn, H. (2016). *The Untold History of Arduino*. Hentet 24.04.22 fra
https://arduinohistory.github.io/

Keyestudio. (2021). *KS0085 Keyestudios Smart Home Kit For Arduino*. Keyestudio.
https://wiki.keyestudio.com/KS0085_Keyestudio_Smart_Home_Kit_for_Arduino

Kunnskapsdepartementet. (2020a). *Kompetansemål etter elektroniske kretser og nettverk*.
Udir. https://www.udir.no/lk20/ele01-03/kompetansemaal-og-vurdering/kv254?Kjerneelementer=true&lang=nob

Kunnskapsdepartementet. (2020b). *Kompetansemål etter konstruksjons- og styreteknikk*.
Udir. https://www.udir.no/lk20/tip01-03/kompetansemaal-og-vurdering/kv252

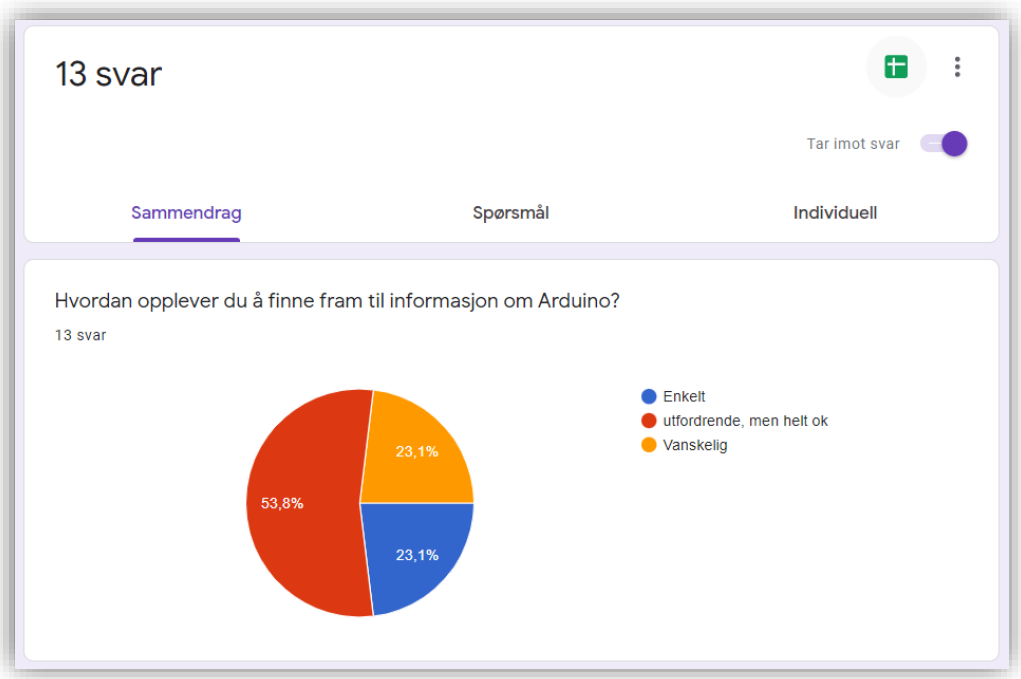
Tinkercad. (u.å.). *AUTODESK*
Tinkercad. Tinkercad. https://www-tinkercad-com.translate.goog/?_x_tr_sl=en&_x_tr_tl=no&_x_tr_hl=no&_x_tr_pto=sc

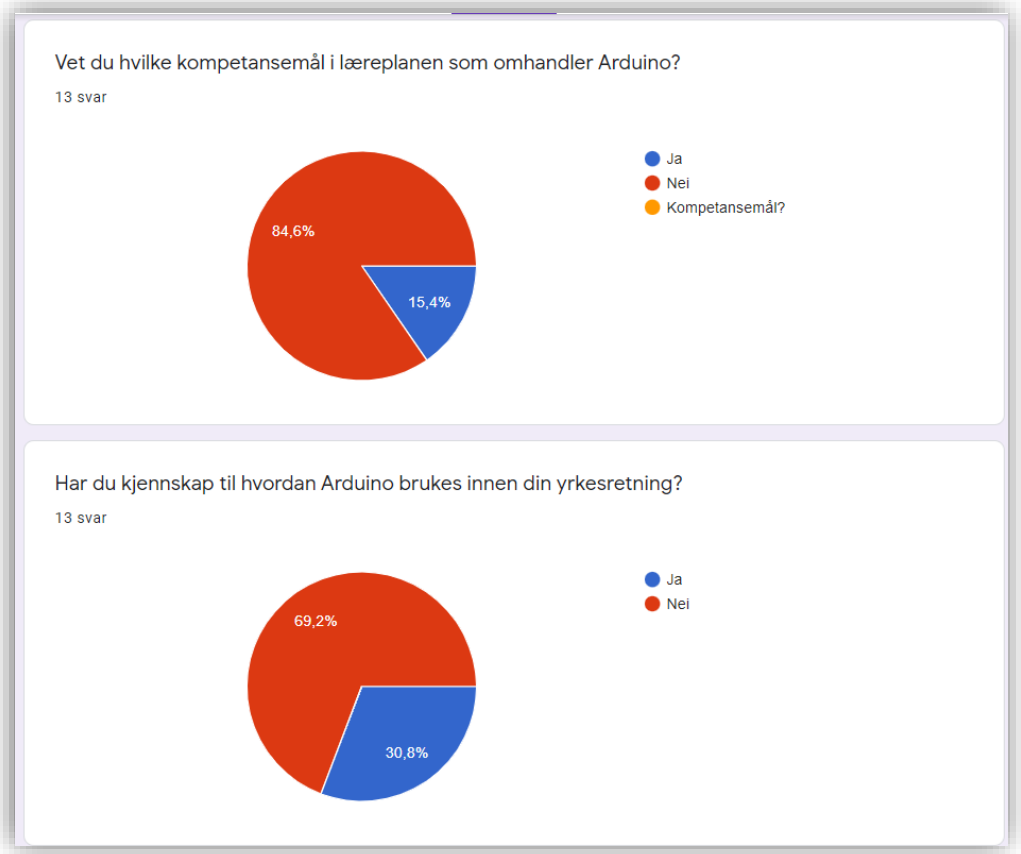
Venås, H. & Vangsnes, S. (2020). *Elektroniske kretser og nettveerk*. Elforlaget.

Wikipedia. (2022, 24. feb. 2022). *Arduino historie*. Wikipedia. Hentet 05.04.22 fra
https://no.wikipedia.org/wiki/Arduino

Vedlegg 3 – Data fra spørreundersøkelsen

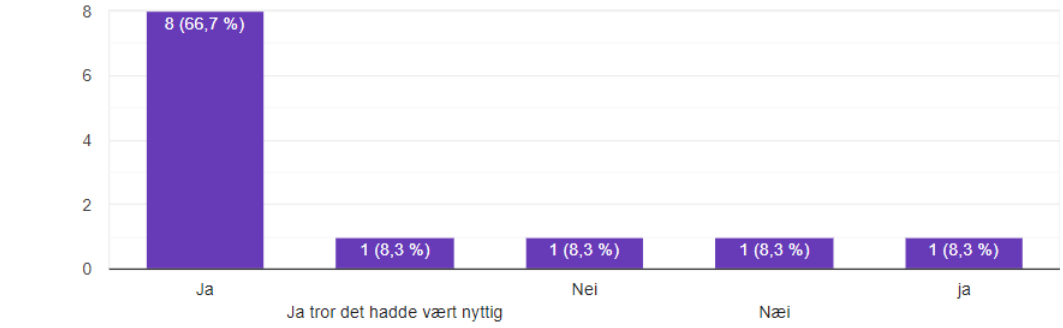
Sammendrag fra før aksjoneringen





Tror du det ville vært nyttig med et kompetansehefte, hvor både pensum, oppgaver med forskjellige nivåer og løsningsforslag har vært samlet på 1 plass?

12 svar

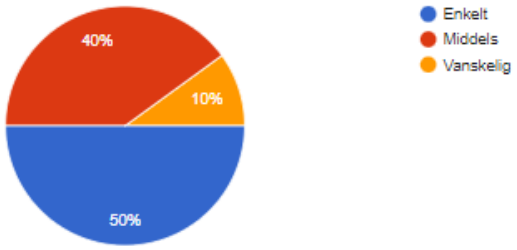


Sammendrag fra etter aksjoneringen

Hvordan opplevde du å finne informasjon til oppgaven som du/dere valgte å gjennomføre?

Kopiér

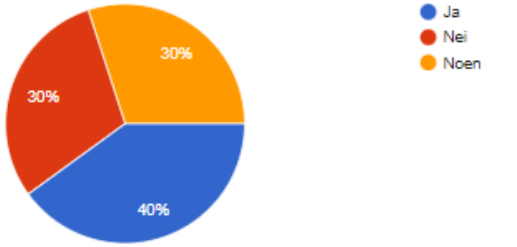
10 svar



Vet du nå hvilke kompetansemål i læreplanen som omhandler Arduino?

Kopiér

10 svar



Ser du nå nytten av å lære Arduino, og er det relevant for ditt yrkesvalg i fremtiden?

Kopiér

10 svar

Hvordan opplevde du å bruker kompetanseheftet?

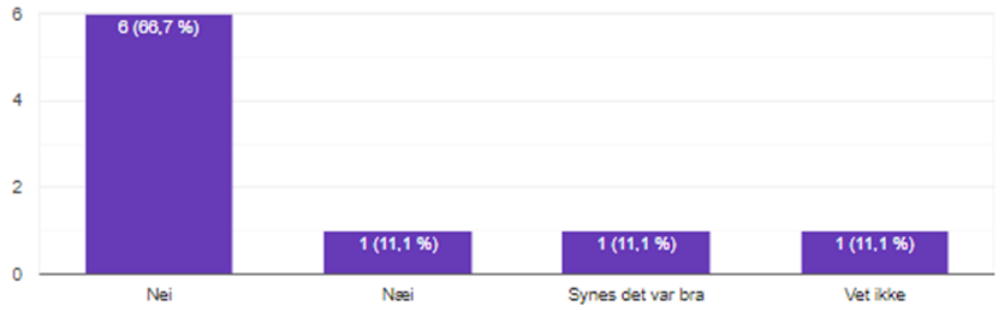
10 svar

- Greit
- Bra
- Litt forvirrende
- Det var bra
- Nyttig
- Lett
- Bra, oversiktelig
- Bra

Har du noen forslag til forbedringer vi kan gjøre med heftet?

Kopiér

9 svar



Kandidatnummer: 10010, 10016 og 10017

Læring om Arduino på Vg1 EL og TIF

NTNU
Norwegian University of
Science and Technology